SEGMENTATION-FREE WORD SPOTTING WITH BAG-OF-FEATURES HIDDEN MARKOV MODELS

Dissertation

zur Erlangung des Grades eines

Doktors der Ingenieurwissenschaften

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

LEONARD ROTHACKER

Dortmund

2019

Tag der mündlichen Prüfung:	17.06.2019
Dekan:	Prof. Dr.-Ing. Gernot A. Fink
Gutachter:	Prof. Dr.-Ing. Gernot A. Fink Prof. Dr. Josep Lladós

ACKNOWLEDGMENTS

This work would not have been possible without the people that helped me to overcome the challenges that I was facing in my research and with whom I could share the moments of success.

First, I would like to thank Prof. Dr.-Ing. Gernot A. Fink for his supervision. Gernot, thank you for your support, your inspiration and the freedom that you gave me for pursuing this work. Our countless discussions were invaluable for the ideas that are presented in this thesis. Furthermore, I would like to thank Prof. Dr. Josep Lladós. Josep, thank you for writing a review of the thesis and for participating in the doctoral committee. Thank you for hosting me at the Computer Vision Center for a research stay. In this regard, I would also like to thank Dr. Marçal Rusiñol. Thank you both for supporting my research in the field of word spotting and for many inspiring discussions. For being my mentor in the structured doctoral program, I would like to thank Prof. Dr. Gabriele Kern-Isberner. Thank you for your advice, for the interesting discussions and for sharing your perspective on my work. For their participation in the doctoral committee, I thank Prof. Dr. Peter Buchholz and Prof. Dr. Erich Schubert.

I am grateful that I could work alongside enthusiastic and highly talented people in the pattern recognition group. We shared a working environment that was challenging in the most positive sense. I am happy to consider you guys to be friends. From my former colleagues, I especially thank Dr.-Ing. Sebastian Sudholt, Dr.-Ing. René Grzeszick, Dr.-Ing. Axel Plinge, Fernando Moya, Eugen Rusakov, Philipp Oberdiek, Fabian Wolf and Thorsten Wilhelm. Our monthly team event has been great fun. Basti, I had the best time touring the world with you when we attended conferences and workshops. René, Axel and Basti, thanks for the many brainstorming sessions in front of the whiteboard. Furthermore, I thank Matthias Kasperidus and Julian Kürby for their help with the development of a word spotting demonstrator. I thank Prof. Dr.-Ing. Gernot A. Fink, Thorsten Wilhelm, Dr.-Ing. Pascal Libuschewski, Dr. Edgar Rothacker and Thekla Seidler for their valuable comments on this thesis.

I am most grateful to my family and friends for their support, for their interest in my work and for their encouragement to pursue a doctoral degree. In this regard, Thekla made the biggest sacrifices. Thank you so much for your tremendous support.

ABSTRACT

The method that is proposed in this thesis makes document images searchable with minimum manual effort. This works in the query-by-example scenario where the user selects an exemplary occurrence of the query word in a document image. Afterwards, an entire collection of document images is searched automatically. The major challenge is to detect relevant words and to sort them according to similarity to the query. However, recognizing text in historic document images can be considered as extremely challenging. Different historic document collections have highly irregular visual appearances due to non-standardized layouts or the large variabilities in handwritten script. An automatic text recognizer requires huge amounts of annotated samples from the collection that are usually not directly available.

In order to search document images with just a single example of the query word, the information that is available about the problem domain is integrated at various levels. Bag-of-features are a powerful image representation that can be adapted to the data automatically. The query word is represented with a hidden Markov model. This statistical sequence model is very suitable for the sequential structure of text. An important assumption is that the visual variability of the text *within* a single collection is limited. For example, this is typically the case if the documents have been written by only a few writers. Furthermore, the proposed method requires only minimal heuristic assumptions about the visual appearance of text. This is achieved by processing document images as a whole without requiring a given segmentation of the images on word level or on line level. The *detection* of potentially relevant document regions is based on similarity to the query. It is not required to recognize words in general. Word size variabilities can be handled by the hidden Markov model. In order to make the computationally costly application of the sequence model feasible in practice, regions are retrieved according to *approximate* similarity with an efficient model decoding algorithm. Since the approximate approach retrieves regions with high recall, re-ranking these regions with the sequence model leads to highly accurate word spotting results. In addition, the method can be extended to textual queries, i.e., query-by-string, if annotated samples become available.

The method is evaluated on five benchmark datasets. In the segmentation-free query-by-example scenario where no annotated sample set is available, the method outperforms all other methods that have been evaluated on any of these five benchmarks. If only a small dataset of annotated samples is available, the performance in the query-by-string scenario is competitive with the state-of-the-art.

PUBLICATIONS

[RRM+18] E. Rusakov, L. Rothacker, H. Mo, and G. A. Fink. "A probabilistic retrieval model for word spotting based on direct attribute prediction." In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 38–43.

[RSR+17] L. Rothacker, S. Sudholt, E. Rusakov, M. Kasperidus, and G. A. Fink. "Word hypotheses for segmentation-free word spotting in historic document images." In: *Proc. Int. Conf. on Document Analysis and Recognition*. Nov. 2017, pp. 1174–1179.

[SRF17] S. Sudholt, L. Rothacker, and G. A. Fink. "Query-by-online word spotting revisited: using CNNs for cross-domain retrieval." In: *Proc. Int. Conf. on Document Analysis and Recognition*. Nov. 2017, pp. 481–486.

[RF16] L. Rothacker and G. A. Fink. "Robust output modeling in bag-of-features HMMs for handwriting recognition." In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition*. Oct. 2016, pp. 199–204.

[WRF16] C. Wieprecht, L. Rothacker, and G. A. Fink. "Word spotting in historical document collections with online-handwritten queries." In: *Proc. Int. Workshop on Document Analysis Systems*. Apr. 2016, pp. 162–167.

[RF15] L. Rothacker and G. A. Fink. "Segmentation-free query-by-string word spotting with bag-of-features HMMs." In: *Proc. Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 661–665.

[RFM+15] L. Rothacker, D. Fisseler, G. G. Müller, F. Weichert, and G. A. Fink. "Retrieving cuneiform structures in a segmentation-free word spotting framework." In: *Proc. Int. Workshop on Historical Document Imaging and Processing*. Aug. 2015, pp. 129–136.

[SGR+15] J. Strassburg, R. Grzeszick, L. Rothacker, and G. A. Fink. "On the influence of superpixel methods for image parsing." In: *Proc. Int. Conf. on Computer Vision Theory and Applications*. Mar. 2015, pp. 518–527.

[SRF15] S. Sudholt, L. Rothacker, and G. A. Fink. "Learning local image descriptors for word spotting." In: *Proc. Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 651–655.

[FRG14] G. A. Fink, L. Rothacker, and R. Grzeszick. "Grouping historical postcards using query-by-example word spotting." In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2014, pp. 470–475.

[RRL+14] L. Rothacker, M. Rusiñol, J. Lladós, and G. A. Fink. "A two-stage approach to segmentation-free query-by-example word spotting." In: *manuscript cultures* 1.7 (2014), pp. 47–57.

[SRF14] S. Sudholt, L. Rothacker, and G. A. Fink. "Kernel density estimation for post recognition score analysis." In: *Proc. German Conference on Pattern Recognition*. Lecture Notes in Computer Science. Sept. 2014, pp. 593–603.

[ARF+13] I. Ahmad, L. Rothacker, G. A. Fink, and S. Mahmoud. "Novel sub-character HMM models for Arabic text recognition." In: *Proc. Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 658–662.

[GRF13] R. Grzeszick, L. Rothacker, and G. A. Fink. "Bag-of-features representations using spatial visual vocabularies for object classification." In: *Proc. Int. Conf. on Image Processing*. Sept. 2013, pp. 2867–2871.

[RFB+13] L. Rothacker, G. A. Fink, P. Banerjee, U. Bhattacharya, and B. B. Chaudhuri. "Bag-of-features HMMs for segmentation-free Bangla word spotting." In: *Proc. Int. Workshop on Multilingual OCR*. Aug. 2013, pp. 1–5.

[RRF13] L. Rothacker, M. Rusiñol, and G. A. Fink. "Bag-of-features HMMs for segmentation-free word spotting in handwritten documents." In: *Proc. Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 1305–1309.

[RVF12] L. Rothacker, S. Vajda, and G. A. Fink. "Bag-of-features representations for offline handwriting recognition applied to Arabic Script." In: *Proc. Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2012, pp. 149–154.

[VRF12] S. Vajda, L. Rothacker, and G. A. Fink. "A method for camera-based interactive whiteboard reading." In: *Camera-Based Document Analysis and Recognition, Revised Selected Papers*. Vol. 7139. Springer, 2012, pp. 112–125.

[VRF11] S. Vajda, L. Rothacker, and G. A. Fink. "A camera-based interactive whiteboard reading system." In: *Proc. Int. Workshop on Camera-Based Document Analysis and Recognition*. Sept. 2011, pp. 91–96.

CONTENTS

ACRONYMS

BoW	bag-of-words
BoF	bag-of-features
CC	connected component
CNN	convolutional neural network
DCM	Dirichlet compound multinomial
DTW	dynamic time warping
EDCM	exponential Dirichlet compound multinomial
EM	expectation maximization
ER	extremal region
GMM	Gaussian mixture model
HMM	hidden Markov model
HoG	histograms of oriented gradients
IFS	inverted file structure
IoU	intersection over union
LM	language model
mAP	mean average precision
MSER	maximally stable extremal region
NMS	non-maximum suppression
PQ	product quantization
PHOC	pyramidal histogram of characters
RoI	region-of-interest
SIFT	scale invariant feature transform
SC	semi-continuous
SVM	support vector machine
vMF	von Mises-Fisher

$\mathbb{N}_{\geqslant 0}$	Set of non-negative integers.
$\mathbb{Q}$	Set of rational numbers.
$\mathbb{R}$	Set of real numbers.
Θ	A set.
$\lvert\Theta\rvert$	Cardinality of the set Θ.
x	Scalar value.
N	Quantitative value.
$\mathfrak{g}$	Value in image pixels (Fraktur symbol).
$\mathfrak{p}_w$	Value in image pixels referring to width.
$\mathfrak{p}_h$	Value in image pixels referring to height.
$\mathfrak{r}_x$	Value in image pixels referring to horizontal direction.
$\mathfrak{r}_y$	Value in image pixels referring to vertical direction.
T_q	Value that is specific to a qualifier (Fraktur subscript).
ϵ_{low}	Value that is specific to a qualifier (text subscript).
$[u_{nt}]$	Matrix with element at row-index n and column-index t.
$[u_{nt}]^\top$	Transpose of a matrix such that $[u_{nt}]^\top = [u_{tn}]$.
$\mathbf{x}$	Column vector.
$\mathbf{x}^\top$	Transpose of vector $\mathbf{x}$, i.e., a row vector.
x_k	Element at index k of vector $\mathbf{x}$.
$\mathbf{x}_t$	Vector at index t.
x_{tk}	Element at index k of vector $\mathbf{x}_t$.
$\mathbf{O}$	Sequence, e.g., of vectors: $(\mathbf{x}_0, \ldots, \mathbf{x}_T)$.
$\mathbf{O}_r^{[l]}$	Sequence at row-index l and column-index r.
$\mathbf{O}_{r \mid q}^{[l]}$	Subsequence at index (l, r), conditioned on qualifier q.
$\mathbf{x}_t^{[l]}$	Vector at row-index l and column-index t.
$x_{tk}^{[l]}$	Element at index k of vector $\mathbf{x}_t^{[l]}$.
$x!$	Factorial of x, i.e., $x! = \prod_{i=1}^{x} i$, if $x \in \mathbb{N}_{>0}$ and $0! = 1$.
$\lVert\mathbf{x}\rVert_p$	p-norm of vector $\mathbf{x}$, i.e., $\lVert\mathbf{x}\rVert_p = \left(\sum_{k=0}^{D-1} \lvert x_k\rvert^p\right)^{\frac{1}{p}}$, $\mathbf{x} \in \mathbb{R}^D$.
$\lfloor x \rfloor$	Integer part of x such that $x - 1 < \lfloor x \rfloor \leqslant x$, $x \in \mathbb{R}$.
$g(\cdot)$	A function.
$p(\cdot)$	Probability mass function or probability density function.
$\mathcal{M}$	Discrete random variable.
$\Omega_{\mathcal{M}}$	Set of elementary events for the random variable $\mathcal{M}$.

1 INTRODUCTION

Searching document collections for occurrences of query words automatically is important for the analysis and interpretation of their contents. It is a widely used standard-functionality for digital documents containing machine-readable text. Unfortunately, this functionality is not directly available for document images. In order to be processed with automatic search queries in the same way, it would be required to transcribe the document images into machine-readable textual representations first. This can be difficult for non-standardized documents and is, therefore, costly and error-prone in these cases. Different non-standardized document collections are highly variable in their visual appearance and cannot be automatically transcribed with off-the-shelf optical character recognition software. Setting up a full transcription recognizer requires huge amounts of annotated sample data that is representative for the application domain. Such an annotated sample set typically consists of machine-readable transcriptions of document images on line level. For non-standardized documents this is not directly available. Obtaining such a sample set partly solves the original problem which diminishes the benefit of applying full transcription recognizers.

Word spotting methods allow for searching document images without requiring a full transcription. Especially for handwritten and historic documents this is most relevant. Their otherwise manual exploration is a laborious and time consuming effort. In contrast to searching in machine-readable textual representations for occurrences of query words, word spotting searches document *images* based on visual appearance. This makes word spotting a specialized image retrieval technique with a number of desirable properties:

- Word spotting is very flexible with respect to its applicability in practice. It makes the most out of its given scenario, even if just the document images without any additional annotations are available. One focus of this thesis are segmentation-free approaches that process entire document images without requiring a given segmentation on word or line level.

- Word spotting is robust with respect to retrieval errors. Search results are typically presented to the user in a list ranked according to similarity to the query. As long as the relevant results are among the top results they are still useful to the user. The method presented in this thesis achieves highly accurate retrieval results. In the scenario where no annotated sample set is available, state-of-the-art methods are largely outperformed.

- Word spotting is fast. This is achieved by computing index representations for document images. Through the index, the computational effort can be reduced at query time. In this thesis, a two-stage method is proposed. Potentially relevant document image regions are detected fast and analyzed in more detail afterwards.
- Words can be queried in different input modalities depending on the requirements of the users. Each modality has its advantages and disadvantages. In this thesis, word image queries and textual queries are considered. These are the most relevant query modalities in practice.

This is achieved by using bag-of-features with hidden Markov models in an integrated word spotting framework. The *bag-of-features* (BoF) is an image representation that can automatically be adapted to the visual characteristics of an image dataset [OD11]. This makes BoF very suitable for historic document collections that have very diverse visual appearances. The *hidden Markov model* (HMM) is a statistical sequence model, which allows for modelling the length variabilities that are typically found in handwritten texts [PF09]. It is used for representing the query. In order to search documents, it is *not* required to split the document images in word segments or in line segments first. Therefore, the method is classified as *segmentation-free*. Along with these aspects that are important for the applicability in practice, different possibilities for integrating BoF in the statistical HMM process will be discussed in this thesis.

The rest of this chapter is structured as follows. Section 1.1 motivates word spotting and its common applications. Word spotting is presented in the context of document analysis showing how it is able to bridge the gap between manual document *image* exploration and manual or automatic analysis based on full document image transcriptions. The architecture of a typical word spotting system and its most common variants are introduced in Section 1.2. In this regard, Section 1.3 puts this thesis in context with word spotting approaches in general and highlights its specific contributions to the field. The introduction concludes with an outline of the structure of the thesis in Section 1.4.

1.1 MOTIVATION

Word spotting is part of the document analysis research area. A major objective in document analysis is to support work on document images with methods for automatic information extraction. In this regard, many methods deal with text. Given a document image, an ideal result would be a full text transcription along with annotations of the exact occurrences of characters and words. For standard-

ized documents considerable progress towards this objective has been made in recent years. Optical character recognition software achieves high recognition rates for high quality printed documents [Smi07]. This is mainly due to the limited variability in the visual appearance of modern fonts, high quality output of modern printers and scanners and standardized document layouts. However, as soon as the variability increases, results deteriorate. This is especially problematic for handwritten and historic documents, cf. [LRF+12]. The visual appearance of handwritten script is extremely writer dependent. Historic documents contain degradations that are due to aging, storage or low quality of ink and paper. The practical consequence is that, compared to the possibilities available for modern documents, the possibilities for automatically analyzing handwritten and historic documents are rather limited. For this reason, the continuously growing digital archives of historic document images are not as accessible as modern documents. Considering that these digital document archives are becoming widely available through the Internet, cf. e.g., [Bal05], this aspect becomes even more important.

The work of these digital libraries is extremely relevant. They preserve mankind's cultural heritage and make it available to the public. History repeats itself and one can only learn for the future when being aware of what happened in the past. Through digital archives history becomes vivid and is likely to reach a much broader audience besides the expert community. An example is the *Library of Congress* in Washington, DC, USA. Various historic document collections are directly accessible online[1].

Towards supporting historic document image exploration with automatic methods, word spotting plays a crucial role. In order to recognize or retrieve text, annotated sample data is required in any case. This is generally a problem, as annotated sample data is hard to obtain. It usually has to be created manually. Methods requiring large amounts of annotated data, like full transcription recognizers, have another disadvantage in this regard. Historic document collections are very particular in their visual appearance. Unlike for modern documents, an annotated sample set for one document collection might not be usable for creating a recognizer for another historic document collection. The manual annotation has to be started over and over again. If a lot of effort has to be put in the manual transcription before methods for automatic processing become available, it might be more efficient to directly organize the manual transcription of the entire document collection [CT14].

Word spotting offers a compromise where the requirements with respect to annotated samples are very flexible. The possibilities go from word spotting systems working with synthesized annotated samples, thus, avoiding any manual labeling effort, over systems that

1 `https://www.loc.gov/collections/`, accessed on July 13, 2019.

just require a single annotated sample of the query word to models that have to be estimated with large volumes of annotated word images. If word spotting is based on a single annotated example of the query word, document images with similar visual appearance can be searched. If more annotated samples are available, these can be used for improving retrieval performance. However, this assumes that the annotated samples are representative for the text in the document images. This is a challenge for synthesizing sample data, in particular.

Besides these prerequisites, the input and output of a word spotting system offer advantages to users. While searching in a transcript limits the input to textual queries, word spotting allows for a full bandwidth of modalities, including text, word images, speech [RAT+15b] or handwriting [WRF16; SRF17]. The latter can, for example, be useful for documents written in non-standard scripts, cf. [BHM16].

In a similar manner, the output of a word spotting system offers a lot more flexibility in contrast to the explicit transcription provided by a recognizer. By acknowledging that an automatic system will typically not be able to produce perfect results, the user can interactively be integrated in the document analysis process [RL14]. This makes word spotting robust with respect to retrieval performance. While errors in full transcriptions diminish the value of the overall result considerably, errors in retrieval systems mainly affect the ranking of the retrieval list. As long as the relevant results are among the top results, the user decides what is correct. In case of historic documents there is a further aspect that has to be taken into account. Different historians might interpret the same passage in a document very differently. This also means that there might not be a single "perfect" transcription. Word spotting leaves the interpretation to the experts and helps with finding document regions that are *potentially* relevant to the query. Therefore, the capabilities and limitations of word spotting methods are transparent. This increases the acceptance of the experts. It has to be noted that in the age of the Internet users are generally very familiar with the benefits of retrieval search engines.

Finally, word spotting can also be seen as a useful tool for obtaining a full transcription recognizer. Due to the advantages discussed above, a word spotting system can be applied on the document images even at a stage where no annotations are available at all. By collecting the retrieval results, an annotated sample set can be built. With this growing sample set the word spotting system can be refined until a sufficient number of high quality annotations is available. This can be the basis for estimating a full transcription recognizer, thus, allowing for a smooth transition from searching to transcribing.

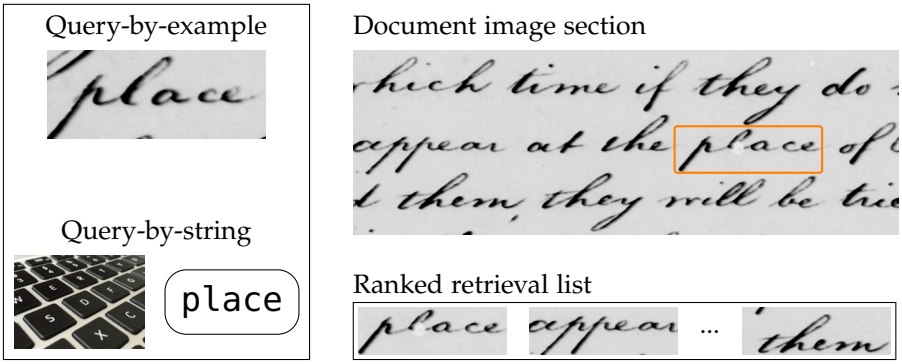


Figure 1: Word spotting system overview. On the left two different modalities for the same query word `place` are shown. The search result is shown on the right. The only relevant region in the document image section shown in this example has been highlighted with an orange frame. In the ranked retrieval list it appears first. It is followed by document image regions in descending order of similarity to the query. Document images, that are shown in this figure and in the following figures, originate from the *George Washington papers* [Was54], unless noted otherwise.

1.2 WORD SPOTTING

The input of a word spotting system consists of document images and a query. The output is a list of document image regions that is typically ranked according to similarity to the query. Figure 1 shows the overall process with two query modalities. In the *query-by-example* scenario, the query is given as an exemplary occurrence of the query word. This means that the user has to locate and select an instance of the query. In the *query-by-string* scenario, the user can enter a textual query on a keyboard. Figure 1 shows a result where the only relevant region in the document image section is highlighted and appears in the ranked retrieval list first.

Consequently, the basic structure of a word spotting system follows the structure of an information retrieval system. Initially, a database of document images is given. For making the document images searchable, image regions are represented in terms of *features*. In this regard, features are numerical representations that contain information which is relevant for word image retrieval. Whether the query is given by-example or by-string, a query model is obtained that can be evaluated with the image representations from the document image database. For query-by-example, it can be sufficient to model visual appearance on word level. Modelling visual appearances of characters is an approach for the query-by-string scenario. Scores that represent similarities between the query and the document image regions are used for generating the ranked retrieval list.

For detecting *relevant* document image regions, it is helpful if words can be identified in the document prior to retrieval. This is usually

the case for printed modern documents. Texts printed in standardized fonts can heuristically be *segmented* into words in many cases. For example, based on the assumption that gaps between words are larger than gaps between characters within words. However, particularly for handwritten and historic documents this often fails. In these cases, the visual variability of text can be substantial even for a single writer. If the entire retrieval pipeline is built on segmented regions, segmentation errors will also lead to errors in the retrieval result.

Segmentation-free methods approach this challenge in an integrated manner. Potentially relevant document image regions are obtained *within* the word spotting process, depending on similarity to the query. Thus, the regions should cover the relevant occurrences of the query in the document images, but they do *not* have to correspond to word segments in general. In contrast, *segmentation-based* methods assume a *given* segmentation of the document into words. Document image regions are usually defined as rectangular bounding boxes.

The development of segmentation-free word spotting methods has mainly been influenced by techniques that have been successful for handwriting recognition and computer vision. With respect to document image representations, methods from computer vision have a strong influence. In order to cope with the great variability found in natural scene images, feature representations that are *automatically adaptable* to the problem domain have been very successful. For this purpose, sample data from the problem domain is required. Furthermore, it is important whether the data is also annotated with respect to the classes that the samples are associated with. A powerful image representation that does not require annotations is the *bag-of-features* (BoF) [OD11]. In computer vision, it has been successfully applied to object [CDF+04] and scene categorization [FP05] as well as large scale image retrieval [SZ03]. The last scenario is particularly related since efficient retrieval is approached by indexing BoF representations. For these reasons, BoF is also suitable for word spotting in historic document images. In this scenario, large archives have to be searched, characteristics of different document collections are very diverse and hardly any annotated samples are available [SJ12; RRL+14; RAT+15a]. If large amounts of annotations are available, *convolutional neural networks* (CNNs) strongly improve recognition and retrieval performance, e.g., for object recognition [CSV+14] and word spotting [SF16].

In the field of handwriting recognition, statistical sequence models, like *hidden Markov models* (HMMs) [PF09; KDN13] or recurrent neural networks [FU15], are the predominant methods. They are well suited for modeling the *sequential* structure of text. Furthermore, they can handle the great variability in human writing. By representing a line image as a sequence of feature vectors, each element of the sequence can be associated with different classes, e.g., characters. In this way, recognition and segmentation of the text line are solved jointly.

This has also been successfully applied for spotting words within text lines [FKF+12; FFM+12]. Furthermore, HMMs offer great flexibility with respect to the amount of annotated sample data which is required for model estimation. In the *semi-continuous* HMM framework, features are modeled with a single probabilistic mixture model which is shared among all HMM states. In this way, the mixture model can be estimated without annotated data, i.e., in an unsupervised manner. HMM state-dependent parameters can be estimated from few annotated samples [PF05]. A single annotated sample can be sufficient in order to estimate a query word HMM for word spotting [RP09b].

1.3 CONTRIBUTIONS

The main contributions of this thesis are an integration of BoF representations and HMMs as well as their efficient application to segmentation-free query-by-example and query-by-string word spotting. A first approach in this direction was made in the Diploma thesis [Rot11] where an integration of BoF sequences and HMMs was introduced for handwriting recognition. The most important results from [Rot11] have been published in [RVF12]. Building on this work, the application of BoF-HMMs to a different challenging application domain and an extensive study of BoF output models for HMMs are presented here. The methods are evaluated on established word spotting benchmarks. Figure 2 shows a schematic visualization of the word spotting system and highlights the different contributions.

The methods presented in this thesis have partially been published at scientific conferences of the document analysis community. In the following, the contributions are discussed in the context of these publications. The research has been carried out in the *pattern recognition group* at TU Dortmund University under the supervision of Prof. Dr.-Ing. Gernot A. Fink. In the early stages, the application of BoF-HMMs in a patch-based word spotting framework has been supported by Dr. Marçal Rusiñol. Based on the HMM toolkit *ESMERALDA* [FP08] and the computer vision toolkit *VLFeat* [VF08], the author of this thesis implemented the word spotting method in Python and C++ and performed all the word spotting experiments unless noted otherwise. In addition to these basic contributions, important methodological contributions will be outlined along with the corresponding publications.

BoF-HMMs have been presented for segmentation-free query-by-example word spotting for the first time in [RRF13]. Incorporating ideas from [RVF12] and [RAT+11], BoF-HMMs have been used for modeling query word images. The visual similarity of the query with document image regions has been evaluated in a patch-based framework. Highly accurate word spotting results could be achieved. Only a single exemplary word image of the query was available for model estimation. Despite the high computational complexity of HMM decod-

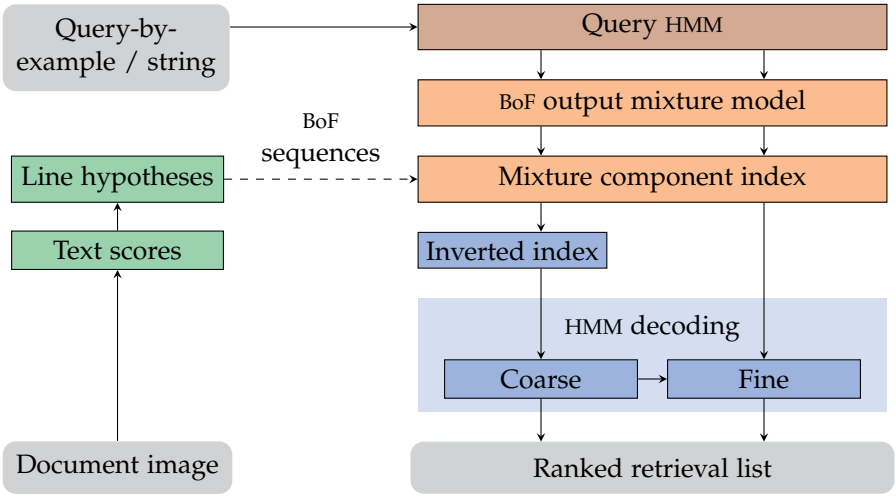


Figure 2: Schematic visualization of the word spotting system presented in this thesis. Input and output nodes are shown in gray with rounded corners. Following the arrows from the input nodes, the output node can be reached. The main contributions are indicated with different colors. Brown refers to HMM modeling. Green refers to text and line hypotheses generation. Orange refers to the output model integration of HMMs and BoF sequences obtained from line hypotheses. This association between the green line hypotheses node and the orange output modeling node is indicated by a dotted line. Blue refers to contributions that are specific to segmentation-free decoding in the two-stage patch-based framework.

ing, their applicability in a patch-based segmentation-free framework was shown. In Figure 2, this is denoted as a query-by-example scenario that makes use of BoF-HMMs and ends in a *fine* analysis of the document image. Important contributions in [RRF13] are the dynamic adaptation of the patch sampling rate as well as the incorporation of BoF vectors at the left-side context and the right-side context of the query word image.

Query-by-example word spotting with BoF-HMMs has successfully been applied to document collections with very different characteristics, such as printed Bengal documents [RFB+13], historic postcards written in German Kurrent [FRG14] and rendered images of 3D cuneiform tablets [RFM+15]. The rendered cuneiform images were provided by the co-author Denis Fisseler. The author of this thesis contributed to the definition of the word spotting benchmarks in all three publications.

In order to improve the applicability of the word spotting method in large scale scenarios, a two-stage extension has been proposed in [RRL+14]. For rapidly searching high volumes of document images, it is important to build index structures allowing for retrieving potentially relevant document image regions instantly. In the patch-based framework, HMM scores need to be computed for each patch independently such that the application of indexing strategies is not

straight forward. In [RRL+14], a method for indexing document image regions based on their BoF representations along with a voting approach for ranking these regions according to similarity to the query has been presented for segmentation-free word spotting. In combination with BoF-HMMs, potentially relevant document image regions can be retrieved fast and then refined with the more accurate statistical sequence model. Since both stages operate on the same features, image regions that have been discarded in the first stage are unlikely to obtain good similarity scores in the second stage. In Figure 2, the first stage is denoted as *coarse* analysis. The arrow to *fine* analysis indicates that both analysis stages are intended to be used jointly. The most important contributions in [RRL+14] are the voting algorithm in the coarse analysis stage and the integration of both stages in the same patch-based decoding framework, which have been developed under the guidance of Dr. Marçal Rusiñol and Prof. Dr. Josep Lladós.

If annotated sample data is available, this allows for supporting query-by-string. HMM character models can be estimated and combined dynamically for creating query models. In [RF15], the first patch-based method for segmentation-free query-by-string word spotting was proposed. The analysis patch size has been generated from character size estimates. In addition to achieving competitive retrieval results, the robustness of the method was demonstrated with respect to limiting the amount of annotated sample data. Figure 2 shows that the two-stage segmentation-free framework is independent of the query modality. In both cases, BoF-HMM query models can be used. The most important contribution in [RF15] is the use of a large inventory of context-dependent character models with limited annotated training material.

In [RF16], different approaches for modeling BoF in the statistical HMM process have been discussed in the context of text categorization, handwriting recognition and word spotting. In this regard, two methods for integrating BoF and HMMs were evaluated. The pseudo-discrete approach, presented in [RVF12], has been compared with a mixture model from probabilistic text clustering [Elk06]. A distinguishing property of the two models is their capability of representing BoF representations. In contrast to the pseudo-discrete model, the mixture model from probabilistic text clustering is very suitable for the characteristics of the BoF representations in the given scenario. This was demonstrated in the evaluation of [RF16]. Although the pseudo-discrete model suffers from the possibility of degenerated cases, it showed very good generalization capabilities for word spotting. In Figure 2, the *output model* is shown as the element bridging the HMMs and the sequences of BoF representations. Important contributions in [RF16] are the analysis of degenerated cases of the pseudo-discrete model and the selection of the probabilistic mixture model from [Elk06] for the application in the given scenario.

Furthermore, considerable advancements of these methods are presented in this thesis. This is possible through the incorporation of text hypotheses. Text hypotheses are obtained with a method that has been presented for word hypothesis generation in [RSR+17]. Inspired by *maximally stable extremal regions* [MCU+04], the author of this thesis proposed a method for hypothesis generation based on text detector scores and word detector scores. The co-authors Sebastian Sudholt and Eugen Rusakov contributed CNNs for computing word detection scores. Sebastian Sudholt also contributed the CNNs for representing word hypotheses with *pyramidal histogram of characters* (PHOC) attributes, i.e., the PHOCNET. While the author of this thesis largely contributed to the design of the segmentation-free word spotting *system*, the system implementation and execution of experiments has been carried out by co-author Matthias Kasperidus.

Using text hypotheses, a *generalization* of the indexing strategy to features that are modeled with *semi-continuous* (SC) HMMs is proposed in this thesis. For this purpose, line hypotheses are derived from text hypotheses. For all line hypotheses, BoF sequences are extracted and *mixture component probabilities* are stored in a *look-up table*. For coarse analysis, potentially relevant document regions are obtained by *inverting* this *index*. Their similarity to the query is computed with a probabilistic voting scheme. For fine analysis, an *extended* patch-decoding strategy is proposed that allows for localizing words highly accurately. The extended strategy includes alignments of patches with a background model, whitespace models and the query model. The extended patch-decoding strategy allows for improving word spotting performance considerably. Furthermore, the patch coordinates can be snapped to text hypothesis coordinates in their local neighborhood.

All of these contributions are evaluated in an integrated framework. The possibilities for supporting historic document exploration are demonstrated. In comparison to segmentation-free query-by-example word spotting methods that do not require annotated samples besides the query word image, state-of-the-art results are outperformed by a large margin on five datasets of historic document images. Regarding the applicability in practice, the most important contribution is that only a single meta parameter requires an adjustment for applying the proposed method to a new document collection. This parameter can be estimated from the document images automatically. Therefore, the method is perfectly suited for historic document collections that are still unexplored or where transcriptions and annotations are not available in machine-readable form.

1.4 STRUCTURE

The thesis is structured as follows. Chapter 2 presents the pattern recognition methods which are the foundation for segmentation-free

word spotting with BoF-HMMs. Chapter 3 discusses related word spotting methods, focussing on the recent state-of-the-art as well as the techniques that have been influential for the development of the methods presented in this thesis. The proposed word spotting method is presented in Chapter 4. The contributions that are highlighted in Section 1.3 are explained in detail and discussed in a comprehensive evaluation in Chapter 5. The proposed method is differentiated from the state-of-the-art, showing limitations and advantages. A conclusion is drawn in Chapter 6.

Supplemental material is provided in three appendices. Appendix A adds experimental evaluations of BoF output models. Formal derivations for the BoF output model that has been proposed in [Elk06] are presented in Appendix B. Finally, a comprehensive categorization of word spotting methods can be found in Appendix C.

2 FUNDAMENTALS

Word spotting with BoF-HMMs is mostly based on methods from automatic image retrieval and handwriting recognition. An important property that these methods share is that they do not require large amounts of annotated samples. Heuristic methods are *not* estimated from sample data. They are successful due to their expert design. Unsupervised methods are estimated from sample data without requiring annotations. Their purpose is to group features according to higher level components. Supervised methods require annotated samples for model estimation. In combination with the unsupervised models, the annotations allow for associating the higher level components with semantic units, i.e., classes. The supervised method, that is presented here, can be estimated from a single annotated sample.

Furthermore, the parameterization of these methods is robust such that they work in different application scenarios and on different datasets without requiring extensive manual parameter tuning. The concepts and methods that will be introduced in the following demonstrated these characteristics in the application to different pattern recognition tasks:

- *Regions-of-interest* (RoIs) are required in order to focus the analysis on potentially relevant image areas. The *maximally stable extremal region* (MSER) detector is a heuristic method that has been applied in different computer vision tasks including text detection in natural scene images (Section 2.1).

- In the context of image retrieval, the computation of visual similarities is based on low-level image representations. The *scale invariant feature transform* (SIFT) descriptor is among the most widely used heuristic features in computer vision (Section 2.2).

- Based on low-level features, the *bag-of-features* (BoF) is a famous mid-level feature representation. Its ability for unsupervised adaptation to the problem domain made it a standard representation for image retrieval (Section 2.3).

- Mixture models represent the feature vector distribution based on a given number of probabilistic mixture component distributions. After unsupervised model estimation, the components represent typical feature vector variations. The *multinomial mixture model* has been used for text classification with bag-of-*words* as well as for clustering images according to visual similarity. It is a foundation for modeling BoF with HMMs (Section 2.4).

- *Hidden Markov models* (HMMs) allow for modeling the sequences of feature vectors that are typical for given classes, e.g., characters and words. Probabilistic mixture models are required for this purpose. If all mixture models in an HMM share the same components and only differ in the mixture proportions, the HMM is called *semi-continuous* (SC). SC-HMMs allow for estimating a model from only a single sample (Section 2.5).

Consequently, word spotting with BoF-HMMs follows a classic pattern recognition pipeline. RoIs are represented with sequences of BoF feature vectors and an HMM models the sequence of BoF vectors that is typical for the query word. Since BoF-HMMs operate in the segmentation-free scenario, it is important to note that RoIs represent alternatives to each other and not a *segmentation* of the document image into lines or words.

2.1 MAXIMALLY STABLE EXTREMAL REGIONS

Maximally stable extremal regions (MSER) is a method for detecting high-contrast regions based on image intensity values [MCU+04]. It has been proposed in the context of obtaining point-wise correspondences between images which show the same scene from different viewpoints [MCU+04]. Based on these correspondences, the viewpoint transformation can be derived. Finally, this allows for a 3D reconstruction of the scene. The detected image regions are matched with each other in order to obtain correspondences. For this purpose, the regions should have distinguishable visual properties such that they can be detected reliably and repeatably from different view points and under different illumination. The most important assumption in the MSER method is that such regions are *extremal*.

Extremal regions (ERs) are defined such that all pixels within the region have an intensity value that is larger (or smaller) than the intensity values of the pixels in the outer region boundary. If the intensity values within the region are larger than the intensity values in the outer boundary, the region is referred to as *maximal intensity region*. Otherwise, i.e., if the intensity values within the region are smaller than the intensity values in the outer boundary, the region is referred to as *minimal intensity region*. In order to limit the number of detections, an ER has to be *maximally stable*. Informally this means that the *intensity difference* of the extremal region with respect to its outer boundary should be large.

The key idea for computing ERs is to threshold, cf. [GW02, p. 77], an image at *all* intensity levels. For 8 bit intensity values this produces 256 binary images. Binary images are obtained such that all pixels with an intensity value below the given threshold are set to zero and to one otherwise. Thus, fewer pixels are set to zero for lower threshold values than for higher threshold values. Afterwards, *connected compo-*

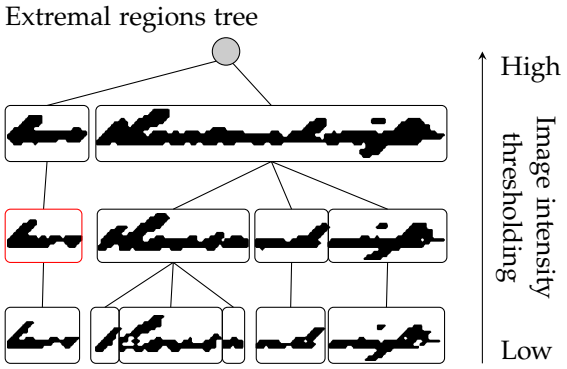


Figure 3: Maximally stable extremal regions are detected in an ER tree. For this purpose the figure shows a conceptual visualization of a few ERs which have been extracted from a small section of a document image at three binarization thresholds. The tree root is indicated as a gray dot. The ERs are minimal intensity regions such that they represent the pen-stroke which typically has a dark visual appearance. The concept of maximal stability is indicated by the MSER which is highlighted in red. For the three binarization thresholds it changes minimally with respect to the ERs on the same tree path.

nents (CCs) are extracted from all binary images. For *minimal* intensity regions, a CC is a subset of pixels with value *zero* that are connected through their 8-neighborhood of surrounding pixels, cf. [GW02, Sec. 2.5.2]. These CCs are extremal because all intensity values within a CC are smaller than the intensity values surrounding the CC. Therefore, CCs can only grow from lower to higher thresholds. This way, a relation is defined among CCs over different thresholds which globally results in a tree, see Figure 3. The criterion for *maximal stability* is defined along any path from any tree leave to the tree root. For a given path, the relative change in the number of pixels in a CC is computed as a function of the tree level. An MSER is detected for each CC on any path where the function reaches a local minimum. The sensitivity of the detector can be controlled through the CCs which are incorporated for computing the relative CC change at a tree level. Instead of considering CCs at the tree level below and above the given level, i.e., with an offset of one, the offset can be increased for making the detector less sensitive. Figure 3 shows a conceptual visualization of an ER tree for a small section of a document image.

Apart from detecting regions for obtaining point-wise correspondences, selecting ERs according to maximal stability is a heuristic that has also been applied to text detection in natural scene images, cf. e.g., [DBW08]. However, the vast majority of MSER detections in a natural scene image will not be containing text and has to be filtered in a post processing step, e.g., by using a classifier. For this reason, it is more efficient to directly integrate the classifier in the region generation process, resulting in *category-specific extremal regions* [MZ05]. The key idea is to represent each ER from the ER tree with a numerical feature

representation. Based on the classification result, an ER is accepted or rejected as a detection. This way, the heuristic maximal stability criterion can be replaced with a decision of a *learned* classifier. Most importantly, this implies that annotated samples are available for estimating the classifier. A combination of both approaches is presented in [NM16]. Instead of accepting or rejecting ERs directly, the maximal stability criterion is applied on the classifier scores. In comparison to the direct classification approach, this mostly avoids redundant detections. It is important to note that the detected text regions do not necessarily correspond to words and have to be grouped by their distances. This is an important aspect in the context of document images. Since these images *mostly* contain text, the challenge is to detect accurate word boundaries rather than appearances of text in general.

2.2 SCALE INVARIANT FEATURE TRANSFORM

Scale invariant feature transform (SIFT) is a method for extracting local image features [Low04]. For this purpose, it consist of an interest point detector and an interest point descriptor. The descriptor is a feature vector that represents the local image neighborhood around the interest point. It has been proposed for obtaining point-wise correspondences in images, e.g., for object recognition [Low04]. The main objective is that the descriptors should be *invariant* to typical image transformations, most importantly rotation and scale. For example, this means that corresponding descriptors which have been extracted from the same object in different rotations should be similar in the descriptor vector-space although their visual appearance differs due to the rotation. The SIFT method achieves this by using the detector and the descriptor in an integrated manner such that the information obtained in the detector stage is used in the descriptor stage. Interest points are also referred to as *keypoints*.

For detection, the image is represented with a difference-of-Gaussian scale space. In the first step, a *Gaussian* scale space is obtained by filtering the image with Gaussian kernels of increasing variance. The process simulates the change in image details if an object is captured at different distances to the camera. The *difference*-of-Gaussian scale space is obtained by subtracting filtered images which are adjacent in the Gaussian scale space. Essentially, this approximates a second-order derivative filter which extracts contour information at different frequency bands. Candidate locations are local extrema in the difference-of-Gaussian scale space. Candidates that are directly located on image edges are suppressed since they are unlikely to be detected repeatably across image transformations. For all remaining candidates, the scale parameter is set to the scale at which it has been detected within the scale space. In addition, orientation parameters are computed with respect to the scales of the candidates. Gradient orienta-

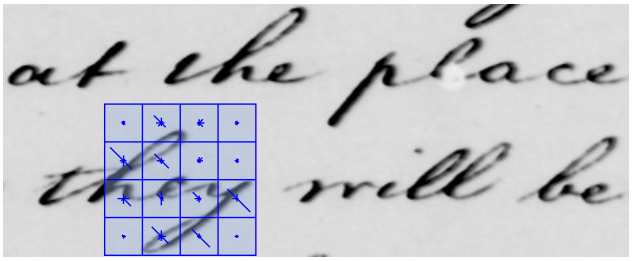


Figure 4: SIFT descriptor. The figure shows a single SIFT descriptor that has been extracted in a section of a document image. The descriptor consists of 4×4 cells, which are indicated in blue. Each cell is represented with a gradient orientation histogram. The 8 histogram orientations, corresponding to $[0, 45, 90, \ldots, 315]$ degrees, are indicated with blue lines within the cells. The lengths of the lines correspond to the weighted and accumulated gradient magnitudes, i.e., the histogram values. In the application to a document image, the descriptor represents image areas with high contrast, i.e., mostly the pen-stroke. It should be noted that the descriptor is larger than in pratice in order to allow for a better visualization.

tions and gradient magnitudes are precomputed for each image from the Gaussian scale space for this purpose. The orientations in the local neighborhood of a candidate location and its scale are obtained with a 36-bin orientation histogram. For this purpose, gradient orientations are quantized and their magnitudes are accumulated to the corresponding histogram bin. Magnitudes are weighted according to their distance to the candidate location with a Gaussian. The detector extracts interest points for all dominant orientations at all candidate locations. Histogram orientations are considered as dominant if their histogram value is within 80% of the maximum histogram value. Interest points are represented with location, scale and orientation.

In the descriptor stage, the local neighborhood of each interest point is represented with a numerical feature vector. A descriptor consists of 16 histograms of 8 orientations. The histograms are computed for 4×4 cells around the interest point location, resulting in a 128 dimensional descriptor vector. The cells and the histograms for a single descriptor are shown in Figure 4. Precomputed gradients are obtained according to the scale of the interest point in order to achieve *scale* invariance. Gradient orientations are rotated according to the interest point orientation in order to achieve *rotation* invariance. Gradient magnitudes are weighted by a Gaussian according to their distance to the interest point locations in analogy to the orientation assignment in the detector stage. Afterwards, the weighted magnitudes are accumulated in the corresponding histogram bins. An interpolation scheme avoids quantization artifacts. Invariance to *linear contrast* changes is achieved by normalizing the descriptor vector to unit length. By clipping the normalized descriptor values at 0.2 and renormalizing to unit length, the descriptor becomes invariant to

non-linear contrast changes to some extent. Clipping high values emphasizes the histogram distributions. Large histogram values would dominate the descriptor otherwise. Furthermore, the descriptor is invariant to changes in *illumination* since gradients are image intensity differences and, therefore, independent of absolute intensity values.

An approach to template-based object recognition with SIFT features and the *generalized Hough transform* [Bal81] has been presented in [Low04]. The basic ideas for this method are important in the *coarse analysis stage* for word spotting with BoF-HMMs.

The generalized Hough transform matches local features with respect to a *reference point*. The reference point is specific to an object that should be detected. For this purpose, the parametric relations between the local features from the template and the reference point in the template are computed. For each local feature from a test image, the most similar local feature from the template image is obtained. For each pair of matching features, the same parametric transformation that relates a local feature from the template with the reference point is applied to the *parameters* of the corresponding local feature in the test image. Finally, each feature in the test image votes for the parameters that have been obtained through the corresponding transformation. A large number of features that is consistent with an object hypothesis results in a large number of votes for the *global* parameter values of the hypothesis. The similarity of the matching features can be considered in the voting process, cf. [Bal81].

If the generalized Hough transform is applied to SIFT features, the parameter-space is 4-dimensional, i.e., two dimensions for the image location, scale and orientation. SIFT features are matched according to the Euclidean distances of their descriptor vectors. In order to better understand the parameter transformation, the SIFT feature parameters can be considered as a vector starting at the feature *location* that is *scaled* and *oriented* according to the parameter values. The parametric transformation rotates and stretches the vector such that it points to the reference point. A larger number of matching features in the test image that correspond to the features in the template will vote for *similar* parameter values, i.e., the hypothesized *pose* of the object. The parameter voting space is quantized in order to avoid ambiguities.

Apart from the successful application in various computer vision tasks, the SIFT descriptor has properties that are making it suitable for the application in historic document images. The orientation histograms abstract from the shape of the script such that the descriptor is invariant to small writing style variabilities. Invariance to contrast changes is helpful in order to cope with typical artifacts in historic documents, like fading ink.

2.3 BAG-OF-FEATURES

The *bag-of-features* (BoF) is a powerful feature vector representation for image regions, cf. [OD11]. It is based on a statistic of local-feature frequencies. The BoF representation has been proposed for image retrieval [SZ03]. The key idea was to apply automatic text retrieval concepts to images. A basic information retrieval concept is the vector space model [BR11, Sec. 3.2.6]. Items from a database as well as the query are represented as points in the same vector space for this purpose. Retrieval can be accomplished by computing similarities between the query and all database items and ranking the items accordingly. Apart from a suitable similarity measure, the numerical vector representation is very important.

In order to represent textual documents of variable length in the same vector space, the *bag-of-words* (BoW) is a standard approach. The vector space can be defined by the most frequent words, so-called *terms*, such that each term corresponds to a dimension in the vector space. Usually this excludes stop words, like *the*, *at* or *on*, which provide no semantic information. Further, the terms are reduced to their word stems in order to achieve invariance with respect to grammatical variants. A textual document can then be represented as a point in the vector space by counting the number of term occurrences, i.e., the representation is a histogram. It should be noted that many vector components are usually zero depending on the number of terms and the length of a textual document.

BoW can be generalized to BoF in order to represent images in the same way. Therefore, a counterpart to a word in a textual document has to be defined for an image. Since a word can be seen as a local text feature, local image descriptors have similar properties which makes them suitable for this purpose. For obtaining local image features, principally any region detector can be used, like SIFT or MSER, cf. [OD11]. However, better results can be achieved if local features are extracted in a regular grid, cf. e.g., [CLV+11]. This is due to the heuristic assumptions in the detection processes. For feature description, the SIFT descriptor has been most successful [OD11] and is considered as a standard descriptor for BoF applications [CLV+11]. Figure 5 shows an application of BoF to a document image, including a dense grid of SIFT descriptors.

After the image descriptor has been established as a counterpart of a word in a textual document, *typical* local image descriptors have to be obtained in order to define the vector space. Computing these features that are typical for the problem domain is an *unsupervised learning* task. For this purpose, a model is required which usually consists of *model parameters* and *meta parameters*. The model parameters are analytically inferred from sample data. The meta parameters are heuristically optimized with respect to a performance measure

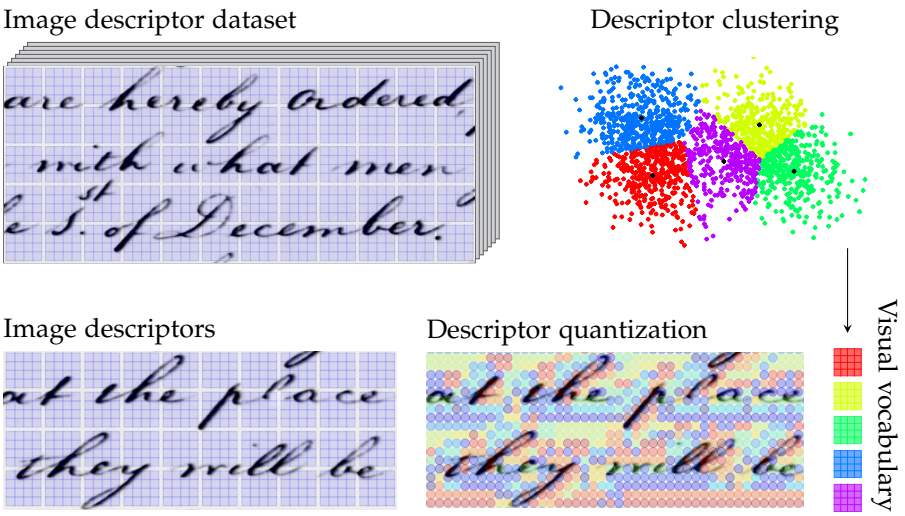


Figure 5: BoF representations. The figure visualizes an application to document images. The *visual vocabulary* is obtained by *clustering* local image *descriptors* from an image dataset. The descriptors are extracted in a dense grid such that they cover the image uniformly. They are indicated in blue and consist of 4×4 cells, like SIFT. In the figure, descriptors do not overlap for a better visualization. The visual vocabulary consists of visual words which correspond to the cluster centroids. Five visual words are indicated in different colors. In order to represent a document image region, each descriptor is quantized with respect to the visual words from the visual vocabulary. The quantization result is exemplarily shown with differently colored descriptor center points. The color indicates the visual word. It can be noted how visually similar pen-strokes are covered with visually similar color patterns. Region representations are given by histograms of visual word frequencies.

that depends on the final application. In an unsupervised learning scenario, the model parameters are estimated *without annotations*. The model estimation process is generally referred to as *training*.

A standard method for computing typical image descriptors is Lloyd's algorithm, cf. [GG92, Sec. 11.3], using Euclidean distances. Given a dataset of sample descriptors, the algorithm randomly initializes a codebook of typical descriptors. Generally, the elements of a codebook are called codewords. In the context of Lloyd's algorithm, the codewords are more specifically referred to as centroids. The *number* of centroids defines the dimensionality of the vector space and is a meta parameter. The codebook contains the model parameters. All descriptors are quantized with respect to the centroids according to smallest Euclidean distance. Afterwards, the centroids are updated such that they minimize the average distance to all descriptors that have been assigned to them during quantization. For Euclidean distances, this is achieved by computing the mean of the corresponding descriptor vectors. The process of quantization and codebook update is iterated until the average quantization error converges. The average

quantization error is the average distance between descriptors and corresponding centroids over the entire dataset. Since the centroids are the prototypical image descriptors they are referred to as *visual words* and the codebook is referred to as *visual vocabulary*. Figure 5 shows an exemplary clustering result for a two-dimensional dataset.

In order to obtain a BoF image representation, descriptors are extracted and quantized with respect to the visual words. In analogy to BoW, a histogram of visual word frequencies is computed. Therefore, the *bag*-of-features is an orderless representation which abstracts from the spatial occurrence of the visual words. Figure 5 shows the quantization result for a section of a document image.

An important difference between BoW and BoF is the expressiveness of the features, i.e., words and visual words. While the occurrence of particular words is usually sufficient in order to categorize a textual document, the appearance of visual words across categories can be expected to be much more diverse. For this reason, the spatial location of visual words is of high importance, cf. [CLV+11]. In contrast, the particular order of words in a textual document is not modeled in the standard representations, cf. [BR11, Chap. 3].

A successful approach for incorporating spatial information in BoF representations is the *spatial pyramid* [LSP06]. In addition to computing the visual word histogram over the entire image region only, the region is subdivided in a pyramidal fashion following a quad tree structure. Such a representation contains L levels with 2^{2l} cells per level where $l \in \{0, 1, \ldots, L-1\}$. Afterwards, visual word histograms are obtained for all cells before they are concatenated. This results in a $D = \sum_{l=0}^{L-1} 2^{2l} V$ dimensional spatial pyramid vector where V is the size of visual vocabulary. Different weighting schemes have been proposed in order to normalize the numbers of visual words per cell that is decreasing if the pyramid level increases, e.g., cell representations are normalized with the 1-norm [CLV+11].

For word spotting, the spatial pyramid plays an important role in order to represent word images. It can be adapted for modeling the sequential structure of text. For this purpose, the quadtree structure is replaced with a temporal structure [RAT+11; SF17]. At the same time the specificity of the representation can be controlled through the number of cells. This allows for coping with word size variabilities.

2.4 MULTINOMIAL MIXTURE MODEL

The multinomial mixture model can represent a statistical distribution of BoF vectors. Due to this reason, it can be seen as a foundation for modeling BoF vectors in the statistical HMM process. After a brief introduction to statistical modeling with multinomial distributions, the multinomial mixture model will be presented.

Modeling BoW representations with multinomial distributions is a standard approach for text classification [BR11, Sec. 8.4.4]. For each text category, a model is estimated such that it maximizes the probability for *generating* the samples of the corresponding class. An unknown sample can be classified according to the model that generates the sample with maximum *posterior* probability. The concept is called *Bayesian inference* and follows from *Bayes' theorem*, cf. [Bis06, p. 22] and [DHS00, Sec. 2.1].

For this purpose, the samples are referred to as *observations*. The model that generated the sample cannot be observed but has to be inferred. The generation of an observation is represented on the right-hand side of Equation 1.

$$posterior = \frac{prior \times likelihood}{evidence} \tag{1}$$

First, the class is selected according to a distribution of *prior* probabilities. Then, the sample is generated according to a distribution which is conditioned on the class model, i.e., the *likelihood*. According to Bayes' theorem, the product of the prior and the likelihood is proportional to the *posterior* probability for the *class model* conditioned on the sample. The *evidence* represents the sample distribution and is, therefore, independent of the class. It is a normalization factor which is required for computing a posterior *probability* but can be neglected for obtaining the class with *maximum* posterior probability.

Generally, the posterior denotes a probability *after* making the observation, i.e., after the sample has been generated. The prior denotes a probability which is independent of any observation, i.e., *before* the sample is generated. If the classes are represented by *discrete* multinomial distributions, the likelihood is a conditional *probability* for the sample. It is conditioned on a class model. The concept is not specific to multinomial distributions but applies to any model which is suitable for representing the data, e.g., *continuous* Gaussian distributions.

The main idea for modeling the generation of BoW vectors with multinomial distributions is to model the generation of each BoW vector component individually. This way, the model resembles the BoW vector computation. In analogy, multinomial distributions can be used for modeling BoF, e.g., for image classification [CDF+04].

In order to model the generation of a BoF vector $\mathbf{x} \in \mathbb{N}_{\geqslant 0}^{V}$, $N = \|\mathbf{x}\|_1$ visual words are independently drawn according to the visual word probabilities $\mathbf{p} = (p(\mathcal{V} = 0), \ldots, p(\mathcal{V} = V-1))^\top$ where V is the size of the visual vocabulary. For this purpose, $\mathcal{V}$ is a discrete random variable over the event space $\Omega_\mathcal{V} = \{0, \ldots, V-1\}$. Thus, the parameter vector $\mathbf{p}$ of a multinomial distribution can be used as a class model. Equation 2 shows the probability mass function of the multinomial

distribution. It consists of the multinomial coefficient and a product over visual word probabilities.

$$p(\mathbf{x}|\mathbf{p}) = \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{x_v} \tag{2}$$

The product models a joint probability that weights visual word probabilities $p(\mathcal{V} = v)$ according to the absolute visual word frequency $x_v \in \mathbb{N}_{\geqslant 0}$ in the exponent. Thus, the higher the frequency the more the overall product is influenced by the respective visual word probability. In case of $x_v = 0$, visual word probability $p(\mathcal{V} = v)$ will not be considered in the product.

The generation of BoF vector $\mathbf{x}$ can be understood with an urn scheme. In this scheme an urn is filled with balls of V different colors where each color corresponds to a visual word. The proportion of differently colored balls is defined by $p(\mathcal{V} = v)$ for all $v \in \Omega_v$. A single visual word is generated by drawing a ball from the urn, noting its color and returning the ball back into the urn. The process is repeated N times. According to the urn scheme, the product in Equation 2 is normalized by the multinomial coefficient. It specifies the number of ways to draw N visual words such that each visual word $v \in \Omega_v$ is drawn x_v times.

This model is sufficient as long as a single class is typically represented by BoF vectors with similar visual word frequencies. Otherwise, the model will generalize too much and not be specific to the samples of the class anymore. A solution to this problem are *mixture models*, cf. [Bis06, Chap. 9]. For this purpose, a mixture of multinomial distributions $\Theta = \{(c_k, \mathbf{p}_k) \,|\, 0 \leqslant k < M\}$ is defined by M mixture components with visual word probabilities $\mathbf{p}_k$ and mixture weights $c_k \geqslant 0$ with $\sum_{k=0}^{M-1} c_k = 1$. The mixture weights define the relative proportions of components in the mixture. Formally, the weights are prior component probabilities $c_k = p(\mathcal{M} = k)$, i.e., probabilities for the components which are independent of the sample. For this purpose, the probability for a mixture component is represented by a discrete random variable $\mathcal{M}$ over the event space $\Omega_{\mathcal{M}} = \{0, \ldots, M-1\}$. The probability mass function for the multinomial mixture model is defined in Equation 3.

$$p(\mathbf{x} \,|\, \Theta) = \sum_{k=0}^{M-1} c_k \, p(\mathbf{x} \,|\, \mathbf{p}_k) \tag{3}$$

The number of mixture components M is a meta parameter. The higher the number of components, the more specific are the visual word configurations that are represented by the individual components. Figure 6 visualizes a multinomial mixture model with two components in a visual word simplex with three visual words. Component $k = 0$ represents BoF vectors where mostly the blue visual

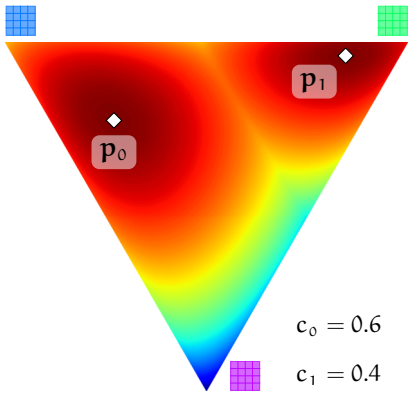


Figure 6: Simplex visualization of the multinomial mixture model. BoF can be visualized in a simplex according to relative visual word frequencies. The simplex vertices are labeled with visual words as indicated by the blue, green and pink descriptors. A point in the simplex refers to a BoF representation. The relative distance of the point to each visual-word vertex is anti-proportional to the relative frequency of the corresponding visual word in the BoF. For example, the simplex center point refers to the BoF with uniform visual word frequencies. The probability mass in the simplex, i.e., probability for the corresponding BoF vectors, is modeled by the multinomial mixture distribution as indicated with blue to red colors in the logarithmic domain. The colors have been scaled to the corresponding minimum and maximum probabilities. Diamond markers indicate the mixture component parameters $\mathbf{p}_k$, i.e., visual word probabilities for each component $k \in \{0, 1\}$. Mixture component weights c_k are specified next to the simplex.

word followed by the pink visual word have high frequencies. Component $k = 1$ represents BoF vectors where the green visual word has high frequencies and the other visual words have low frequencies.

Given a dataset of annotated samples, the training of a multinomial mixture model can be performed according to the *maximum likelihood* criterion with the *expectation maximization* (EM) algorithm, cf. [Biso6, Sec. 9.3]. The main idea for this estimation is to optimize the *incomplete data log-likelihood function*. It expresses the probability for generating the samples of a class given the model in the logarithmic domain. However, the maximum likelihood optimization, which is based on differentiating the likelihood function with respect to the model parameters, cannot be performed directly. This is due to the mixture model in which a sample can be generated by *any* mixture component according to the posterior probability for the component given the sample. Initially, these posteriors are *unknown*. This is the reason for referring to the likelihood function as *incomplete*. By treating the posteriors as *hidden* variables in the optimization process, the *complete* data log-likelihood can be obtained. It contains estimates for the component posteriors that are computed from a given model. The process of computing these estimates is referred to as the *expectation* step and

is directly based on Bayes' theorem. Equation 4 defines the posterior for mixture component k conditioned on sample $\mathbf{x}$.

$$p(\mathcal{M} = k \mid \mathbf{x}, \Theta) = \frac{c_k \, p(\mathbf{x} \mid \mathbf{p}_k)}{\sum_{l=0}^{M-1} c_l \, p(\mathbf{x} \mid \mathbf{p}_l)} \tag{4}$$

The denominator corresponds the sample distribution, i.e., the evidence, cf. Equation 1, and is represented by the mixture model as defined in Equation 3.

Afterwards, the maximum likelihood optimization of the model parameters can be performed. This is referred to as the *maximization* step. For each component, the visual word probabilities are estimated as weighted relative visual frequencies where the weights for all samples are computed in the expectation step according to Equation 4. It is important to note that the visual word probabilities must not be zero, cf. Equation 2. This can be ensured by smoothing the visual word probability distributions with a uniform distribution [MN98, Equ. 6], i.e., so-called *Laplace smoothing*. A detailed discussion of multinomial mixture models for BoW representations can be found in [NMT+00]. The derivation of the maximum likelihood parameter estimation for a multinomial distribution is presented in [Bis06, Sec. 2.2].

The expectation and maximization steps are iterated until the complete data log-likelihood converges. Most importantly, it can be shown that optimizing the complete data log-likelihood also optimizes the incomplete data log-likelihood, cf. [Bis06, Sec. 9.4]. The mixture model for the first iteration is initialized randomly. It should be noted that the optimization is performed in the logarithmic domain in order to simplify the derivations and for numeric stability. The logarithm of the objective function does not change the optimization result since the logarithm function grows strictly monotonically.

For modeling BoF image representations, an assumption of the multinomial model is that BoF vectors $\mathbf{x}$ are *not* sparse. This is due to the product of probabilities $p(\mathcal{V} = v)$ in Equation 2 which implies that $p(\mathcal{V} = v) > 0$ for all $v \in \Omega_v$. However, in the context of word spotting, BoF representations are typically *very* sparse due to the use of large visual vocabularies and small document image regions that contain comparably few visual words. In addition to the sparsity, the use of Equation 2 in a mixture model makes the violation of the condition even more severe. This is because multinomial mixture components specialize on modeling only certain visual word configurations.

For coping with these challenges, different heuristics have been evaluated in the context of text classification [RST+03]. However, a more systematic approach is proposed in [MKE05] by using Dirichlet compound multinomial distributions. For this reason, the multinomial mixture model can be seen as an important foundation for the mixture models that will be discussed in Section 4.4. The EM algorithm is conceptually the same for all of these models and differs

mostly in the maximization step. The multinomial mixture model will be used as a reference in the evaluation of probabilistic BoF models for the HMM integration in Section 5.3.4. The corresponding parameter evaluation can be found in Section A.1.

2.5 HIDDEN MARKOV MODELS

Hidden Markov models (HMMs) are statistical sequence models that have been used in various recognition tasks where the representation of sequential data is required, cf. [Fin14, Chap. 2]. In particular, this includes handwriting recognition where sequences of feature vectors are extracted from line images or word images within so-called frames. Recognition is based on a set of *elementary modeling units*. In the context of handwriting recognition, these elementary units usually correspond to characters. Thus, the elementary modeling units are referred to as character HMMs in this case.

Character models can be concatenated in order to obtain a word model. Models that consist of other models, i.e., models that are not elementary, are referred to as *compound* HMMs, cf. [Fin14, Sec. 8.3]. The units can be chosen according to the needs of the application, i.e., character-level units or word-level units, cf. [PF09].

HMMs represent the sequences of feature vectors that are *typical* for the corresponding units. Provided that the features represent the visual appearance of the pen-stroke in a frame, a character HMM represents which features are typical for consecutive sections, like the beginning, middle and end of the corresponding character. In order to do so, HMMs consist of states and each state is associated with a statistical distribution over the feature vector space, cf. [Fin14, Sec. 5.1] and Figure 7. For discrete feature spaces, the probability mass function has high probability for feature vectors that are typical in the corresponding state. An example for a probability mass function that represents a distribution of BoF vectors has been shown in Figure 6. The model that is used in order to represent these distributions is referred to as the *output model* [Fin14, Sec. 5.1]. The choice of a suitable output model for representing BoF vectors is an important aspect in the presentation and evaluation of BoF-HMMs (Chapter 4 and 5).

In order to represent typical sequences of feature vectors, HMMs use a generative statistical approach, cf. [Fin14, Chap. 5]. In this generative process, one state *generates* a feature vector at each point in time. In analogy to the generative statistical model that has been presented in Section 2.4, the state sequence that generated the *observed* feature vector sequence most likely, must be *inferred*. In order to recognize text in a line image, the observation sequence is obtained by feature extraction. Afterwords, the *hidden* state sequence is inferred that generated the observation sequence most likely. Since the states are associated with the elementary modeling units, e.g., character HMMs, this

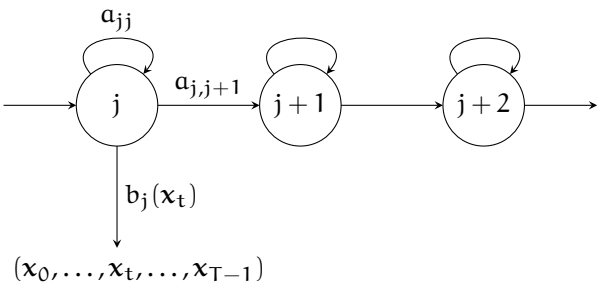


Figure 7: Hidden Markov model. The figure shows three states of an HMM. Transition probabilities and output probabilities are indicated for state j. State transitions are organized in a linear topology allowing for self-transitions and transitions to the next state. For state j, these are denoted as a_{jj} and $a_{j,j+1}$. The output probability $b_j(x_t)$ denotes the probability for generating feature vector x_t in state j. Feature vector x_t is part of the *observed* feature vector sequence $(x_0, \ldots, x_{T-1})$. Probabilistic inference is the task of computing the formerly *hidden* state sequence that generated the observation sequence most likely.

corresponds to recognition. Furthermore, the spatial location of each observation, i.e., each feature vector, in the text line is know. Thus, along with the recognition, HMMs implicitly infer a *segmentation*.

It can be noted, that recognition on character level is much less constrained than recognition on word level. Word HMMs can only be defined with a lexicon. Thus, the number of possible character sequences is reduced substantially. Further context information can be incorporated with a *language model* (LM). A word-level language model represents word sequence probabilities. Word sequences of length n are called *n*-grams. HMMs allow for a direct integration of a bigram LM by incorporating the word sequence probabilities at transitions between words, cf. [Fin14, Chap. 12]. This combination of statistical character models and statistical word models illustrates, how HMMs support the integration of context information at various levels.

The statistical HMM process is modeled with a generative finite state machine, see Figure 7. The finite state machine represents a discrete stochastic process which models the generation of a feature vector sequence in two stages. For generating a feature vector at time $t \in \{0, \ldots, T-1\}$, the *active* state, i.e., the generating state, is determined in the first stage. Probabilistically this is modeled by a random variable S_t over the event space $\Omega_s = \{0, \ldots, S-1\}$ where S is the number of states in the HMM. The behavior of the process is controlled by discrete distributions of start probabilities and transition probabilities. The start probabilities are denoted as vector $\boldsymbol{\pi} = (\pi_0, \ldots, \pi_{S-1})^\top$ with $\pi_j = p(S_0 = j)$ for all $j \in \Omega_s$. The transition probabilities $a_{ij} = p(S_t = j \,|\, S_{t-1} = i)$ are organized in matrix $[a_{ij}]$ with $(i, j) \in \Omega_s \times \Omega_s$. Furthermore, the probability for a state at time t *only* depends on the preceding state at time $t-1$. Generally, any limitation of the temporal

context is known as *Markov property*, cf. [Fin14, p. 72]. If the temporal context is limited to a *single* state, the stochastic process is referred to as a first-order Markov process. The limitation is essential for efficient inference and model estimation. Figure 7 shows three states of an HMM. Transition probabilities are indicated for state j. State j will be assumed to be the active state in the following.

After state j is selected in the first stage, the generation of feature vector $\mathbf{x}_t$ is modeled in the second stage. For this purpose, a probability distribution over the feature space is represented by $b_j(\mathbf{x}_t) = p(\mathbf{x}_t | s_t = j)$. This output probability distribution depends only on the active state and is independent of any preceding states or preceding feature vectors. This is known as *output independence assumption*, cf. [Fin14, p. 72]. The output model has to be chosen according to the characteristics of the feature space. If the output model uses continuous probability density functions[1], the HMMs are referred to as *continuous* HMMs. Figure 7 shows a sequence of feature vectors and indicates the generation of feature vector $\mathbf{x}_t$ in state j.

A popular choice for the output model is the multivariate *Gaussian mixture model* (GMM) [Fin14, Sec. 5.2]. Multivariate Gaussian distributions are continuous probability density functions that allow for representing the feature space effectively. This is due to a requirement of the features that should be discriminative for different shapes of the pen-stroke. Thus for a shape, the Gaussian distribution models the typical feature vector and the typical variation with respect to this feature vector in the feature space [PF09]. However, a single Gaussian per state is insufficient, since a single class, e.g., a character, will typically be represented by different pen-stroke shapes. Mixture models allow for modeling the high intra-class variability of handwritten script, cf. Section 2.4.

While a GMM per state allows for a very accurate representation of the data, the number of model parameters grows considerably with every state, cf. [Fin14, Sec. 9.2]. For example, this is influenced by the dimensionality of the feature space and the number of mixture components. Since HMMs correspond to classes and have to be estimated from *annotated training data*, this can be a problem. The number of model parameters and the number of training samples that are required for estimating these parameters are directly related, cf. [Fin14, p. 153]. A common possibility for reducing the number of model parameters is mixture tying, cf. [Fin14, Sec. 9.2.3]. For this purpose, only the mixture weights are specific to the states and the mixture components are *shared* by all states. HMMs with shared mixture components are referred to as *semi-continuous* (SC) HMMs. Equation 5 shows that the mixture weights c_{jk} are state-dependent. This is indicated by index j. The mixture component distribution is conditioned on mixture

1 The distinction between probabilities and densities will *only* be made explicit if this is important in the context of the presented aspect.

component index k. For this purpose, the probability for a mixture component at time t is represented by the random variable $\mathcal{M}_t$ over the event space $\Omega_{\mathcal{M}}$. The mixture component distributions are independent of state j. The HMM is denoted by λ and is fully defined by the number of states and their indices, start probabilities, transition probabilities and the parameters for the output probability distributions.

$$b_j(\boldsymbol{x}_t) = \sum_{k=0}^{M-1} c_{jk}\, p(\boldsymbol{x}_t \mid \mathcal{M}_t = k, \lambda) \tag{5}$$

The key idea for recognition with HMMs is to decode the *hidden* state sequence that generated the *observed* feature vector sequence most likely. Since the states are associated with the HMMs, a sequence of semantic units is obtained. This corresponds to a *joint* computation of recognition *and* segmentation.

Given a feature vector sequence $\mathbf{O} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{T-1})$ and HMM λ, the probability for a specific state sequence $\boldsymbol{S} = (s_0, \ldots, s_{T-1})$ with $s_t \in \Omega_s$ is defined in Equation 6.

$$p(\boldsymbol{S} \mid \mathbf{O}, \lambda) = \frac{p(\mathbf{O}, \boldsymbol{S} \mid \lambda)}{p(\mathbf{O} \mid \lambda)} \tag{6}$$

The joint probability $p(\mathbf{O}, \boldsymbol{S} \mid \lambda)$ can simply be computed by multiplying the start probability, transition probabilities and output probabilities along the state sequence. However, the naive computation of the total output probability $p(\mathbf{O} \mid \lambda)$, according to $p(\mathbf{O} \mid \lambda) = \sum_{\boldsymbol{S}} p(\mathbf{O}, \boldsymbol{S} \mid \lambda)$, requires to marginalize over all possible state sequences, cf. [Fin14, Sec. 5.5.1]. This is computationally infeasible because the exhaustive search does not take advantage of the model assumptions.

An efficient approach for computing $p(\mathbf{O} \mid \lambda)$ exploits the independence assumptions of the HMM. The *forward algorithm*, cf. [Fin14, Sec. 5.5.2], recursively computes the probability for generating the feature vector sequence until time t along *any* state sequence that reaches state i at time t. By storing the partial results in the *forward variables* $\alpha_t(i)$, the independence assumptions allow for reusing the partial results from the recursion step at time t in the next recursion step at time $t+1$. For the probability of generating the remaining feature vectors until the end of the feature vector sequence along *any* state sequence that starts in state j at time t, *backward variables* $\beta_t(j)$ can be obtained in analogy. The *total* output probability represents the probability of generating the feature vector sequence along *any* state sequence in the HMM and can be computed either with the forward algorithm or with the backward algorithm. For this purpose, the forward variables obtained in the last recursion step are accumulated over all states. The computation with the backward algorithm works in analogy, cf. [Fin14, Fig. 5.6]. Apart from the possibility to efficiently compute the total output probability, the forward algorithm and the backward algorithm are the foundation for efficient model estimation.

In contrast to computing the *total* probability for generating a feature vector sequence along *any* state sequence, the *Viterbi algorithm* computes the *optimal* probability for generating a feature vector sequence along the *optimal* state sequence, cf. [Fin14, Sec. 5.6.1]. Provided that $\mathbf{S}^*$ is the state sequence that generates features vector sequence $\mathbf{O}$ with maximum probability, the Viterbi algorithm computes $\mathbf{S}^*$ as shown in Equation 7 and 8.

$$\mathbf{S}^* = \arg\max_{\mathbf{S}} p(\mathbf{S} \mid \mathbf{O}, \lambda) \tag{7}$$

$$= \arg\max_{\mathbf{S}} p(\mathbf{O}, \mathbf{S} \mid \lambda) \tag{8}$$

For this purpose, the Viterbi algorithm maximizes $p(\mathbf{O}, \mathbf{S} \mid \lambda)$ over all state sequences, see Equation 8. This is sufficient, since the denominator in Equation 6 is independent of $\mathbf{S}$. Therefore, $\mathbf{S}^*$ is the state sequence with maximum posterior probability.

With respect to model estimation, the *Baum-Welch algorithm* is an EM algorithm that is widely applied for this purpose, [Fin14, Sec. 5.7.4]. The forward and backward variables play a crucial role for computing posterior probabilities in the expectation step. The maximum likelihood criterion is based on the total output probability $p(\mathbf{O} \mid \lambda)$. For optimizing the model parameters, the associations between the training samples and the states as well as the mixture components are missing. In the semi-continuous scenario, the associations are represented by state posteriors $p(\mathcal{S}_t = i \mid \mathbf{O}, \lambda)$, transition posteriors $p(\mathcal{S}_t = i, \mathcal{S}_{t+1} = j \mid \mathbf{O}, \lambda)$, state-dependent mixture component posteriors $p(\mathcal{S}_t = i, \mathcal{M}_t = k \mid \mathbf{O}, \lambda)$ and state-independent mixture component posteriors $p(\mathcal{M}_t = k \mid \mathbf{O}, \lambda)$.

Given these posteriors from the expectation step, updates for state-dependent parameters are obtained in the maximization step. This includes start probabilities [Fin14, Equ. 5.18], transition probabilities [Fin14, Equ. 5.17] and mixture weights [Fin14, Equ. 5.21]. In the semi-continuous scenario the mixture components are state-independent and updated according to [Fin14, Equ. 5.25 and 5.26]. With respect to the updates for the transition probabilities it should be noted that the non-zero probabilities are defined by the model topology. The topology is a meta parameter. For handwriting recognition, the *linear* and *Bakis* topologies are most common. The linear topology only allows self transitions and transitions to the next state, see Figure 7. The Bakis topology extends the linear topology with skip transitions. Other important meta parameters are the number of states in an HMM and the number of shared mixture components.

The iterative procedure of expectation and maximization steps requires an initial model. For this purpose, the *flat start* is a common approach. In the semi-continuous scenario, the initial GMM can be obtained from the training data in an unsupervised manner with the corresponding EM algorithm, cf. [Fin14, Sec. 4.4.2]. The state-depen-

dent parameters are initialized uniformly. The Baum-Welch algorithm terminates after a defined number of iterations or when the relative change of the total output probability falls below a threshold.

Word spotting with HMMs is essentially based on Equation 6. Given a feature vector sequence that has been extracted from a line image, the idea is to compute a posterior probability for any state sequence that contains the query word. For this purpose, a query model is required that represents the occurrence of the query in a text line. In this regard, compare the numerator of Equation 6 which represents the output probability along a single state sequence. The posterior is obtained after normalizing with the total output probability. Compare the denominator of Equation 6. Since the total output probability can be expressed as marginalization over all possible state sequences, this can be seen as the output probability with respect to *any possible transcription* of the text line. The posterior probability is high if the numerator and the denominator are similar. This is the case if the total probability in the denominator is dominated by the same state sequences that are represented by the numerator, i.e., state sequences that contain the query word. Therefore, the denominator is essentially a recognition model, the so-called *filler model*, and the word spotting performance is bounded by the recognition performance.

If annotated training material is available for estimating the filler model, HMMs allow for computing the similarity score for the query and the corresponding segmentation of the text line jointly. For word spotting this is a powerful property that sets them apart from methods that are based on holistic representations of *potential* word segments. Relevant words that are not represented by these segments will not be part of the retrieval list.

If *no* annotated training dataset from the problem domain is available, SC-HMMs have been used for segmentation-based query-by-example word spotting [RP09b] as well as segmentation-based query-by-string word spotting [RP12b]. The shared mixture model in SC-HMMs can be estimated in an unsupervised manner. The remaining model parameters can be estimated independently and from very few annotated samples [PF05]. Only the query word image has been used for this purpose in [RP09b]. Synthetically generated annotated samples have been used in [RP12b]. In both cases, the filler model is approximated by the distribution of the feature vectors. This distribution is represented by the shared mixture model.

3 WORD SPOTTING

Word spotting provides the possibility to search document images automatically. Locating words is one of the most important functionalities for working with larger document collections. This is especially interesting in cases in which it is hard to create an accurate transcription automatically. Therefore, word spotting has become popular for supporting the analysis of handwritten and historic documents, cf. Chapter 1.

For this purpose, the user provides a query and the word spotting system retrieves document image regions according to similarity to the query without transcribing the documents first. Consequently, any word spotting system consists of methods for:

- extracting relevant document image regions (Section 3.1),
- representing document image regions numerically (Section 3.2),
- computing similarities with respect to the query based on numerical representations (Section 3.3).

If region extraction is closely integrated with the document image representation and retrieval, all three aspects constitute a word spotting method. Otherwise, a word spotting method consists of a numeric feature representation and retrieval.

In the following, word spotting methods will be presented in this context. The presentation is concluded with a discussion (Section 3.4) which compares methods with respect to their applicability in practice. Methodological aspects for segmentation-free retrieval as well as HMMs are addressed in detail. This allows for distinguishing the properties of the methods and setting the methods apart from BoF-HMMs that are proposed for word spotting in Chapter 4. A comprehensive list of word spotting methods including a categorization with respect to word spotting scenarios and methodology can be found in Appendix C.

3.1 DOCUMENT IMAGE REGIONS

For spotting the query word in a document image, plausible document image regions must be obtained. These regions are processed during retrieval and are the basis for creating a ranked retrieval list.

Word spotting methods can be grouped in the categories *word-level*, *line-level* and *document-level* according to the regions required by the method. Figure 8 shows examples for regions given as word and line

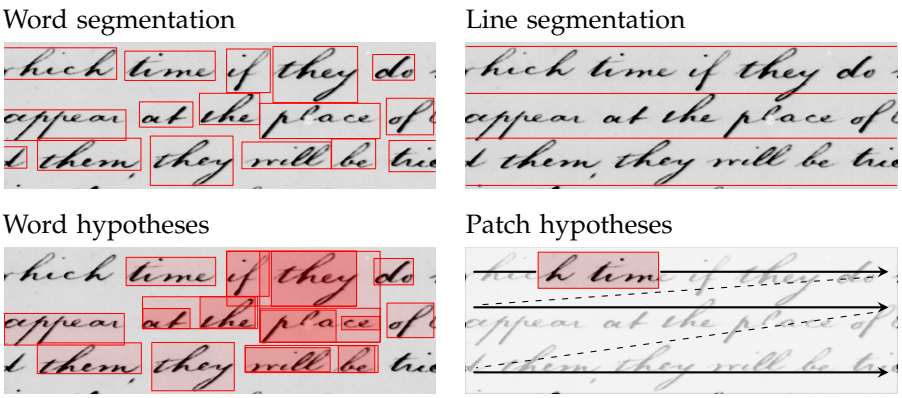


Figure 8: Document image regions. The top row shows document regions that have been obtained by heuristic segmentation on word-level and on line-level. Word spotting systems rank these regions according to similarity to the query. In contrast, the bottom row exemplarily indicates region hypotheses that do not represent final segmentations. They are shown semi-transparently for this reason. The word spotting system *selects* and ranks hypotheses according to relevancy with the query. Word-level hypotheses are extracted from the document image while patch-based hypotheses are uniformly sized and densely sampled over the document image.

segments as well as word- and patch-based hypotheses for word spotting on document level. The examples illustrate that *hypotheses* provide competing alternatives for potentially relevant document image regions. The segmentation of the document image in relevant and non-relevant regions is performed during retrieval while taking the query into account. Therefore, it is *not* important to identify *only* correct word image regions, but it is important that the word image regions that are relevant to the query are among the proposed hypotheses. Under the assumption that relevant regions will be more similar to the query than irrelevant regions, the large majority of irrelevant regions can be filtered out with *non-maximum suppression* (NMS). Among overlapping regions, the filter only keeps the region with highest similarity, for this purpose [NG06].

In contrast to document region hypotheses, word or line segments are given to the word spotting method beforehand. During retrieval this reduces the search space considerably. However, a method for reliably extracting the segments is required.

Properties and requirements for word and line segmentation are discussed in Section 3.1.1. Afterwards, different strategies for generating document-level region hypotheses are presented in Section 3.1.2 and Section 3.1.3.

3.1.1 *Word and line segments*

Word spotting methods that operate on *word-level* require a given segmentation of the document image into words. Since this segmentation is considered as correct and all further operations are based upon it, these methods are referred to as segmentation-based. The main limitation is the requirement to provide the segmentation prior to retrieval. Errors in the segmentation will directly lead to errors in the retrieval result. These methods are, therefore, primarily successful in word spotting scenarios where the segmentation is trivial, e.g., due to regular character spacing, cf. e.g., [KH93; Hul94]. In order to be applied in more difficult scenarios, like historic and handwritten documents, extensive manual parameter tuning is required, cf. e.g., [MHR96]. Word segmentation methods that are particularly sensitive to the chosen parameters are based on projection profiles or *connected components* (CCs), cf. [Kis14]. More advanced methods are based on scale space representations [MR05] and CNNs [WB15]. The latter method is interesting because it aims at learning the visual appearance of words without using a model for distinguishing different word classes. While this avoids the problem of explicit word recognition, the method still relies on the presence of discriminative characteristics in the document collection, like typical word and line spacings. In historic and handwritten documents this can be problematic due to the highly irregular visual appearance of text. Word spotting on word-level usually works by matching the numeric feature representations of the word segments against the numeric feature representation of the query. Word segments are ranked accordingly and presented to the user in a retrieval list, cf. e.g., [RM07].

Word spotting methods that operate on *line-level* only require a given segmentation of the document image into lines. Since the challenges in line segmentation are a subset of the challenges in word segmentation, line segmentation can be considered as simpler. The main challenges lie in handling skewed and touching text lines. Touching text lines pose a severe problem since the exact text line boundaries are very hard to define in this case. As for word segmentation, most line segmentation methods are based on projection profiles and CCs. In order to spot words in a text line, the line is represented as a sequence of feature vectors. The query is searched within this sequence, cf. Section 3.3.2. Due to this treatment, many different segmentations for locating the query word are considered and evaluated against each other. An explicit segmentation of the text line into words is avoided. Line-based word spotting methods are segmentation-free on line-level for these reasons, cf. e.g., [KAA+00; FKF+12; FFM+12]. Word spotting on line-level often works by ranking entire text lines according to relevance to the query [FKF+12]. In addition, it is also possible to present the estimated query word locations to the user.

3.1.2 *Patch hypotheses*

Patch hypotheses are mostly dependent on the query word. The patch geometry, e.g., width and height, is derived from the geometry of the query bounding box, cf. e.g., [GP09; ZT13; AGF+14b; KKG15; GV15a; RAT+15a]. The patch positions are either uniformly distributed in the entire document image, cf. [AGF+14b; RAT+15a], within text areas, cf. [GP09], or located in document image regions that are visually similar to the query, cf. [ZT13; KKG15; GV15a]. Patch-based approaches are typically applied in query-by-example scenarios, cf. [RAT+15a]. If the query is given as string, character size estimates are required, cf. e.g., [GV15a].

The first approach towards segmentation-free word spotting on document level has been presented in [KGG97]. In their two-stage method, potentially relevant document image regions are identified first and processed in more detail afterwards. The first stage consists of computing the normalized cross-correlation of different examples of the query word with the document images. Normalized cross-correlation makes template matching more robust against intensity and contrast variations, cf. [KGG97, Appx. A]. In the second stage, binary word images are heuristically segmented from the document image regions around the top correlation peaks. These word images are finally evaluated against the query word templates. The first stage can be considered as a precursor of patch-based word spotting. The normalized cross-correlation results in a patch-based framework where the patch size is equal to the template size and a query template is matched at every document image position. However, the detected interest points are only used as a starting point for performing word image segmentation.

The first patch-based segmentation-free method has been presented in [GP09]. Patches are matched with an exemplary query word template, yielding a score at every patch position. The patches have the same size as the template. Locally best matching patches are retrieved, i.e., out of competing patches a patch is selected with NMS. The approach is related to the first stage in [KGG97]. An important difference is the retrieval of patches, instead of interest points, according to NMS.

The patch-based framework presented in [GP09] has essentially been used in many segmentation-free query-by-example word spotting methods, cf. e.g., [AGF+14b; RDE+14; RAT+15a; GV15b; RKE16]. With respect to the architecture of a patch-based framework, an important aspect concerns the patch geometry. Since patch-based methods generally assume limited word size variability, accurate detections are easier to achieve if the patches have a similar geometry as the query, e.g., [AGF+14b; GV15b; RKE16]. However, using patch geometries that are adapted to the query comes at the cost of higher

computational efforts at query time. For example, in [RAT+15a] only four different patch sizes are considered that can be entirely precomputed for efficient retrieval. At query time the patch size is chosen that is closest to the size of the query word image. The query-by-example methods presented in [AGF+14b; GV15b] and [RKE16] use patches of the same size as the query. The high computational efforts are addressed by using data structures allowing for computing patch representations efficiently at query time, cf. Section 3.3.

The final category of patch-based methods is inspired from object detection with keypoint matching, cf. [Low04]. The basic idea in the query-by-example scenarios considered in [ZT13; KKG15; HF16; ZPG17] is to match local image descriptors from the query word image with local image descriptors in the document images. Hypotheses for possible query word locations are document image regions that contain a sufficient number of matching keypoints [ZT13; KKG15; HF16] or match with a reference keypoint from the query [ZPG17]. Besides analyzing the local neighborhood of individual, matching keypoints [ZT13; KKG15; ZPG17], sets of candidate keypoints can be selected with densely sampled patches [HF16]. The patch geometry depends on the query size and can be transformed according to matching keypoint configurations [KKG15; HF16].

3.1.3 *Word hypotheses*

Word hypotheses are document image regions that are likely to contain words. Words are detected in the document images independently of the query. Detectors are either defined heuristically, cf. e.g., [LLE07; KWD14; GV17; GV18], or estimated from sample data as presented in [WLB17; RSR+17]. The bounding box geometry, e.g., width and height, of a word hypothesis is either based on the detector result [KWD14] or based on further analysis of the query.

The first attempts towards segmentation-free word spotting based on word hypotheses have been made in [KAA+00]. In the query-by-example scenario, word starting and word ending positions are detected within given text line images. Hypotheses are derived by combining these start–end positions with each other. In order to limit the search space, additional constraints are applied. These depend on heuristics that have to comply with the query template, i.e., width, gap statistics and ink-background transitions. Overlapping hypotheses are suppressed according to similarity to the query with NMS.

In [LBE05; LLE07] this idea is extended to document-level query-by-example word spotting. The approach is based on small *regions-of-interest* (RoIs), also referred to as zones-of-interest, that are detected in the query image and so-called guides that are detected in the document images. Guides in the document images are considered as possible word starting positions. RoIs are matched with document im-

age regions according to their spatial relation with the possible word starting positions. Competing hypotheses are evaluated against each other, cf. [KAA+00], and suppressed with NMS.

Another possibility for generating text hypotheses are CCs [KWD14; RLF15; GV17; GV18; RSR+17; WLB17]. Hypotheses are created either by grouping CCs in a given binary document image [KWD14; RLF15; GV17; GV18] or by computing different binary images in which CCs correspond to parts of words, to words or to combinations of (parts of) words [RSR+17; WLB17]. An important assumption of the methods presented in [KWD14; GV17; GV18] is that words do not touch. Otherwise multiple words will be represented in a single hypothesis and cannot be detected individually.

The approach in [KWD14] aims at being as simplistic as possible. Hypotheses are defined by grouping CCs according to their size and distances to each other. For this purpose, a number of thresholds must be defined. In contrast, the graph-based method in [RLF15] is considerably more complex. Document images are binarized and skeletonized, cf. [Gat14, pp. 75–77]. From the skeletonized CCs, graphs are constructed by splitting the CCs in convex CCs. These are quantized with respect to a grapheme codebook that is estimated in an unsupervised manner. Each resulting grapheme is represented as a graph vertex. Vertices of adjacent graphemes in the document image are connected with edges. With subgraph matching between query and document image representations, different convex CC combinations are considered. This costly procedure is robust with respect to touching words.

The approaches presented in [RSR+17; WLB17] use models for word hypothesis extraction that are estimated from sample data in a supervised manner. In [RSR+17] hypotheses are based on local word detector scores. By using an adaptation of the *maximally stable extremal regions* detector [MCU+04], word hypotheses are given as ERs. Essentially, these ERs correspond to CCs in binary detector score maps that are obtained at different thresholds, cf. Section 4.2.1. Word detector scores are computed with CNNs. Similarly, the document images are binarized at different thresholds in [WLB17]. After morphological pen stroke dilation with different structuring elements, CCs correspond to word hypotheses. Hypotheses are classified into words and non-words with a CNN. The approach has originally been presented for word segmentation [WB15]. In [WLB17] these hypotheses are referred to as dilated text proposals. They are used in order to complement hypotheses generated by a region proposal network [RHG+15]. Both methods [RSR+17; WLB17] are robust with respect to touching words. This is due to their models that are estimated from annotated sample data and their consideration of multiple binarization thresholds in the CC extraction.

3.2 DOCUMENT IMAGE REPRESENTATIONS

For computing similarities between document image regions and the query, numerical feature representations are required. Together with the measure used for computing similarity to the query, these are characteristic for a word spotting method. In the following, document image representations will be presented in this context. The focus will be on their prerequisites and properties. Their capability to *discriminate* between relevant and non-relevant words as well as to *generalize* with respect to words that are relevant to the query will be discussed.

Feature representations for word spotting are strongly inspired from handwriting recognition and computer vision. Typically, features are based on the pen stroke (Section 3.2.1), on the appearance of the document image (Section 3.2.2) or features are learned according to semantic properties (Section 3.2.3). Examples for the different approaches can be found in Figure 9 to 11. It is important to note that features are computed for larger document image sections, as shown in the figures. Then, they are aggregated in order to represent document image regions, cf. Section 3.1. For representing a region holistically, a single feature vector is extracted. Features are extracted frame-wise in order to obtain a sequence of feature vectors or at interest points in order to obtain a set of local image features.

3.2.1 *Pen-stroke features*

Pen-stroke features encode the geometric shape of the writing. For this purpose, structural properties of the pen stroke are considered or pen-stroke pixels are encoded directly, cf. [TG14; FB14]. While these features have been studied and optimized extensively, it is essential that the pen stroke can be identified reliably. In order to do so, the document images have to be binarized or skeletonized, cf. [Gat14, pp. 75–77]. The pen-stroke contour in a section of a historic document image is shown in Figure 9. Typically, this can be achieved in modern document scenarios where the document images have high contrast. Due to the usually high sensitivity to even small variations of the pen stroke, these features require writing style normalization, like slant, skew or size, cf. [Gat14, pp. 112–122] and [FB14, pp. 397–398].

Inspired from speech and handwriting recognition, models using sequences of pen-stroke features have a long tradition. Commonly, features are extracted from single-column frames in the word- or text-line image. Popular features are the upper and lower contour of the pen stroke and statistics of the pen-stroke pixel distribution, like the number of foreground-background transitions. These features are used with HMMs and *dynamic time warping* (DTW). They can be found in the first approaches to word spotting [CWB93; KA94; KAA+00;

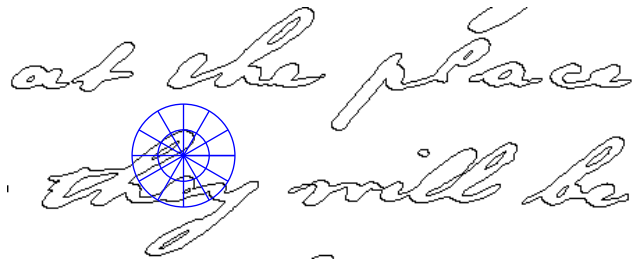


Figure 9: Feature extraction with shape context. The figure shows the contour of the pen stroke that has been obtained after binarization and edge filtering. Shape context descriptors are obtained at interest points along the contour. They capture the local pen-stroke pixel distribution by counting pen-stroke pixels in a radial grid of bins around the descriptor center. In the figure, a single descriptor is shown in blue, also cf. [RL14]. Document regions are represented with sets of shape context descriptors either directly or using them in a BoF framework, cf. Figure 5.

RM03] up to recent developments [TVR+16; MRR+16] including recurrent neural networks [FFM+12].

Apart from sequence models, many word spotting methods use holistic representations. With respect to a method's ability to discriminate and generalize, this has an important aspect. Statistical sequence models largely influence the specificity in the retrieval process with the chosen model architecture. However, if a document image region is represented holistically, i.e., with a single feature vector, retrieval becomes a simple nearest neighbor search. Thus, the feature representation mainly controls the specificity.

In order to obtain a holistic representation from the well-established sequential features, frequency-domain transformations, like discrete Fourier transform, cf. [GW02, Chap. 4.2], can be performed [KGG97; RLM03]. Lower-order frequency coefficients are used as feature vector. This approach has been successful for historic handwritten documents [KGG97; RLM03]. The generalization capabilities can be controlled through the number of coefficients.

More advanced representations are built on histograms. The main idea is to count discrete features. The features can either be counted in a grid of cells or in an entire document image region, see Figure 9. Histograms for different cells are concatenated. Therefore, the dimensionality of the feature vector does not depend on the size of the region, but only on the number of cells and on the number of histogram bins. The ability to chose suitable features and use them in an uniform framework, makes histogram representations extremely versatile and popular.

Popular representations that use histograms of pen-stroke features include the Loci descriptor, cf. [FLF11], the blurred shape model descriptor, cf. [FFF+11], and shape context, cf. [LS07]. The shape context

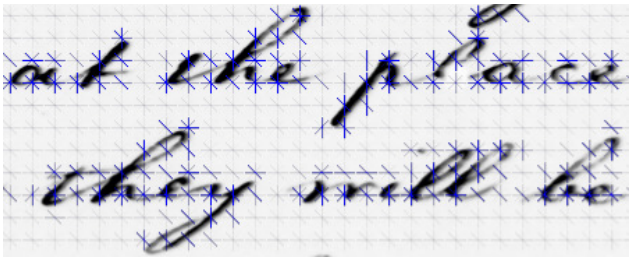


Figure 10: Feature extraction with gradient histograms. In the figure, gradient orientation histograms are arranged in a regular grid. The strengths of the different histogram orientations are indicated with gray to blue colors. The features represent the image texture and no binarization is required. They are robust with respect to contrast variations to a large extent, cf. [AFV13]. Representations are obtained by concatenating histograms in document image regions. For normalization, histograms are arranged in blocks in order to take the local document image context into account.

and blurred shape model descriptors count pen-stroke pixels within grids of cells. These grids need to have high resolution for obtaining a sufficiently specific representation, see Figure 9. The Loci descriptor, in contrast, is given by a single histogram of *Locu* codes [Glu67]. Each Locu encodes the number of foreground-background transitions in given directions starting from a reference point in the word image. Due to many different possible Locu codes, the Loci descriptor is sufficiently specific without subdividing a word image in a gird of cells.

3.2.2 *Appearance features*

Appearance features are designed for encoding entire document image regions and *not only* the pen-stroke specifically. Therefore, it is not necessary to detect the pen-stroke in the document image with binarization or skeletonization. This makes these representations more robust for applications in historic document images. Small variations of the visual pen-stroke appearance will result in small variations of the feature vector. In contrast, pen-stroke features can change rapidly if a small variation of the visual pen-stroke appearance changes the pen-stroke detection result.

The first word spotting method that was only based on appearance features has been presented in [RFR03]. Local image patches are extracted at Harris corner points [HS88] and their pixel intensity values are matched with *sum of squared differences*, cf. [Sze11, Sec. 8.1]. Although this representation does not represent the pen-stroke explicitly, it is very sensitive to variabilities in the document image, like background artifacts.

Appearance features that are more robust in this regard are based on *gradient histograms*, see Figure 10. The main idea is to quantize ori-

entations in gradient images. Within cell structures in document image regions, gradient magnitudes are accumulated in corresponding orientation bins. After histogram normalization, the representations are robust with respect to small document image brightness and contrast variations. The pen-stroke will dominate the orientations captured in the histogram as long as the gradients that are corresponding to pen-stroke edge-pixels have a larger magnitude than the gradients in the background. Furthermore, the representation is robust against varying pen-stroke width to a large extent.

The methods presented in [RP08a; TT09] use gradient histograms for obtaining sequential feature representations. The *local gradient histograms* presented in [RP08a] are SIFT [Low04] inspired. An important result is how to align the frame with respect to the text in a word or line image. Fitting the frame to the text area increases retrieval performance considerably. In contrast, the study in [TT09] is closely inspired by *histograms of oriented gradients* (HoG) [DT05]. The differences mainly lie in the structural parameters, such as cell layout or histogram normalization.

HoG are also widely used for holistic representations. Particularly in the context of segmentation-free word spotting, different approaches exist for achieving accurate retrieval results while keeping the computational complexity at query time low.

For retrieval based on word hypotheses, a holistic representation is considered in [KWD14]. HoG and *local binary pattern*, cf. [AHP06], descriptors are computed for the hypothesized document image regions initially. In order to improve the generalization capabilities, a random projection technique is applied. The process consists of two steps. In the first step, descriptors are linearly projected onto randomly selected prototype vectors. In the second step, the final feature representation is obtained by performing max-pooling of the projection coefficients with respect to a random partition of the projection-coefficient vector-space. Thus, for each subset in the random partition, a single value is obtained that is maximal compared to the values of the other vector components in the same subset. Therefore, the dimensionality of the final feature representation is equal to the number of subsets. It is important to note that the prototype vectors as well as the random partition are defined once and are used for all region and query representations. The prototype vectors are randomly drawn from the word hypothesis descriptors and the subsets in the random partition are sized uniformly. Based on this compact representation, distance-based retrieval can be performed. The method is inspired from face recognition [LLM+13]. This also explains the use of local binary pattern descriptors which have shown good performance in this domain.

For patch-based retrieval [AGF+14b; RKE16], it is possible to compute only HoG cells, instead of entire region descriptors, in the docu-

ment image initially. Groups of cells are dynamically combined in order to represent different patch hypotheses at query time. Although this is fast on the one hand, the hypothesis representations do not change smoothly for overlapping document image regions on the other hand. This can be problematic for selecting the most relevant hypotheses in the segmentation-free scenario. In [AGF+14b] this is addressed within query model estimation. The exemplary query word image is augmented with translated instances of itself. This is also confirmed experimentally for distance-based retrieval in [RKE16].

A solution to this problem are BoF representations, see Figure 5. By using SIFT features in a BoF framework, the positive properties of orientation histograms can be exploited. Problems for segmentation-free processing can be avoided at the same time. This is achieved by adding an additional layer in the modeling hierarchy, i.e., histograms of quantized SIFT descriptors. These histograms are computed on a coarser level than the gradient-orientation histograms and, therefore, change smoothly for overlapping regions. Furthermore, the generalization capabilities can be improved.

While the general applicability of BoF representations for word spotting has been investigated in [AD07], the benefits for segmentation-free processing have been presented in [RAT+11] first. For this purpose, patch-hypotheses are represented with temporal adaptations of spatial pyramids which add spatial information in writing direction. These are embedded in a *latent semantic indexing* subspace [DDF+90] in order to obtain more compact representations with increased generalization capabilities. Essentially, the subspace encodes feature co-occurrences which allows for better handling of redundancies and ambiguities in the original feature space.

Another powerful BoF extension is the *Fisher vector* [PSM10]. Fisher vectors are based on a stochastic visual vocabulary, given as a GMM. A document image region is represented by the log-likelihood gradient vector of the GMM parameters with respect to the SIFT features in this region. Therefore, the Fisher vector encodes how the model would have to change in order to optimally represent the SIFT features. If the SIFT descriptors are augmented with their relative position coordinates in the document image, Fisher vectors also encode spatial information, cf. [SPC12; GRF13].

The superiority of Fisher vectors over HoG descriptors is demonstrated in [AGF+14b]. However, due to the high computational demands these representations are mainly suitable for segmentation-based scenarios [AGF+14a] or for re-ranking [AGF+14b].

3.2.3 *Semantic features*

Semantic features are obtained by transforming document region representations such that they contain class information of the problem

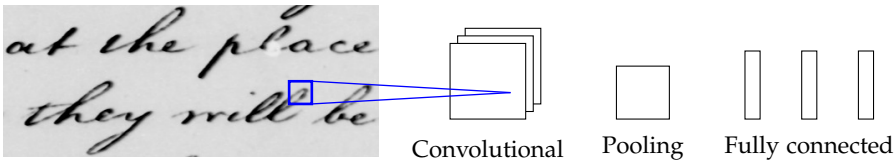


Figure 11: Feature extraction with CNNs. The figure shows the basic CNN building blocks and indicates how the input image is processed. Deep architectures are obtained by stacking convolutional and pooling layers consecutively. In order to predict a semantic representation for a given input image, a classifier is used that typically consists of three fully connected layers. If the output of the last fully connected layer is not directly suitable for retrieval, the region representations are obtained from one of the previous layers.

domain. Document image regions are represented as pixel intensities or with appearance-based features. Class information is learned on *character*, *word* or *attribute* level. For this purpose, training examples are required that are annotated accordingly. Attribute representations characterize a document image region with respect to *different* properties. This sets them apart from character-level and word-level representations that are specific to a *single* semantic unit. Attribute representations are learned from word-level annotations.

Sample data that is annotated on character level is either used in query-by-string scenarios where the visual variability in the document images is very limited, cf. [ETF+04; CZF06; CBG09; LOL+09], or it is automatically generated with an existing recognizer, cf. [JVZ14; TCH+15]. An early approach using semantic features has been presented in [CZF06]. Using manually annotated character templates, a class-discriminant subspace is estimated with linear discriminant analysis, cf. [DHS00, Sec. 3.8.3]. Similarly, local image descriptors have been linearly embedded such that semantic descriptor correspondences are reflected by their distances in the subspace [SRF15]. The training dataset consists of corresponding and non-corresponding descriptor pairs. It is automatically obtained from annotated word images. Descriptor relations are based on their distance in descriptor space and their relative spatial locations within the word images. For query-by-example word spotting, these features can be used within spatial pyramid BoF representations.

Furthermore, features for query-by-example word spotting that are learned on character level and on word level, are computed with CNNs [SK15; SRG16], see Figure 11. These CNNs use classification layers which assign a single class label to a given input. In order to use such networks for retrieval, a common approach is to discard one or more fully-connected classification layers of the CNN. Document region representations are based on the output of the network. In [SK15] the last fully-connected layer is removed. For fixed-sized input word images, the output is directly used as a feature vector. In

[SRG16] all fully-connected classification layers are discarded and a fixed-size representation is aggregated from the convolutional filter activations of the last convolution layer. This allows for more flexibility with respect to the input size of the word images. However, both representations have been estimated for classification and not for retrieval. This is where attribute representations have a considerable advantage.

The main idea of attribute representations is to define a string embedding for word labels and to learn a transformation of word images into the same attribute vector space, cf. [ART+13; AGF+14a; WB16; SF16]. Afterwards, the attribute vector is used as a feature vector and query-by-example and query-by-string can be performed in analogy to each other. Attribute representations for word spotting are typically based on character and n-gram frequencies [ART+13] or character and n-gram occurrences [AGF+14a; WB16]. In this regard, the most popular string embedding is the *pyramidal histogram of characters* (PHOC) [AGF+14a]. The binary embedding indicates presence and absence of characters in spatial sections of the string in a pyramidal fashion. Texts are, therefore, characterized by presence of characters on different spatial resolutions. Attributes become gradually more specific from lower to higher spatial resolution. This allows for modeling explicitly what different classes have in common. Similar words, e.g., *office* and *officer*, will have similar attribute representations. This is an advantage for learning from annotated sample data because the learning process is guided by expert design. For example, it does not have to be derived fully automatically that the distinguishing property between *office* and *officer* is a single character. This is the most important difference to learning word-level representations.

Approaches for transforming word images into attribute space include *latent semantic indexing* [ART+13], *support vector machines* (SVMs) [AGF+14a] and CNNs [WB16; SF16]. The most common method is to treat the transformation as multi-label classification, cf. [BWG10]. This requires the string embedding to be binary, as is the PHOC. Each attribute is considered as a separate class and is individually predicted with an ensemble of SVMs [AGF+14a] or with a CNN, i.e., the PHOCNET [SF16]. The CNN approach has the advantage that all attributes are predicted with a single classifier. This allows for sharing information between the different classes. In contrast, the SVMs are estimated individually. Furthermore, the CNN has a deep structure, which is trained in an end-to-end manner. The SVMs only correspond to the last neural network layer and require a given feature representation, like the Fisher vector, cf. [AGF+14a].

3.3 RETRIEVAL

Once numerical representations have been obtained, document image regions are analyzed according to similarity to the query. Mainly, two different approaches can be identified. Feature matching is *directly* based on similarity of the query and document region representations (Section 3.3.1). In contrast, model-based approaches define a structure *on top* of the feature space (Section 3.3.2). The model describes the query on a more general level, typically in terms of character classes or word classes. For this reason, it can be seen as an abstraction from the numerical representations. Model parameters are usually estimated from a larger number of annotated samples and similarity is computed with respect to the query model. In this regard, statistical sequence models, particularly HMMs, are most popular.

Finally, efficiency is an aspect that is important for all word spotting systems (Section 3.3.3). In order to achieve fast retrieval times, the number of document image regions and the complexity of computing similarity are important. A common strategy is to index region representations and retrieve relevant regions according to approximate similarity measures. Based on these candidates, the sorted retrieval list is obtained after re-ranking with more accurate methods.

3.3.1 *Feature-based similarity*

Retrieval becomes a nearest neighbor search, if the query is represented such that it can directly be compared with document regions in the same feature space, as shown in Figure 12. The query can be considered as a template and document image regions are sorted according to similarity to the query. While query-by-example scenarios can be addressed naturally in this manner, a possibility to support query-by-string in the same way is to synthesize a query template image [MMS03; LOL+09; LFG12]. Unfortunately, this limits the application domain to scenarios with only small visual variability in the document images. A more suitable alternative has been provided with semantic attribute representations, cf. Section 3.2.3.

For word spotting with feature-based similarity, it is important to choose a suitable similarity measure. Distances can be interpreted as negative similarities and can, therefore, be used analogously. Within lower dimensional feature spaces, measures are often based on Euclidean distance, e.g., [RM03; RFR03; GP09; FFF+11; FLF11; ZT13; KWD14]. Within higher dimensional spaces, Euclidean distance is not discriminative, cf. [NS06; PSM10]. This is due to the *squared difference* of vector components which emphasizes vector components with large differences. For high dimensional and histogram-like representations, similarity measures that take the histogram distribution into account have been successful, e.g., [ZSH03; AD07; RL14; AGF+14a;

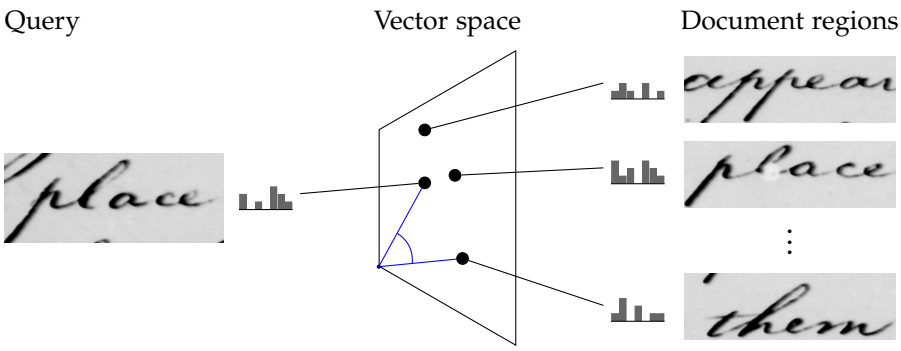


Figure 12: Feature-based similarity. The figure shows a query word image and document image regions that are represented in the same vector space. The holistic features encode visual appearance such that visual similarity corresponds to similarity in the vector space. For this purpose, the figure shows histogram representations for the word images. The similarity between two vectors is computed by their angle as indicated in blue, i.e., cosine similarity. Cosine similarity is independent of the vector lengths and emphasizes the distributions of values in the vectors.

SF15; RKE16; GV17; WLB17; RSR+17]. Measuring the angle between two vectors with cosine similarity, cf. [BR11, Sec. 3.2.6], is the most common approach in this regard. For this purpose, the similarities of R document image regions with respect to the query are stored in vector $\boldsymbol{\nu} \in \mathbb{R}^R$. The holistic feature representation of the query is denoted as $\boldsymbol{x}_q \in \mathbb{R}^D$ and the holistic feature presentation of the document image region with index $r \in \{0, \ldots, R-1\}$ is denoted as $\boldsymbol{x}_r \in \mathbb{R}^D$ where D is the dimensionality of the feature vector space. Equation 9 defines the cosine similarity for each component of the vector $\boldsymbol{\nu}$. In the segmentation-based scenario, the ranked retrieval list is obtained as the sequence of document regions sorted according to the corresponding similarity values. In the segmentation-free scenario, only the regions with the highest similarity values among overlapping regions are selected for the retrieval list, i.e., NMS. The concept of retrieval with cosine similarity is visualized in Figure 12.

$$\nu_r = \frac{\boldsymbol{x}_q^\top \boldsymbol{x}_r}{\|\boldsymbol{x}_q\|_2 \, \|\boldsymbol{x}_r\|_2} \tag{9}$$

Cosine similarity is obtained as the scalar product of the unit-length normalized vectors. Sorting document regions by minimal Euclidean distances of the corresponding unit-length normalized vectors, results in a ranking that is equivalent to the ranking that is obtained with cosine similarity. This is exploited in [ART+15].

Apart from measuring similarity between holistic document region representations, different approaches have been investigated for computing similarities based on local features. This way, the dynamic properties of handwritten script can be modeled in the matching process. The first method in this direction has been presented in [KH93].

In the printed modern document scenario, the similarity between two binarized word images is based on the Euclidean distance map, cf. [Dan80], of their XOR image. For every matching pen-stroke pixel indicated in the XOR image, the Euclidean distance map assigns the minimum distance to a non-differing pixel. By accumulating all minimum distances, a similarity score for the two word images is obtained. This score is less influenced by single outlier pixels than larger differing pixel blobs in the XOR image. For this purpose, it is crucial that word images are pairwise aligned.

The method has been extended to handwritten historic documents in [MHR96]. This was achieved by increasing the robustness in the matching process. Instead of a single, fixed alignment, a more sophisticated strategy including baseline estimates as well as different horizontal and vertical alignments are evaluated. For two word images, the best Euclidean distance map score that can be achieved with any alignment is considered. The method was the first to perform word spotting on handwritten and historic documents.

In order to increase robustness, more complex matching schemes have been used with local features. A simple example in this regard is *dynamic time warping* (DTW), cf. [SC78]. Similarity is based on an optimal alignment of two sequences of feature vectors [KAA+00; RM03; TNK05; TT09]. The features can be considered as local, because they are extracted frame-wise in writing direction, cf. Section 3.2. State-of-the-art results in the segmentation-based query-by-example scenario where no annotated training dataset is available have been reported with a DTW variant in [RLS+18]. Word images are divided in a sequence of overlapping zones. Each zone is represented with a descriptor that is based on discrete Fourier transform coefficients, cf. [GW02, Chap. 4.2], of orientation histograms. In order to compute the similarity of two word images, the dynamic programming matching-algorithm incorporates the spatial consistency of the zones.

A further degree-of-freedom is added if local features are extracted at interest points. In graph-based approaches the pen-stroke is represented with vertices and edges. Vertices are associated with local feature representations of the pen-stroke and edges define relations between them. For query-by-example word spotting the query word graph is matched with graph structures in the document images [KGG97; How13; WEG+14a; RLF15; SFR18b].

In a similar manner, local image features from the document images can be matched with local image features in the query word template. The spatial consistency of matching features is used as a similarity measure in [RFR03; ZT13; KKG15; HF16]. Alternatively, the cumulative distance [LLE07] or average distance [ZPG17] of features that have been matched in a cohesive elastic manner can be considered. Spatial consistency is enforced by restricting matches to local neighborhoods around reference points in the document images, e.g.,

based on RoIs and guides in [LLE07]. In [ZPG17] the objective is to be robust with respect to word size variability. After detecting potential query word *center* keypoints in the document image, keypoints in the local neighborhoods are projected in size normalized coordinate systems that are relative to the center points. Keypoints from the query are projected in a coordinate system that is relative to the corresponding center keypoint in the query word image as well. Matches between keypoints from the query and the documents are only considered for computing similarity scores if they are spatially consistent within the normalized coordinate systems. It should be noted that both approaches in [LLE07] and [ZPG17] heavily depend on manual parameter tuning.

Keypoint-based approaches establish the relation of local features in the query word image and local features in the document image directly. Spatial consistency measures are based on *corresponding* features. Graph-based methods obtain a relationship between features in *their* local neighborhoods first. Thus, graph-based methods can *directly* exploit these spatial relations in the matching algorithms, e.g., with *graph edit distance*, cf. [RB09], as in [WEG+14a; RLF15; SFR18b].

3.3.2 *Model-based similarity*

In order to perform word spotting with query models, the similarities of document region representations with respect to the query model are computed. For this purpose, the query can be modeled on word level or on character level. Word-level approaches are based on SVMs, e.g., [PR09; AGF+14b], or statistical models, e.g., [RLM03], including HMMs [RP12a; TVR+16]. Character-level approaches are mostly based on sequence models, such as HMMs [CWB93; CZF06; FKF+12], recurrent neural networks [FFM+12; SGL+16] or hybrids of HMMs and neural networks [TCH+15; BMC+15]. A common property of these methods is that all the models can, or have to, be estimated from multiple annotated training examples. Due to their flexibility with respect to the required amount of annotated training material, SVMs and HMMs are particularly relevant. Furthermore, both models have been used for segmentation-free word spotting.

An SVM defines a hyperplane that separates document region representations into relevant and non-relevant with respect to the query [PR09; AGF+14b]. Accordingly, it is estimated from multiple relevant and non-relevant examples. Distances to the hyperplane are interpreted as similarity scores in order to obtain the *ranked* retrieval list. In [PR09], class information is modeled on word level which limits the user to a predefined lexicon of query words. In contrast, [AGF+14b] considers a query-by-example scenario where no annotated training material besides the query word image is provided. For estimating the SVM, non-relevant examples are randomly sampled from the doc-

ument collection. Multiple examples that are relevant to the query are obtained by translating the query word region, cf. Section 3.2.2. Furthermore, this so-called *exemplar SVM* [MGE11] can be re-estimated after query expansion [AGF+14b], also cf. [BR11, Chap. 5].

Since SVM-hyperplanes separate the feature space they are referred to as *discriminative* approaches. Statistical approaches model the *generation* of feature vectors using statistical distributions over the document-region feature-space. Thus, retrieval is based on the probability of *generating* the document-region features with the query model. For this purpose, the common approach is Bayes' theorem, cf. Equation 1. The *likelihood* of the features conditioned on the query, is normalized with the *evidence*. The likelihood is weighted with a prior for the query word. Typically, the prior is assumed to be represented by an uniform distribution and is, therefore, neglected. The likelihood is represented by the query model. The evidence represents the distribution of the features in the feature vector space without being conditioned on any specific class model. Since the logarithm of this *odds ratio* is considered in practice, this form of normalization is referred to as log-odds scoring [BHK97]. Disregarding the prior, Equation 10 expresses the score for the query with respect to a document region in terms of Equation 1.

$$\log \mathit{posterior} \approx \log \mathit{likelihood} - \log \mathit{evidence} \tag{10}$$

In contrast to the evidence, the likelihood is conditioned on the query. Thus, the evidence should theoretically be greater than the likelihood or equal to the likelihood. Consequently, the highest similarity value in the logarithmic domain is theoretically zero. The score is an approximation of the query *posterior* probability.

In practice, modelling the evidence is a major challenge because it has to represent the semantic structure over the entire feature vector space. In [RLM03], this is achieved by limiting the queries to a lexicon. The evidence is modeled as the total probability of the likelihoods over all word classes. The quality of a score depends on how well the corresponding document region is represented by any of the class models. The restriction to a lexicon is avoided in [RP08b] by representing the evidence as a GMM over the feature vector space. Unfortunately, this results in a model that is unspecific to the semantic structure of the problem domain. Semantic information is only incorporated in the query model which causes difficulties for scoring document regions that are *not* represented well by the query model.

The most common statistical method for query-by-string word spotting are HMMs. A widely noticed approach is to model the occurrence of the query word in a text line [FKF+12]. Figure 13 visualizes the overall process. Given a document image, the segmented text line images, cf. Section 3.1.1, are normalized and represented with sequences of feature vectors, cf. Section 3.2.1. Within the statistical approach, the

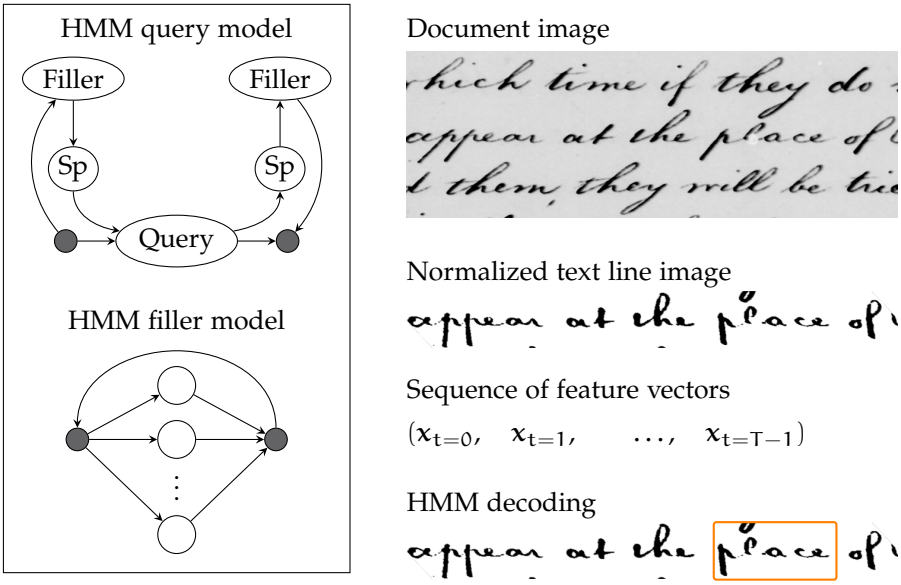


Figure 13: HMM-based word spotting system. For decoding a text line, the probability of generating the sequence of feature vectors with the *query* HMM versus generating the sequence with the *filler* HMM is considered. Start and end nodes are shown in gray. The other circles denote character models, where *Sp* stands for *space*. Models representing sequences of characters are shown with ellipses. Arrows indicate possible transitions between models. The orange box in the bottom right visualizes the most plausible decoding result for the query `place`.

likelihood is represented by an *HMM query model* and the evidence is modeled by the so-called *filler* model, cf. Equation 10. The filler models an *arbitrary sequence* of characters. The query model is a compound HMM which consists of three different models. The *query* word is modeled by concatenating the corresponding character HMMs. Equation 11 shows an example for the query word `place`.

$$\lambda(\texttt{place}) = \lambda(\texttt{p}) \circ \lambda(\texttt{l}) \circ \lambda(\texttt{a}) \circ \lambda(\texttt{c}) \circ \lambda(\texttt{e}) \tag{11}$$

The *space* model and the *filler* model represent the context of the query word within the text line. The structure of the compound query model defines transitions between the models such that it models appearances of the query word at the beginning, at the end or in the middle of the text line. Though the space model is also included in the filler model, its explicit occurrence in the compound query model helps to recognize if shorter words are appearing within longer words. In the following, the *compound* query HMM will be denoted as $\lambda_{\mathfrak{k}}$ and the *filler* HMM will be denoted as $\lambda_{\mathfrak{f}}$.

A probabilistic score that is based on an approximation of the query posterior probability given the feature vector sequence of the text line is obtained from the quotient of the compound query model score and filler model score, cf. [FKF+12]. For this purpose, L line images

are represented with feature vector sequences $\mathbf{O}^{[l]}$ with index $l \in \{0, \ldots, L-1\}$. For each line image, the similarity score is stored in vector $\boldsymbol{\nu} \in \mathbb{R}^L$ as shown in Equation 12.

$$\nu_l = \sqrt[\widehat{F}_q]{\frac{p\left(\mathbf{O}^{[l]}, \mathbf{S}_t^* \mid \lambda_t\right)}{p\left(\mathbf{O}^{[l]}, \mathbf{S}_f^* \mid \lambda_f\right)}} \tag{12}$$

For the line image with index l, the likelihood is approximated with the optimal output probability $p\left(\mathbf{O}^{[l]}, \mathbf{S}_t^* \mid \lambda_t\right)$ for the feature vector sequence $\mathbf{O}^{[l]}$ and the optimal state sequence $\mathbf{S}_t^*$ obtained for the compound query HMM λ_t. In analogy, the evidence is approximated with the optimal output probability for the feature vector sequence $\mathbf{O}^{[l]}$ and the optimal state sequence $\mathbf{S}_f^*$ obtained for the filler HMM λ_f. Due to the varying length of the query word detections, the odds ratio is finally normalized with the length $\widehat{F}_q$ of the most likely occurrence of the query within the feature vector sequence. After normalization, the score is the geometric mean over the approximated state sequence probability that corresponds to the query word, according to the alignment obtained for model λ_t.

However, this approach for scoring text lines according to relevancy with the query is not without problems. Although no explicit transcription of the text line has to be provided, the quality of the score depends on the recognition result that is computed in the filler model. As formally shown in [PTV15a], the query model score can be interpreted as a score for a word-level transcription result that contains the query word and the filler model score can be interpreted as a score for any word sequence. Consequently, transcription errors will result in smaller differences between query model scores and filler model scores, even though the query word might not have been spotted correctly. This produces *false positives*. Empirically, the effect can be confirmed in different HMM-based word spotting approaches that have increased recognition capabilities through integration of a bigram *language model* (LM) [FFB+13], higher order *n*-gram LMs [TPV15; TPV16] and lexicon-based recognition [PTV14; TVR+16].

Word spotting that is based on word-level recognition with HMMs has been approached with *word graphs* [PTV14; TVR+16]. Word graphs represent the most plausible transcriptions of a text line on word level. Word segmentation hypotheses are organized in a directed acyclic graph for this purpose. Each node is labeled with a frame position in the feature vector sequence of the text line image. Each edge is labeled with a word and a score indicating if the word occurs within the associated positions in the feature vector sequence. The scores are referred to as *edge posteriors* and are obtained using *forward* and *backward* variables in analogy to state transition posterior probabilities in the Baum-Welch algorithm, cf. [Fin14, Sec. 5.7.4]. Frame-level query posterior probabilities are computed by accumulating edge posterior probabilities according to the following two conditions. All edge posterior

probabilities are taken into account that are labeled with the query word. The frame positions, which are associated with these edges, have to enclose the frame position for the frame-level query posterior probability considered. The similarity score for the text line is given by the *maximum* frame-level query posterior probability within the text line. Word graphs have the drawback that they restrict the user to query words from a predefined lexicon.

Query-by-string methods addressing this limitation have been presented in [PTV14] and [TPV16]. In order to spot query words that are not part of the word graph, a fallback to a character-based filler approach is used in [PTV14]. Similar to the word graph, a character lattice represents possible transcription hypotheses of the text line on character level [TV13]. Since the filler score is independent of the query word, it is obtained as the maximum score of all complete paths in the character lattice. This corresponds to the score of the optimal transcription of the text line image. The query model score is obtained as the maximum score for a path containing the query character sequence. The text line is ranked based on the log-odds scores for the query word model and the filler model. Word graph and character lattice scores are combined following a backing-off strategy. This refers to the fallback from a specialized to a more general model, also cf. [Fin14, Sec. 6.5.5].

In [TPV16] frame-level character sequence posterior probabilities are computed for line images in close analogy to the computation of frame-level word posterior probabilities in word graphs. For this purpose, a character lattice is computed first. Sequence posterior probabilities are obtained based on edge posteriors which represent the probability for characters in the given section of the feature vector sequence. These character sequence probabilities are also referred to as *posteriorgrams*, cf. [HSW09]. For ranking line images according to relevancy with the query, posteriorgrams are normalized with respect to query length.

The difference of the word graph and posteriorgram approaches in comparison to a character filler approach is that query posterior probabilities are not based on the optimal but the total output probability of possible text line transcriptions, cf. [TVR+16; TPV16]. As formally derived for the character filler approach [PTV15a], using the log-odds of optimal output probabilities can be seen as an approximation to using the log-odds of total output probabilities. Regarding retrieval performance, word graphs benefit from the lexicon. By integrating a language model, results can be improved even further [TVR+16]. A direct comparison is possible between character filler and posteriorgram approaches. As can be seen in the results reported in [PTV15a] and [TPV16], results consistently improve with higher order LMs. In this regard, improvements are considerably better if scores are based on the total output probability.

With respect to query-by-example word spotting a method based on word graphs was presented in [VTP15]. The formal derivation of their method boils down to performing an n-best word-level recognition of the example image. According to the recognition probabilities, the n-best results are then used as queries for decoding the word graphs of all line images in the document collection. In this scenario the user is not limited to a given lexicon of query words anymore. However, retrieval can be expected to fail if the query word image is not at least similar to any of the words in the lexicon.

Finally, HMMs have also been used for query-by-example word spotting scenarios where no annotated training material but only the exemplary occurrence of the query word is given. In order to estimate a query word HMM from a single example, a *semi-continuous* (SC) model is used in [RP09b]. The shared GMM is estimated in an unsupervised manner and only the state-dependent mixture and transition probabilities are estimated at query time. Since no annotated training samples are available for estimating a filler model, the GMM is used as a so-called *universal background model* [RP08b] for HMM score normalization. Word region hypotheses are ranked according to the log-odds scores of the query model and the background model, cf. Equation 10.

A possibility to avoid score normalization for SC-HMMs has been proposed in [RP12a]. In contrast to [RP09b], not only the query word but each word image region is modeled with an HMM. Similarity between the query word HMM and the word region HMMs is then measured by DTW between state-dependent mixture model weights. This is possible due to the shared GMM in the SC setting. Similarity between mixture weight vectors is based on the discrete Bhattacharyya coefficient [Bha43], cf. [CRM00], which measures the similarity of two discrete probability distributions.

3.3.3 *Efficiency*

In practice, word spotting systems have to search large collections of document images. In order to guarantee fast retrieval times, operations that are independent of the query are performed initially. At query time, this indexed information is accessed efficiently, thus, reducing the computational effort required for obtaining potentially relevant document image regions. Data structures that integrate well with the document region representations are essential for this purpose. The set of candidate regions can be ranked according to similarity measures that are adjusted to these data structures [LS07; AGF+12; RAT+15a; RKE16; TPV15; TVR+16]. Since this adjustment often results in approximate similarities, additional re-ranking can improve retrieval results considerably [SJ12; AGF+14b; RLF15; GV15a; SFR18b]. It has to be noted that the methods [KWD14; WLB17; GV17; RSR+17] address the efficiency aspect by keeping the number of word hy-

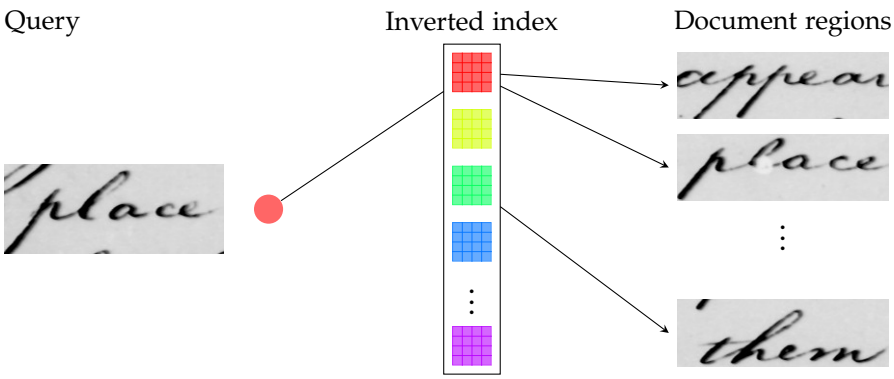


Figure 14: Document region indexing. The process of retrieving regions through an IFS is visualized. Document image regions are represented in terms of codewords. Here, the codewords are shown as visual words that are used in BoF representations, cf. Figure 5. Five visual words are indicated within the *inverted index*. Associations between codewords and document image regions are exemplarily indicated with arrows. In order to retrieve regions that are relevant with respect to a query, the codewords that are associated with the query are used for look-ups in the inverted index. In the figure, this is indicated for the red visual word.

potheses low in their segmentation-free application scenario, cf. Section 3.1.3. In order to improve scalability, indexing strategies for word hypothesis representations could be applied in analogy to [SJ12] or [RDE+14; AGF+14b].

Using index structures in order to solely reduce the number of region candidates offers the best possibilities for making a trade-off between efficiency and accuracy. In this scenario word spotting is performed in a two-stage process. In the first stage the objective is to optimize precision and recall. Precision is important in order to obtain short retrieval lists. In addition, recall is important in order to obtain retrieval lists that include the relevant regions at the same time. Computationally expensive methods for optimizing the ranking in the resulting retrieval list are applied in the second stage. Therefore, retrieval lists are mostly affected by reduced recall. A trade-off between efficiency and recall has to be made by selecting the number of re-ranked regions, cf. e.g., [AGF+14b]. In contrast, methods that do not perform re-ranking have to find a trade-off between efficiency and retrieval performance in general. Approximate similarity measures tend to affect recall as well as precision at different recall levels in a similar manner [RAT+15a].

The most common data structure for efficient retrieval is the *inverted file structure* (IFS), cf. [BR11, Chap. 9.2]. Figure 14 visualizes the concept. The basic idea is to represent document image regions with a codebook of feature codewords. Typically, the codebook is either defined heuristically [LS07; FLF11; GV15a; RLF15; GV18] or obtained automatically through clustering [SJ12]. The *inverted* index contains

an entry for each codeword and stores links to the associated document image regions. Once the codewords have been obtained for the query, the index allows for fast look-ups. The purpose of the IFS is to retrieve as few document regions including mostly all document regions that are relevant to the query. The successful application depends on the average number of IFS entries per codeword. For example, if a codeword is relevant to many document image regions, all of these regions will have to be processed after the look-up for this codeword. Different strategies can be followed for combining the results of multiple IFS look-ups, cf. [BR11, Sec. 9.2.3]. The union of retrieved regions is considered in order to emphasize recall whereas the intersection of retrieved regions emphasizes precision. Further, it is possible to obtain a (top-n) ranked retrieval list by encoding non-binary relevance information for regions with respect to the codewords in the IFS [SJ12; GV15a], cf. [BR11, Sec. 9.2.4].

Codebooks used in inverted indices are based on shape context [LS07] and Loci [FLF11] descriptors, binary attribute representations [RLF15; GV15a; GV18] or BoF [SJ12], cf. Section 3.2. Index codewords for the local descriptors are implicitly defined by enumerating over all possible descriptor instances. For attribute representations, the attributes can also be used as index codewords in analogy to using visual words as index codewords for BoF representations, see Figure 14.

An interesting IFS application is presented in [LS07]. By indexing shape context descriptors, the IFS is used for efficient feature matching. Shape context descriptors from the query vote for word images in which they are occurring. Votes are accumulated and used as similarity measure. In [RLF15] an approximate similarity measure for subgraph matching is based on the same idea. The voting scheme perfectly integrates with the IFS. At query time, it reduces the computational effort to simple additions. This is extremely efficient assuming that there exist only few IFS entries for the codewords which are obtained for the query.

The most widely and successfully used approximate similarity measure for word spotting is based on *product quantization* (PQ) [JDS11]. It has been applied to segmentation-free word spotting [AGF+12; AGF+14b; RAT+15a; RKE16] and is capable of handling large amounts of region representations that are obtained in patch-based approaches, cf. Section 3.1.2. The key idea is to compress document region representations by quantization. In order to achieve high accuracy despite the compression, the quantization error has to be as small as possible. This can be optimized with large codebooks. For example, in order to encode a vector with a 64-bit centroid index, $k = 2^{64}$ centroids are required. However, estimating codebooks at this size is computationally demanding and requires a multiple of k samples [JDS11]. PQ addresses this problem by quantizing disjoint sets of feature vector components independently. These are given as a partition of all

components of the feature vector space. Afterwards, a vector is encoded by concatenating the centroid indices obtained by the so-called *sub-quantizers*. This encoding as well as the *global* codebook are implicitly defined as elements of the Cartesian product of sub-quantizer centroid indices and sub-quantizer codebooks, respectively. The approach is referred to as *product quantization* for this reason. As a result a global codebook of size 2^{64} can be obtained with 8 sub-quantizers using codebooks of size 256 each [JDS11], for example.

While compression is useful for storing large amounts of document image representations in memory, retrieval speed can be improved by computing approximate distances based on the centroids that are associated with the query representation and the region representations. For this purpose, precomputed look-up tables contain pairwise distances of the centroids for each sub-quantizer. Afterwards, the similarity measure is simply based on accumulating distances obtained from the look-up tables.

Assuming that only few sub-quantizers are required, computational complexity is linear in the number of document image regions. In order to avoid an exhaustive search, a two-stage integration with an IFS for indexing large databases is proposed in [JDS11]. While this approach has not been investigated for word spotting, yet, a pyramidal matching scheme is applied with the same motivation in the patch-based segmentation-free scenario considered in [RKE16]. The pyramid is based on a Gaussian scale space, representing document images from coarse to fine details, cf. [Low04]. PQ compressed features are extracted at each scale and used for representing patches in analogy to [AGF+12; AGF+14b]. In the pyramid, the number of patches increases from coarse to fine scales. The document image area of a single patch on a coarse scale is represented by multiple overlapping patches on a finer scale. After matching the query on coarse scales, candidate patches are passed on to finer scales and the overall search space is reduced.

The indexing strategies that have been presented so far have primarily been used for word spotting with feature-based similarity measures, cf. Section 3.3.1. Model-based approaches are not as well established in this regard. One exception is the *exemplar SVM* [AGF+12; AGF+14b]. After model estimation, the weight vector is used for retrieving document image regions according to approximate similarity with PQ. The only other exceptions are the word graph [TVR+16] and the character lattice [TV13; TPV15; TPV16] approaches that address the efficiency aspect for HMMs.

A common property of the word graph and the character lattice, cf. Section 3.3.2, is that they model n-best recognition results in a graph structure. Therefore, the recognition result is completely independent of the query word and can be precomputed for each text line image. The retrieval time depends on the complexity of the graph since all

edges that are relevant for the query have to be searched. The complexity of the graph is controlled during text line decoding. The more transcription hypotheses are modeled in the graph the higher is its complexity.

3.4 DISCUSSION

Word spotting systems allow for accessing document collections rapidly, thus, avoiding tedious manual exploration. However, this is only possible if they do not require a lot of manual preparation in order to work with a new dataset. This manual effort typically lies in the partial annotation of the dataset and in the meta parameter fine-tuning. It has to be noted that expert knowledge regarding the interpretation of document images as well as the interpretation of meta parameters is mandatory for this purpose. In the worst case, even the design of novel methods might be required, e.g., for document image segmentation, cf. Section 3.1.1, or for pen-stroke features, cf. Section 3.2.1, that are adjusted to the special dataset characteristics.

Fine-tuning meta parameters in order to optimize retrieval accuracy is beneficial for any word spotting method. The sensitivity of different meta parameters often depends on the number of model parameters that can be estimated from training data automatically. If a meta parameter is very sensitive, it has to be adjusted with care because it will greatly influence the overall system performance. Unfortunately, the complexity increases with the number of model parameters and more and more training data is required. In this regard it is important if the training data has to be annotated and on which level the annotations are required. For example, creating line-level annotations is considerable less manual effort than annotating on word level. Representations that can be estimated without annotated data in an unsupervised manner, e.g., BoF, are particularly interesting in this regard.

In summary, the ideal word spotting method should minimize the manual effort by minimizing the number of sensitive meta parameters and the number of annotated training samples. In addition, it has to be accurate and fast at the same time. Due to the fact that all these requirements are hard to meet jointly, methods make trade-offs. For example, if the number of annotated training samples should be low, i.e., down to a single sample in query-by-example scenarios, heuristics are applied in order to generalize to unseen occurrences of the query word and in order to discriminate word instances that are visually similar but irrelevant to the query. However, due to the limited evidence, the ability to discriminate will decrease as the generalization capability increases and vice versa. Furthermore, the manual effort of locating an occurrence of an infrequent query word can be substantial in query-by-example scenarios. If more annotated training

material is available, this effort can be avoided with query-by-string. In general, also retrieval performance will improve due to the improved generalization and discrimination capabilities of the models.

A comprehensive list of word spotting methods and a categorization with respect to their characteristic can be found in Appendix C. In the remainder of this section, methods will be compared with respect to selected aspects. The methods are either segmentation-free on document level (Section 3.4.1) or based on HMMs (Section 3.4.2). This is due to the following desirable properties which can be derived from the requirements that have been discussed above.

Under the heuristic assumptions for word hypothesis generation, segmentation-free methods do not have to be adapted in order to be applied to a new dataset, cf. Section 3.1. As these hypotheses represent alternatives to each other, word hypothesis generation does not include as many heuristic assumptions as segmentation. Therefore, it can be expected to be more robust.

If a segmentation on line level can be obtained reliably, HMMs are the most prominently used sequence models for word spotting, cf. Section 3.3.2. Sequence models have the advantage that the segmentation within the text line is implicit. It is based on aligning frame representations with the models. Given a line segmentation, approaches for defining and representing frames can be expected to be more robust than approaches for word hypothesis generation. Furthermore, HMMs offer great flexibility with respect to the required amount of annotated training examples. This sets them apart from recurrent neural networks and CNNs where large amounts of annotated samples are mandatory for model estimation.

3.4.1 *Word spotting on document level*

Table 1 shows an overview of word spotting methods that are segmentation-free on document level. The methods are characterized with respect to how region *hypotheses* are generated, how the large numbers of hypotheses are processed *efficiently* and how hypotheses are *selected* among competing hypotheses. Besides the *method* used for ranking the hypotheses according to similarity to the query, cf. Section 3.2 and Section 3.3, these are the aspects that are characteristic for segmentation-free word spotting. Furthermore, the prerequisites and limitations of the methods can be identified.

Approaches for generating hypotheses are based on patches or on text detection. Patch-based approaches are denoted with *dense patches* and *descriptor matches* in Table 1. All other generation approaches rely on text detection. *CC prototypes* are obtained through clustering. Detector-based approaches have the advantage that they are robust with respect to word size variabilities. Patch-based approaches have advantages if words are touching. The latter is particularly problematic if

CCs are used as text detection result. The only method that is able to cope with touching words and bases hypothesis generation solely on CCs is presented in [RLF15]. This is achieved by CC splitting and subgraph matching which is a considerable computational effort. Hypotheses generated with the region proposal network [WLB17] are potentially able to handle touching words as well. In contrast to [RLF15], large amounts of annotated training samples are required.

For efficient retrieval in the segmentation-free scenario, *word region* hypotheses extraction, *local feature* matching, *inverted indices* and approximate similarity, e.g., based on *PQ*, are most popular. By computing representations prior to retrieval, the computational complexity is reduced at query time. Word regions have the advantage that the search space is reduced to the relevant document image areas. However, if relevant word regions are not among the hypotheses, limitations as in the segmentation-based scenario apply, cf. Section 3.1.1. In a similar manner, local features are typically detected in text regions. In contrast to the word region approach, it is not required to detect word boundaries. However, in order to detect local features reliably and reproducibly in document images, large numbers of keypoints must be extracted, e.g., [HF16]. Afterwards, a large number of potential matches between the features from the query and the features from the document images have to be analyzed, e.g., [ZPG17].

Word region and feature matching approaches aim at keeping the number of region candidates low. In contrast, IFS and PQ are most suitable for selecting potentially relevant regions from a large number of region candidates. PQ can be seen as a generalization of the IFS-based voting scheme for computing approximate similarities between high-dimensional vectors. Given the sub-quantizer codewords that the query and the document regions are represented with, look-up tables store pairwise distances between codewords for each sub-quantizer. The look-up tables can be seen as inverted indices. The approximate distance between two vectors is given by accumulating the precomputed distances between the two codewords obtained for each sub-quantizer. This can be seen as a voting scheme that is based on IFS look-ups. Consequently, the IFS voting scheme can directly be applied if the feature representation is suitable. This avoids the additional effort of estimating sub-quantizers. It can even be more accurate since it builds on codeword representations that are part of the original feature design. Furthermore, it can also be faster since not all regions have to be processed necessarily. However, it has to be noted that for applications of IFS and PQ considerable improvements can be achieved with re-ranking. This is due to the mostly approximate similarity obtained in the voting schemes.

Finally, almost all methods select regions for the retrieval list with NMS. This has the advantage that the regions do not overlap and are, therefore, independent to each other as in the segmentation-based

scenario. In Table 1 only four methods do not select regions with NMS. Most notably, alternative approaches are followed in [HF16] and [ZPG17]. In [HF16], keypoints belonging to a keypoint configuration that matches with the query are removed. Thus, a specific keypoint in the document image can only be used for retrieving a single region. In [ZPG17], overlapping regions are merged instead of selecting only the one with the highest similarity score.

3.4.2 *Word spotting with hidden Markov models*

Table 2 shows an overview of HMM-based word spotting methods. The characteristic properties are the *output model*, the approach to *retrieval* and *score normalization*. These properties allow conclusions regarding the visual variability that can be modeled as well as the prerequisites such as a lexicon or the amount of annotated training data that is required.

An interesting aspect of word spotting with HMMs is that *probabilistic* scores can be obtained for potentially relevant document image regions. This is due to the stochastic approach of HMMs and sets them apart from most word spotting methods that only allow for obtaining a ranking. Probabilistic scores can be useful if retrieval results are processed by another automatic system. For users, similarity visualizations, e.g., using color coding, can be improved. However, apart from the query model, the stochastic distribution of text, i.e., the filler model, has to be estimated in order to obtain the posterior probability of the query. Although this does not involve a transcription explicitly, an n-best transcription is performed implicitly. Therefore, the filler can be considered as a full recognition model which is not easy to obtain in all word spotting scenarios. Table 2 shows methods that make trade-offs from filler models that are based on full recognizers including LMs over a background model that solely approximates the statistical feature distribution to an approach that takes advantage of the sequential modeling but avoids score normalization altogether.

Output models that have been used for word spotting with HMMs are largely based on *Gaussian* distributions. A single density per state has been sufficient for the printed document scenarios considered in [CWB93; ETF+04]. For handwritten documents, GMMs have been used. For query-by-example word spotting where no further training material is available, SC-HMMs have the advantage that the GMM can be estimated in an unsupervised manner from the feature vectors in the document images. Afterwards, the single example is *only* used for estimating transition and mixture probabilities [RP09b; RP12a]. The approach has also been extended to query-by-string where the query word model is estimated from multiple synthetically generated query word samples in the same manner [RP12b]. The gap between the synthetic fonts and the handwritten word images is bridged by

always using the original GMM which represents the feature vector distribution of the handwritten word images. In cases where large amounts of annotated training data are available, a GMM using diagonal covariance matrices is estimated per state. This increases the number of free parameters which can lead to more powerful models, e.g., [FFB+13; TVR+16]. A noteworthy approach that is not based on Gaussians is the deep belief network integration in [TCH+15]. The deep belief network predicts character posterior probabilities which are decoded with an HMM. However, frame-level annotations are required for training the deep neural network.

For retrieval with HMMs, mainly three different approaches can be identified. In the traditional *query-filler ratio* scoring, see Figure 13, the odds of the optimal path probabilities (Viterbi algorithm) for the query and the filler models is used, e.g., [FKF+12]. This can be considered as an approximation of the posterior probability of the query given the document image region [PTV15a]. The methodologically sound approach is to predict the query *posterior probability* with the total probability of any path that is relevant for the query (forward-backward algorithm). Similar to the computation of state posterior probabilities, cf. [Fin14, Equ. 5.14], the probability for any transcription that contains the query word is normalized with the probability for any transcription of the document region. Thus, the quality of the query posterior probability directly depends on the quality of the recognition. It should be noted that the recognition result is represented by the filler model in the *query-filler ratio* approach.

Finally, it is also possible to avoid scoring by considering the relevance of a document image region as a two class *recognition* problem. The probabilities for transitioning into the query or the filler model have to be adjusted to the users needs, e.g., in order to favor precision or recall, cf. [TCH+15]. Therefore, a ranked retrieval list is only obtained after multiple recognitions with varying transition probabilities. Only the approach presented in [RP12a] goes even further and bases retrieval on DTW distances of state mixture weights. This way, different HMM models can be compared directly. In contrast to the other HMM-based methods, it is required that an HMM is estimated for each document image region.

Closely related to the retrieval approach is *score normalization*. For the majority of the methods in Table 2, normalization depends on the filler model. It indicates the complexity and the structure of the recognition model. In this regard the LM integration is particularly important, because it allows for the best retrieval performance that can be achieved with HMM-based methods. Higher order n-grams ($n > 2$) allow for considerable improvements if the query posterior is based on the *total* output probabilities [TVR+16; TPV16].

Method	Hypotheses	Efficiency	Selection
Keyword signature matching [KGG97]	Dense patches	Frequency domain	n-best patch scores
Cohesive elastic matching [LLE07; LOL+09]	Word starting points	Local features	NMS of word starting point scores
Connected components [MC09]	CC prototypes	Word regions	All CC prototype matches
Patch-based matching [GP09]	Dense patches	Text area detection	NMS of patch scores
Heat kernel signatures [ZT13]	Descriptor matches	Local features	NMS of descriptor sequence scores
Inkball models [How13], cf. [PZG+14]	Dense patches	None	NMS on patch scores
Random projections [KWD14]	CC grouping	Word regions	NMS of CC group scores
Exemplar SVM [AGF+14b]	Dense patches	PQ codebook	NMS of patch scores
Spatial pyramid hashing [RDE+14]	Dense patches	Probabilistic hashing	NMS of patch scores
Graph embedding and indexing [RLF15]	CC prototype graph	IFS on subgraphs	NMS of subgraph similarity scores
Spatial pyramid indexing [RAT+15a]	Dense patches	PQ codebook	NMS of patch scores
Attribute SVMs [GV15a], cf. [GV15b; GV18]	CCs and patches	IFS on CCs and integral histogram	NMS of patch scores
Feature matching [KKG15; ZPG17]	Descriptor matches	Local features	NMS [KKG15] or merging [ZPG17] of spatial consistency scores
Relaxed feature matching [HF16]	Dense patches	Local features	Spatially consistent matches
Scale-space pyramid [RKE16]	Dense patches	Pyramidal refinement	NMS of patch scores
Region proposal network [WLB17]	CNN region proposals and CC group classification	Word regions	NMS of region scores
R-PHOC [GV17]	CC group classification	Word regions	NMS of region scores
Word hypotheses [RSR+17]	ERs on detector scores	Word regions	NMS of region scores

Table 1: Segmentation-free word spotting methods overview. The table shows properties that are characteristic for methods that do not require any document image segmentation on word- or line-level.

Method	Output model	Retrieval	Score normalization
Character HMMs [CWB93]	Gaussian (whitespace GMM)	Query–filler ratio	Sub-character filler
Pseudo 2D-HMMs [KA94]	Discrete (pixel probabilities)	Query–filler ratio	Word filler
Generalized HMMs [ETF+04], cf. [CZF06]	Gaussian, variable frame width	Query–filler recognition	Context-dependent character filler, query–filler transition probabilities
SC-HMMs [RP09b]	Shared GMM	Query–GMM ratio	Universal background model
Synthetic queries and SC-HMMs [RP12b]	Shared GMM	Query–GMM ratio	Universal background model
Model-based sequence similarity [RP12a]	Shared GMM (no covariance)	DTW on HMM state mixture weights	Warping path length
Character HMMs [FKF+12]	GMM (no covariance)	Query–filler ratio	Character filler, query length
Character lattice [TV13]	GMM (no covariance)	Query–filler ratio	Character filler, query length
Character HMMs and bigrams [FFB+13]	GMM (no covariance)	Query–filler ratio	Character-LM filler, query length
HMM n-gram-character lattice [TPV15]	GMM (no covariance)	Query–filler ratio	Character-LM filler, query length
Deep HMM [TCH+15]	Deep belief network	Query–filler recognition	Character filler, query–filler transition probabilities
HMM word graphs [TVR+16; VTP15]	GMM (no covariance)	Word posterior probabilities	LM word graph
HMM n-gram-character lattice [TPV16]	GMM (no covariance)	Character sequence posterior probabilities	LM character lattice, query length

Table 2: HMM-based word spotting methods overview. The table shows properties that are important for word spotting with HMMs.

4

SEGMENTATION-FREE WORD SPOTTING WITH BAG-OF-FEATURES HIDDEN MARKOV MODELS

The proposed word spotting method retrieves document image regions according to similarity to a query word image or a query string. No segmentation of the document image into lines or words is required. Document images are processed in an effort to minimize assumptions about the visual appearance of script. Since the required amount of training annotations should be low at the same time, constraints aim to be as general as possible. This is achieved by:

- detecting query words in a patch-based framework that integrates aspects from retrieval based on line segments and text hypotheses (Section 4.2),
- using BoF representations that are automatically adapted to the visual characteristics of the document images (Section 4.3),
- modeling BoF in the HMM process such that the statistical properties of the BoF representations are considered (Section 4.4),
- modeling the query with HMMs in order to take the sequential structure of text into account (Section 4.5),
- performing retrieval with two decoding stages that are fully integrated with each other in order to obtain accurate results fast (Section 4.6).

The most important assumptions for word spotting with BoF-HMMs are related to the document layout and the visual variability of the text. For document region extraction and processing, it is required that the text has a horizontal orientation. Otherwise, the text orientation has to be normalized first. It is, specifically, not a restriction if words are touching. Furthermore, length variabilities can be handled to a large extent.

With respect to the visual variability of the text, the amount of annotated samples that are available for query model estimation is crucial. Relevant words that are visually dissimilar to the training samples, will not be retrieved with high accuracy. However, due to the statistical sequence model, this can be compensated if only parts of words are not well represented by the model. This is particularly important for the query-by-example scenario where only a single annotated sample of the query word is provided.

In the following, the architecture of the overall word spotting system will be outlined (Section 4.1). This way, the methodological contributions can be presented in relation to each other.

4.1 ARCHITECTURE

The proposed word spotting system includes methods for generating document region hypotheses, their representation, query modeling and retrieval. The different components are integrated closely. This allows for addressing query-by-example and query-by-string word spotting as well as two different decoding strategies in the same methodological framework. Semi-continuous HMMs are considered for this purpose. This makes the output model largely independent of the HMM states allowing for great flexibility with respect to model estimation, model decoding and caching. Figure 15 shows a schematic overview. The figure caption explains the relations between the components.

In order to retrieve document regions efficiently, it is important to cache representations that are independent of the query. For this purpose, text detection is performed on the document images. Based on *text detector scores*, *text hypotheses* are computed. Afterwards, text hypotheses are combined in order to define *line hypotheses*. For text detection, the proposed method does not rely on a single gray-level intensity binarization threshold. This makes the application under different document image conditions more robust.

For each line hypothesis, a sequence of BoF vectors is extracted. Different mixture models are considered for modeling BoF sequences as outputs of the HMM probabilistically. Since the mixture-component posterior-probabilities are independent of the query, they are stored in a *mixture component index*.

The key idea for addressing both query-by-example and query-by-string in a unified word spotting framework, is to obtain a *query word HMM* in both scenarios. Afterwards, retrieval is always performed in the same manner. For query-by-string, *character HMMs* are estimated based on word- or line-level annotations. For query-by-example, the query word model is estimated from a single exemplary word image. In both cases the output model is not adapted in the *semi-continuous* (SC) HMM estimation. This makes the estimation in the query-by-example scenario feasible. Furthermore, query size estimates are required for patch-based decoding. While the patch size is given by the exemplary word image for query-by-example, character sizes are estimated based on the training annotations for query-by-string.

Retrieval is performed in two different decoding stages. This allows for a trade-off between efficiency and accuracy. In the first decoding stage, a coarse search is based on probabilistic *mixture component voting*. Mixture components with non-zero probability in the query word model vote for cells in the document images in which the same mixture components have non-zero probability. The voting mass is the joint probability for the mixture component in the query word model and the mixture component in the document image cell. By following

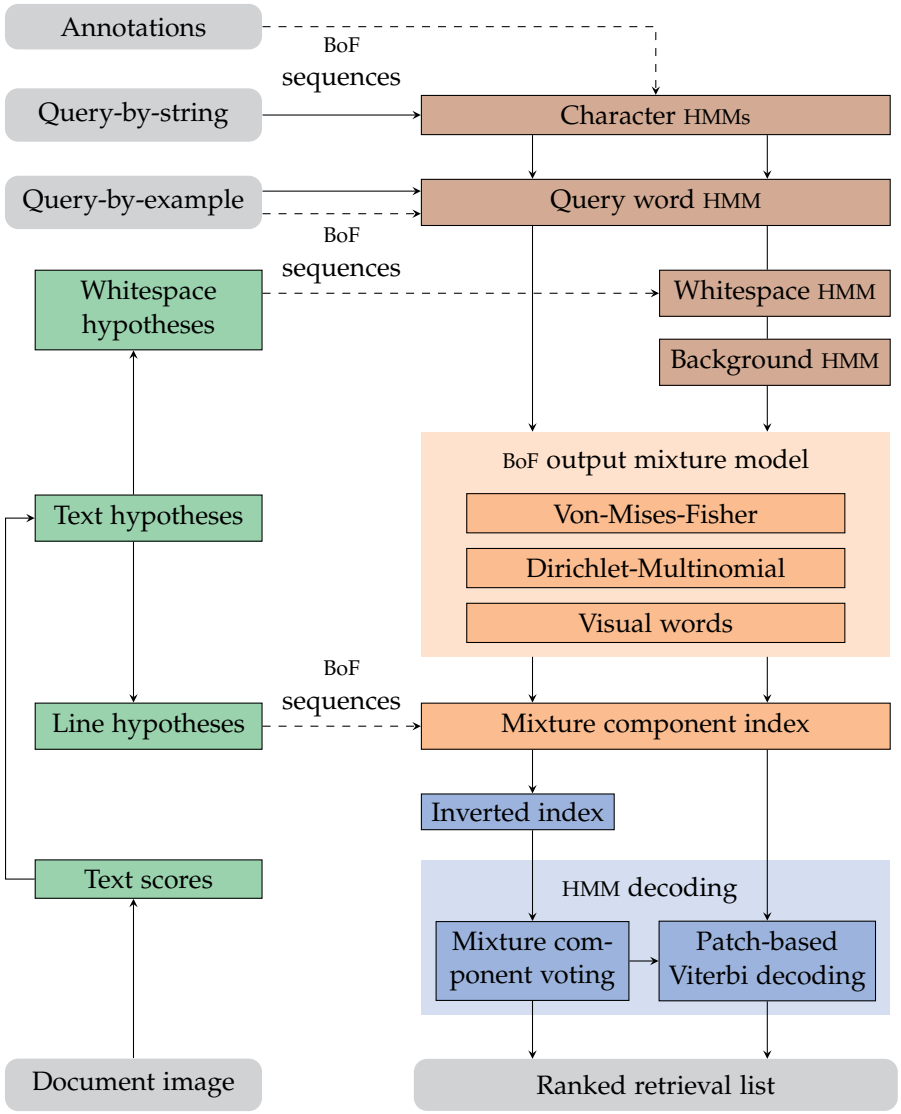


Figure 15: Word spotting system architecture. The figure shows a schematic visualization. The components are categorized into *input and output* (gray), *region hypotheses* (green), *HMM modeling* (brown), *BoF–HMM integration* (orange) and *HMM-based retrieval* (blue), cf. Figure 2. Arrows indicate the processing direction. The left side of the figure shows the inputs and document region hypotheses. The right side of the figure visualizes the output, query modeling as well as query model decoding. Two decoding stages can be identified. For this purpose, two vertical processing paths are shown by arrows that go from HMM modeling nodes (brown) to HMM decoding nodes (blue) to the output (gray). The results from the first decoding stage can be used in the second decoding stage. The node arrangement indicates which components are shared between the stages and which components are specific to one of the stages. BoF sequence extraction is shown with dashed lines. Dashed lines connect nodes that define document regions (on the left) with nodes that use BoF sequences (on the right). Document regions are defined by hypotheses as well as user inputs, e.g., word-level annotations or a query word image.

an Hough-style voting approach, the HMM state structure is considered. Relevant cells in the document images can rapidly be obtained through an *inverted* mixture component index. Based on the accumulated voting mass in each cell, potentially relevant document image patches are retrieved. Thus, the approach performs a coarse analysis that allows for fast detection based on approximate similarity scores.

The second decoding stage performs a fine analysis of the patches that have been identified in the first stage. For this purpose, a patch is represented with a compound query HMM. The compound HMM consists of a *background* model and *whitespace* models as well as the *query word* model. The whitespace models represent the left-side and right-side context of the query word within a patch. The background model represents arbitrary document image contents.

Using the *Viterbi algorithm*, similarity scores are based on the *optimal output probability* for generating the BoF sequences of the corresponding *patches* with the query model. Given the optimal alignment of a BoF vector sequence with the compound query HMM, the similarity score for a patch is obtained as the length-normalized *partial* output probability for the *query word* model. It is important to note that this does *not* correspond to a query posterior approximation. The scores represent the probabilities for the observations along a specific path in the query word HMM. Therefore, the challenge of estimating a high-quality filler model can be avoided. The background model and whitespace models are important in order to allow for coping with word size variabilities in the patch-based framework. The alignment of the BoF sequence with the models of the compound HMM allows for a detailed localization of the query word within a patch.

Whitespace model estimation requires whitespace region hypotheses. *Whitespace hypotheses* are obtained from text hypotheses in a bottom-up manner in the query-by-example scenario. In the query-by-string scenario, whitespace regions are obtained in analogy but based on training annotations. Finally, the spotted locations can be refined even further by taking text hypotheses into account.

The close integration of the two decoding stages allows for a very good trade-off because the models complement each other with respect to efficiency and accuracy. By sharing the feature representation and considering the same query model as well as its structure, the first stage achieves high recall at high speed while the second stage computes accurate rankings for selected document image regions.

4.2 DOCUMENT IMAGE REGIONS

Document regions are used in order to define the search context for spotting words in document images, estimate whitespace HMMs and to refine the localization of spotted words. Accordingly, three types of region hypotheses can be distinguished. Text hypotheses are derived

from text detector scores and represent mostly text components (Section 4.2.1). Line hypotheses are based on text hypotheses and represent possible line regions in document images (Section 4.2.2). Whitespace hypotheses are obtained with a voting scheme that is based on text hypotheses. They mostly represent document-background regions that are located to the left and to the right of words (Section 4.2.3).

4.2.1 *Text hypotheses*

Text hypotheses represent text components in document images and are the basis for all region-based operations. An important requirement for text hypotheses is that for each word and each of its bounds (left, right, upper, lower), there exists at least one text hypothesis with a similar bound.

It is important to note, that detecting all word bounds accurately is hard to achieve for handwritten document images and for historic handwritten document images in particular. These documents contain hundreds of words and, consequently, hundreds of hypotheses are required. Furthermore, the hypotheses have to be highly accurate. The size of words is usually small in comparison to the size of document images. Thus, a mismatch that is marginal with respect to the size of a document image is substantial with respect to the size of a word. This is a fundamental difference to object detection in natural scene images where typically fewer and larger objects are considered, e.g., [EEV+15]. Furthermore, the task of text detection in natural scene images is not comparable to the detection of words in a document image. In natural scene images the challenge is rather to differentiate text and background and not the differentiation of words in close proximity that might be touching each other, e.g., [NM16]. For this reason, object detection approaches from computer vision are mostly unsuitable. For example, a region proposal network alone has not been sufficient for word hypotheses generation [WLB17]. Word region proposals had to be augmented with dilated text proposals [WB15] in order to achieve high recall.

The proposed method takes the above mentioned challenges into account without requiring any annotated training material. For this purpose, the *maximally stable extremal region* (MSER) method [MCU+04] is adapted, cf. Section 2.1. The most important difference of the proposed method in comparison to MSER is that the ER tree is not based on thresholding image intensities but text detector score values. Text detector scores are expected to change smoothly. Therefore, the number of extracted regions can mainly be controlled by the number of thresholds. In contrast, MSER uses all possible intensity values as thresholds. The number of regions is controlled heuristically by selecting only the *maximally stable* ERs.

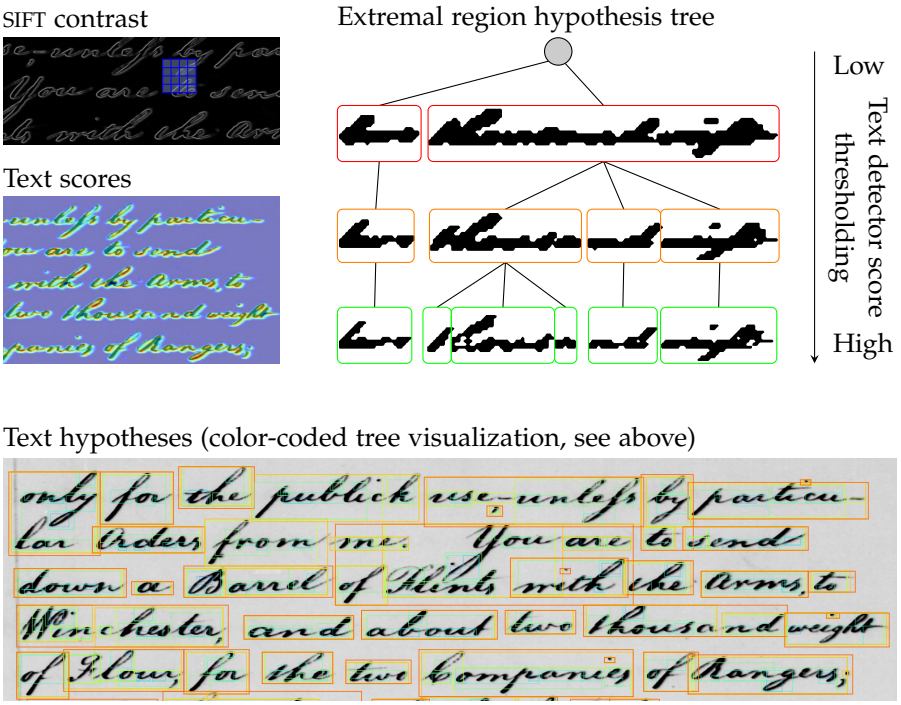


Figure 16: Text hypotheses generation. *Text detector scores* are computed with *SIFT contrast scores* in a dense grid. This is indicated by a single SIFT descriptor (blue) shown in an edge image of a document section. The resulting *text score* map is shown in blue to red colors. Blue indicates low contrast and red indicates high contrast in the local image neighborhood. *Text hypotheses* are obtained from an *ER tree* by thresholding text scores at different levels. Three levels, from the tree leaves towards the tree root, are exemplarily indicated with green to red colors. Text hypotheses typically cover parts of characters, parts of words, words, groups of words and also background clutter.

The main assumption is that text areas in document images have higher contrast than background areas. Consequently, text areas can be detected by measuring contrast. In order to be robust with respect to noise and touching ascenders and descenders in adjacent text lines, contrast is measured in a local image neighborhood, see Figure 16. This leads to smooth contrast variations between words, even if words are touching. Contrast scores are based on accumulated gradient magnitudes. Technically, they are computed as SIFT contrast normalization scores that are obtained for SIFT descriptors in a dense grid. The grid resolution equals to the resolution of the dense descriptor grid used for extracting BoF representations (cf. Section 4.3).

Text hypotheses are obtained from the ER tree according to the following conditions. All ERs are considered as potential text regions except for tree leaves and regions with implausible heights (see below). The number of thresholds controls the number of regions in the ER tree and, therefore, the resolution at which local text detector

score changes are detected. Threshold values are evenly spaced over the interval of minimum and maximum text score values. Figure 16 visualizes the concept. The color-coded text hypotheses indicate the structure of the tree for a larger document image section. Tree leaves are discarded because they can be considered as clutter. Thus, hypotheses are created for all ERs with child nodes. This can be seen as a heuristic for *minimal stability*. Implausible region heights are filtered based on lower and upper quantiles, i.e., 0.1 and 0.999, of a Weibull distribution, cf. [FEH+11, Chap. 46]. The distribution is estimated such that all potential text region heights are generated with maximum likelihood. The use of a Weibull distribution is motivated by its successful application to outlier detection, cf. e.g., [SRM+11]. Filtering helps to suppress text hypotheses that represent background clutter. The quantile values of 0.1 and 0.999 are chosen in order to define an interval that is sufficiently large such that most of the text components are represented by text hypotheses. The proposed word spotting method is robust against background clutter to a large extent. Similarly, the number of thresholds in the ER tree is a meta parameter that has to be high enough in order to capture text in low-contrast document image regions, see Section 5.3.2.

4.2.2 *Line hypotheses*

Line hypotheses are document regions that are bounded by the document in horizontal direction and that are bounded by text components in vertical direction. Line hypotheses guide the patch-based decoding process in order to analyze document image regions that contain text with a height that is similar to the height of the decoding patch. Document region representations can be precomputed because line hypotheses are independent of a particular query.

The line hypotheses generation process is based on text hypotheses. Text hypotheses mostly represent parts-of-characters, characters and groups-of-characters but also background clutter. In order to obtain an accurate line hypothesis for each word, it is necessary to consider combinations of text hypotheses. In contrast to the generation of word hypotheses, combinations only have to be considered in a single dimension, i.e., in vertical direction. Figure 17 shows an overview of the hypothesis generation process.

For each text hypothesis a set of line hypotheses is generated. For this purpose, the active text hypothesis defines a search context in the document image. Horizontally, the search context includes the entire document width. Vertically, the search context encloses the upper and lower bound of the active text hypothesis. While the upper bound of all line hypotheses, generated in the current set, is the upper bound of the search context, the lower bounds are given by the lower bounds of all text hypotheses within the search context. The search

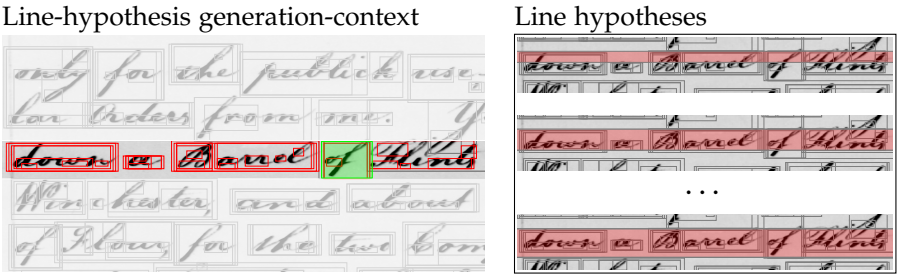


Figure 17: Line hypotheses generation. A set of line hypotheses is generated for each text hypothesis. In the figure, an exemplary text hypothesis is shown in green. Its upper and lower bounds define the *line-hypothesis generation-context*, as indicated by the image area that is *not* grayed out. The upper bound of the text hypothesis defines the upper bound of all line hypotheses that are generated within this context. The lower bounds are defined by the lower bounds of all text hypotheses within the context, including the current text hypothesis. A few *line hypotheses* are shown with red overlays on the right.

context could be increased in order to generate line hypotheses that are higher than the active text component. However, this is not required assuming there exists at least one text hypothesis spanning over the entire height of the largest word in the line. After processing all text hypotheses, line hypotheses are grouped according to line height. Line hypothesis positions and heights are quantized according to the dense descriptor grid coordinates (cf. Section 4.3).

In order to be robust with respect to word height variability, line positions for each height are augmented with positions of similar line heights. Line heights are considered as similar to a given height h if they are in the interval $[0.5h, 2h]$. In order to analyze the local neighborhood of potentially relevant text components in the patch-based decoding framework, line hypothesis positions are extended in the local neighborhood with a morphological filter. For this purpose, a one-dimensional binary mask is representing the current line position configuration for each line height. Additional hypotheses are generated based on a one-dimensional binary dilation operation, cf. [GW02, Sec. 9.2.1], that is applied to each binary line position mask. The size of the structuring element is given by the (rounded up) quotient of the line height and the vertical patch sampling step. According to the patch sampling step, a larger patch evaluation context will be considered for higher lines than for smaller lines.

4.2.3 *Whitespace hypotheses*

Whitespace hypotheses represent document background regions that are mostly located to the left and to the right of words. They are required in order to estimate whitespace HMMs. Whitespace HMMs

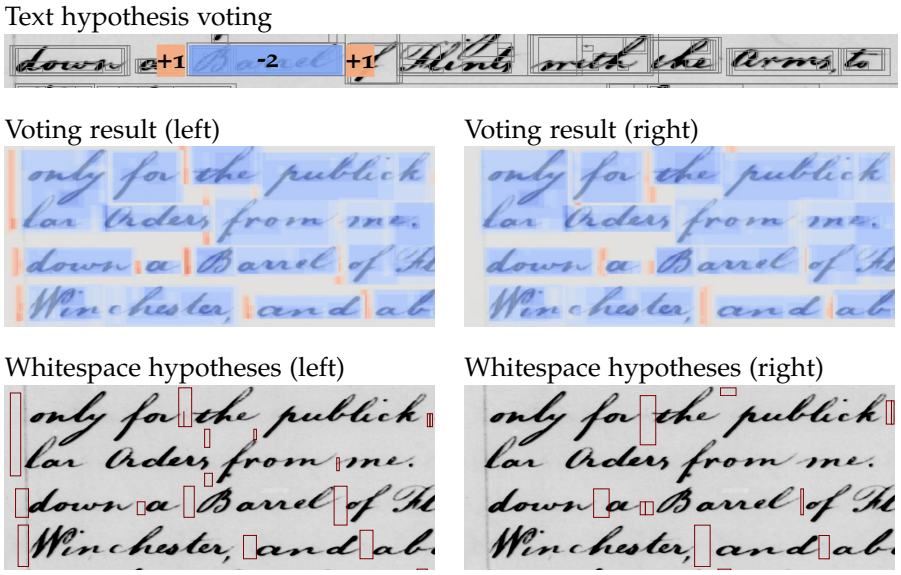


Figure 18: Whitespace hypotheses are generated by a voting scheme. Each text hypothesis *votes for* the document image regions to the left and to the right of its bounding box (+1). Furthermore, it *votes against* its bounding box region (-2). Votes are accumulated per pixel. In the figure, these regions are indicated in orange and blue. *Voting results* are obtained separately for whitespace regions oriented to the left and to the right. The accumulator values can be negative (blue colors), zero (gray) and positive (orange colors). *Whitespace hypotheses* are represented as region bounding boxes. They are obtained after thresholding at zero followed by a connected component analysis of the binary scores.

are used in the patch-based Viterbi decoding framework and allow for detecting words more accurately. Better similarity scores can be obtained for the patches due to the more detailed modelling.

For the query-by-example scenario, only a single annotated example and no other annotated training material is available. In this scenario, whitespace hypotheses are obtained with a voting scheme that is based on text hypotheses. Alternatively, the voting scheme can also be applied with word-level bounding box annotations if these are available. Figure 18 shows an overview of the process. It is important to note that whitespace hypotheses should only represent the *immediate* context of words and that it is *not* required to detect all relevant whitespace occurrences in the documents.

In order to obtain whitespace hypotheses from text components in a bottom-up manner, a procedure is required that is robust with respect to the text hypotheses. Text hypotheses mostly represent text on different levels, i.e., parts-of-character, parts-of-words or even parts-of-multiple-words. For this purpose, an important assumption is that the document regions to the left and to the right of text hypotheses are more likely to contain whitespace than the inner area of the text hypothesis regions. This is modeled in a per-pixel voting scheme. An

accumulator matrix is initialized with zeros and each matrix element corresponds to a pixel in the document image. Afterwards, each text hypothesis votes *against* its inner bounding box area and *for* the document image regions to the left and to the right. Inner regions are down-voted by -2 and potential whitespace regions are up-voted by +1. The whitespace voting area is defined such that the local image descriptors, used for representing the image regions, overlap at most 0% and up to 25% with the corresponding text hypothesis. After all text hypotheses have voted, document regions with positive accumulator values are considered as whitespace hypotheses. Bounding boxes are obtained after thresholding the accumulator matrix at zero and performing a connected component analysis.

The choice of the voting weights is heuristic. Since words are typically represented by multiple text hypotheses, the area to the left and to the right of a text hypotheses does not necessarily correspond to whitespace. With the proposed procedure, it requires two positive votes in order to compensate for a negative vote. Thus, false positives are avoided at the cost of missing a substantial number of whitespace regions. Due to the overall high number of true positives, this is not a limitation for whitespace HMM estimation, see Section 5.3.7.

Finally, it has to be noted that left-side and right-side whitespace hypotheses are estimated separately. This allows for obtaining more specific query models. Within Viterbi alignment, query-word-starting and query-word-ending positions can be decoded more accurately. Generally, this assumption is motivated by related HMM-based word spotting methods. It was shown that word spotting performance improves with the specificity of the filler model that is used for modeling the context of a query word in the text line [PTV15a].

4.3 DOCUMENT REGION REPRESENTATION

Document image regions are represented with sequences of BoF vectors. Regions are either given by manual bounding box annotations, e.g., in the query-by-example scenario, or regions are based on hypotheses, i.e., whitespace hypotheses and line hypotheses.

BoF have shown excellent performance in word spotting applications, e.g., [ART+15]. By extracting BoF sequences in writing direction, the sequential characteristic of text is preserved. An important design choice regards the local image descriptor in the BoF. For word spotting, the SIFT descriptor [Low04] has become the de-facto standard. Even though this descriptor is not specifically designed for document images, it captures discriminative properties of the pen stroke due to its use of gradient histograms. Furthermore, the descriptor can easily be replaced if a more suitable representation is available. In any case, BoF are adapted to the problem domain since they are based upon visual words, i.e., prototypical descriptors.

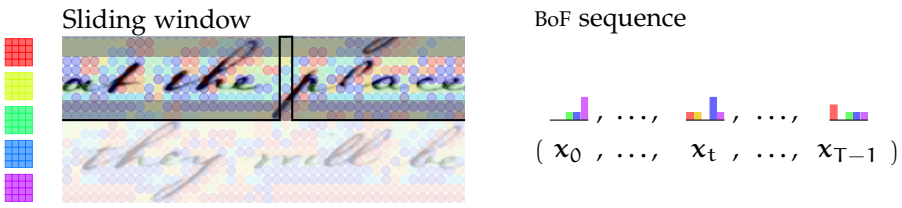


Figure 19: Based on a dense grid of quantized image descriptors, cf. Figure 5, a sequence of BoF representations is generated for a document region. In the figure, a section of a line hypothesis is indicated for this purpose. Visual words from the visual vocabulary are shown on the left. Within the document image, descriptor center points are shown as colored dots. The color indicates the quantization result. In this example, the grid sampling step is larger than in practice for a better visualization. Given the document region, the sequence of BoF representations is obtained by *sliding a window* over the grid columns in writing direction. A BoF is created at each window position. Visual words in the gray shaded areas are omitted because the corresponding descriptors overlap with the upper and lower region bounds. The sequence of BoF histograms is shown on the right. Histogram-bar colors correspond to the visual-word colors.

Irrespective of the local image descriptor, BoF are histogram representations. This is an advantage for patch-based decoding since BoF change smoothly for overlapping image regions. The effect is most important for patches that overlap vertically. Horizontal overlap is modeled within different HMM states. Smoothly changing representations allow for better localizations [RKE16].

Figure 19 shows an overview of the BoF sequence generation process for a document region. Document images are represented with a dense grid of local image descriptors. Descriptors are localized by their center points and the descriptor vectors are quantized with respect to a visual vocabulary. The descriptor size and orientation are uniform. Rotation invariance, cf. [Low04], is not a desirable property since the pen-stroke orientation is an important discriminative characteristic. For a region, only those descriptors are considered that do not overlap with the upper and lower region bounds. Thus, *valid* descriptors only represent the document region and not its upper and lower context. A minimum number of descriptors is assured by a lower threshold. The sequence of BoF vectors is obtained by sliding a window over the document region in writing direction. The window is moved over all grid columns such that it covers exactly one column at each position. BoF histograms are obtained from *valid* visual words.

The BoF representation at index t in a sequence is denoted as vector $\mathbf{x}_t$. The frequency of the visual word with index $v \in \{0, \ldots, V-1\}$ in $\mathbf{x}_t$ is a scalar x_{tv}, i.e., $\mathbf{x}_t = (x_{t0}, \ldots, x_{t,V-1})^\top$. Given a document region, the dense grid of valid visual-word indices within the region bounds is defined by the matrix $[u_{nt}]$ with $u_{nt} \in \{0, \ldots, V-1\}$. The in-

dices $(n, t) \in \{0, \dots, N-1\} \times \{0, \dots, T-1\}$ refer to rows and columns. BoF vector $\mathbf{x}_t \in \mathbb{N}_{\geqslant 0}^{V}$ can then be defined in terms of its vector components in Equation 13. Function $\delta : \{0, \dots, V-1\} \times \{0, \dots, V-1\} \to \{0, 1\}$ is used for counting visual words.

$$x_{tv} = \sum_{n=0}^{N-1} \delta(u_{nt}, v) \quad \text{with} \quad \delta(u_{nt}, v) = \begin{cases} 1 & : u_{nt} = v \\ 0 & : u_{nt} \neq v \end{cases} \tag{13}$$

Technically, BoF representations are very high dimensional and extremely sparse. Sparse vector representations are used in order to store BoF sequences efficiently. It has to be noted that contrary to other BoF applications in word spotting, e.g., [ART+15], descriptors are not pruned based on document image contrast. This has the advantage that it is not required to handle special cases, like windows without visual words. On the other side, the number of descriptors is large due to the typically high resolution of the descriptor grid.

The large number of descriptors leads to a high computational effort if the visual vocabulary is computed with Lloyd's algorithm [Llo82]. Typically, this is addressed by clustering only a randomly selected subset of all descriptors, cf. e.g., [ART+15]. However, in large document collections only a small fraction of the total number of descriptors is considered this way. In order to increase the reproducibility of the results, a strategy inspired by the *k-means++* initialization [AV07] is followed. Initial centroids are computed with Lloyd's algorithm, which is applied on 2DV randomly sampled descriptors. D denotes the descriptor dimensionality and 2 is a heuristic factor that ensures a sufficient number of samples. Afterwards, MacQueen's algorithm [Mac67] clusters the entire set of descriptors. The algorithm is suitable for this purpose because it iterates over the sample set only once and updates the codebook with every sample. It converges to an optimal codebook if the number of samples approaches infinity and subsequent samples are statistically independent, cf. [Fin14, p. 62]. After shuffling the descriptors, the algorithm offers a trade-off between accuracy and efficiency in the given scenario.

The most important meta parameters are the dense grid sampling step, the size of the descriptors and the size of the visual vocabulary, see Section 5.3.3. The grid sampling step has a considerable effect on accuracy and efficiency. The descriptor size largely controls the generalization capabilities of the BoF representation.

4.4 BAG-OF-FEATURES OUTPUT MODELS

BoF sequences are modeled as observations in the statistical HMM process with a probabilistic mixture model. The mixture model is required for the SC-HMM integration. Due to the special characteristics of BoF representations in this word spotting scenario, modeling

BoF with a GMM directly is infeasible. BoF vectors are very high dimensional and extremely sparse since they are extracted from line hypotheses. Thus, the robust estimation of GMM parameters is only possible after dimensionality reduction, cf. [Fin14, Sec. 9.1]. Unfortunately, it can be expected that dimensionality reduction results in a sub-optimal solution. This is due to the large gap between the typical BoF dimensionality (> 1000) and the typical feature vector dimensionality for GMM-HMM integrations (< 100). Empirically, this has been confirmed for handwritten word recognition with SC-HMMs [RVF12]. In order to estimate the shared GMM, BoF representations have been reduced from 2000 to 30 dimensions with principle component analysis, cf. [DHS00, Sec. 3.8.1]. The approach has been outperformed by a direct BoF-HMM integration (cf. Section 4.4.3). A similar result has been obtained for segmentation-free query-by-example word spotting with spatial pyramid matching [RAT+15a]. Dimensionality reduction is performed with *latent semantic indexing* in order to improve the generalization capabilities of the patch descriptors. However, a reduction to 64 dimensions produces results that are considerably worse compared to the original descriptor. The effect can be observed for vocabulary sizes of 2048 and above. It has to be noted that the dimensionality of the descriptors is three times larger than the size of the visual vocabulary due to the spatial pyramid configuration.

Mixture distributions that allow for modeling BoF directly, will be discussed in the following (Section 4.4.1 to 4.4.3). The characteristics and assumptions of the models will be compared and put in perspective with the requirements in the given scenario. For this purpose, a mixture model $\Theta = \{(c_k, \Theta_k) \,|\, 0 \leqslant k < M\}$ is defined by M mixture components with parameters Θ_k and their mixture weights c_k. The weights are prior component probabilities $c_k = p(\mathcal{M} = k \,|\, \Theta)$. M is a meta parameter, see Section 5.3.4, and $\mathcal{M}$ is a random variable that represents a discrete probability distribution over the event space $\Omega_{\mathcal{M}} = \{0, \ldots, M-1\}$. Since different distributions of the mixture components will be considered, component parameters Θ_k are defined along with the component models in the following sections. Therefore, the mixture model in Equation 14 is a generalization of the mixture model that has been defined in Equation 3.

$$p(\mathbf{x}_t \,|\, \Theta) = \sum_{k=0}^{M-1} p(\mathcal{M} = k \,|\, \Theta)\, p(\mathbf{x}_t \,|\, \mathcal{M} = k, \Theta) \tag{14}$$

Once the output model is chosen, it can be evaluated for all line hypotheses. This is due to the independence of the output model and the line hypotheses with respect to the query. The output model can be integrated with the HMM by indexing mixture component probabilities in a look-up table for each line hypothesis (Section 4.4.4).

Furthermore, the line hypotheses are used in order to obtain a training dataset for unsupervised estimation of the mixture models pre-

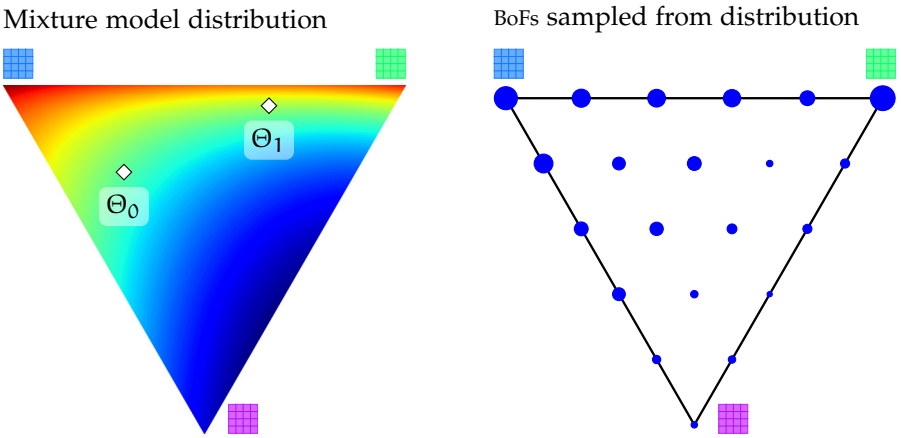


Figure 20: Visual-word simplex visualization. On the left, the figure shows a visualization of a statistical mixture distribution over the visual word simplex with blue to red colors. Blue indicates low and red indicates high probability. The white diamond markers indicate model parameters, i.e., the expected values of the mixture component distributions Θ_k with $k \in \{0, 1\}$. On the right, the figure shows BoF vectors that have been randomly sampled from the mixture distribution as points in the simplex. It has to be noted, that the sample distribution is discrete regardless of the visual word frequency scaling. BoF vectors in the simplex are indicated with blue dots. The relative amount of duplicates is indicated by the size of the dots. Statistical distributions that will be visualized in the following have been estimated with these samples.

sented in Section 4.4.1 and 4.4.2. For this purpose, BoF vectors are extracted from all line hypotheses on all document images of a document collection. The mixture model presented in Section 4.4.3 is directly based on visual words and does not require a model estimation step that is based on line hypotheses.

Different distributions will be discussed based on a three-dimensional toy example that resembles some of the characteristics of the original BoF representations. This refers mostly to their discrete characteristic and the presence of BoF vectors with visual word frequencies that are zero. The three-dimensional example can be visualized in a 2-simplex. Each of the vertices corresponds to a visual word. Figure 20 shows the source distribution and indicates a BoF sample set that has been randomly drawn from this distribution. In the following, all mixture models will be presented based on this toy example[1].

The source distribution is a mixture of *Dirichlet compound multinomial* (DCM) distributions and is theoretically suitable for modeling high-dimensional and sparse BoF vectors [Elk06]. This is due to the model's capability of distributing the probability mass on the surface of the simplex, i.e., edges in the three-dimensional example, see Figure 20. A BoF is only located within the simplex if all visual words

1 The multinomial mixture model in Figure 6 is based on the toy example, too.

have non-zero frequency. However, BoF that are high-dimensional and sparse have only few non-zero entries. In the example in Figure 20, BoF representations with non-zero frequencies for the blue and green visual word are most likely (jointly and exclusively). The pink visual word mostly occurs with high frequencies of the blue visual word. This can be observed for the distribution of probability mass and for the sampled BoF representations. Samples are drawn in three stages. In the first stage, a mixture component is chosen according to the components prior probabilities. In the last two stages the component is evaluated. A multinomial distribution is drawn from a Dirichlet distribution and the BoF is drawn from the multinomial distribution, cf. Appendix B. The concentration parameters of the Dirichlet distribution are chosen such that the probability mass of the DCM distribution is mostly distributed on the edges. Each BoF vector contains five visual words that are scaled to relative frequencies in order to be visualized in the simplex.

4.4.1 *Von Mises-Fisher*

The *von Mises-Fisher* (vMF) distribution allows for probabilistic modeling of directional data [BDG+05]. It is relevant for word spotting with BoF-HMMs due to the wide use of cosine similarity for matching BoF representations in the literature, cf. Section 3.3.1. The distribution models the generation of BoF vectors on the unit sphere, i.e., $\mathbf{x}_t \in \mathbb{R}^V$, $\|\mathbf{x}_t\|_2 = 1$, and has properties that are similar to a multivariate Gaussian distribution [BDG+05]. The vMF probability density function is defined by a mean direction $\boldsymbol{\mu} \in \mathbb{R}^V$, $\|\boldsymbol{\mu}\|_2 = 1$, concentration parameter $\kappa \in \mathbb{R}_{\geqslant 0}$ and for dimensionality $V \geqslant 2$, see Equation 15.

$$p(\mathbf{x}_t \mid \boldsymbol{\mu}, \kappa) = \frac{\kappa^{z-1}}{(2\pi)^z I_{z-1}(\kappa)} e^{\kappa \boldsymbol{\mu}^\top \mathbf{x}_t} \quad \text{with} \quad z = \frac{V}{2} \tag{15}$$

The concentration parameter controls by how much the density function focusses around $\boldsymbol{\mu}$. Similar to a Gaussian, Equation 15 consists of a normalization factor and an exponential term. Since $\|\boldsymbol{\mu}\|_2 = 1$ and $\|\mathbf{x}_t\|_2 = 1$, the exponent is the cosine similarity of $\boldsymbol{\mu}$ and $\mathbf{x}_t$ scaled by κ. The normalization factor includes $I_{z-1}(\cdot)$, i.e., the modified Bessel function of the first kind and order $z-1$. In order to improve numeric stability, the density is computed in the logarithmic domain. $I_{z-1}(\kappa)$ is approximated as suggested in [Elk06][2].

By using the vMF distribution as a model for the mixture components in Equation 14, a vMF mixture model is obtained. The model is estimated with an EM algorithm that is very similar to the EM algorithm used for estimating a GMM, cf. [Fin14, Sec. 4.4.2]. Given the number of mixture components, the model is initialized with a variant of Lloyd's algorithm. For this purpose, training samples are asso-

2 The approximation is based on a series expansion [OLB+10, Equ. 10.41.3].

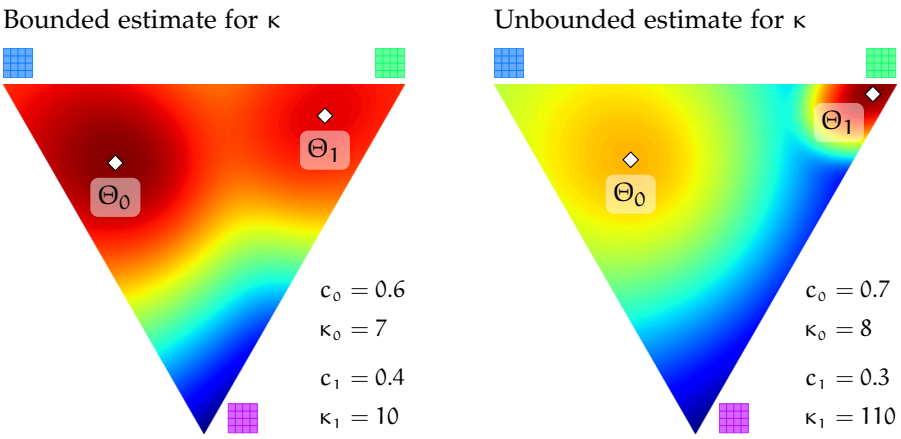


Figure 21: vMF mixture model visualization on the visual word simplex. The figure shows results for two different model parameter estimation strategies. On the left, concentration parameters are limited to an upper bound that has been set heuristically. On the right, concentration parameters have been estimated according to an approximate maximum likelihood criterion. In both visualizations, probability density is indicated with blue to red colors in the logarithmic domain. It has to be noted that the colors are scaled to the corresponding minimum and maximum values in both plots. Mixture model components are weighted by c_k and defined by a tuple $\Theta_k = (\mu_k, \kappa_k)$ where $k \in \{0, 1\}$. White diamond markers show mean parameter vectors μ_k that have been projected on the simplex. Mixture weights c_k and concentration parameters κ_k are specified with each of the plots. The values have been rounded for a qualitative interpretation. The κ bound has been set to 10.

ciated with centroids according to largest cosine similarity. Centroids are updated by the unit-length normalized sum of the samples that they have been associated with. The normalization ensures that the updated centroids lie on the unit sphere [BDG+05, Equ. 2.4]. A formal specification of the algorithm can be found in [BDG+05, Alg. 3]. Based on the hard assignments of centroids and samples, a vMF mixture component is initialized for every centroid. Mixture coefficients c_k are given by the relative number of samples that have been associated with the centroids. The mean directions μ_k are directly given by the centroids. The concentration parameters κ_k are estimated from the samples that have been associated with the centroids according to [BDG+05, Equ. 4.4].

Based on the initial parameters, the iterative model refinement is continued with soft assignments in the EM algorithm. Instead of associating each sample with a single centroid, each sample is associated with all mixture components based on posterior probabilities $p(\mathcal{M} = k | \mathbf{x}_t, \Theta)$ for all $k \in \Omega_\mathcal{M}$, cf. [BDG+05, Equ. 3.7] and also Equation 4. For this reason, hard assignments can been seen as a special case where a single component has probability one and the other components have probability zero. The estimation of the posterior prob-

abilities for the mixture components given a sample and the current model, is computed in the *expectation* step.

Based on the updated associations, the model parameters are updated within the *maximization* step. The updates are performed in analogy to the estimation of initial parameters (see above). The important difference is that all samples $\mathbf{x}_t$ will be considered for the update of a single mixture component Θ_k according to posterior probabilities $p(\mathcal{M} = k|\mathbf{x}_t, \Theta)$. The EM steps are iterated with each updated model. It can be shown that the procedure improves the model such that the probability of generating the sample set with the model improves or stays equal. The formal specification of the EM algorithm for vMF mixture models can be found in [BDG+05, Alg. 1]. It also includes the updates for the model parameters. It has to be noted that the estimation of the concentration parameters κ_k is not without problems. This is due to the maximum likelihood estimate for κ which includes the ratio of Bessel functions $I_z(\kappa)/I_{z-1}(\kappa)$ [BDG+05, Equ. 2.5]. Solving the estimate for κ analytically is impossible. For this reasons, different approximations for κ for high-dimensional data are discussed in [BDG+05]. However, these do not yield satisfactory results for the BoF vectors in the given scenario. This is because κ_k is always estimated to be substantially larger than the dimensionality of the BoF vectors which leads to poor generalization capabilities. The problem can be addressed by limiting κ_k to a maximum value that is set heuristically, i.e., with a meta parameter, see Table 8 in Section 5.3.4.

Figure 21 shows visualizations of vMF mixture model estimates based on the three-dimensional sample set presented in Figure 20. Even for the toy example, the unbounded κ estimates tend to overfit the data and are very sensitive to the random initialization of Lloyd's algorithm. A possible explanation is the discrete characteristic of the data where many samples occur multiple times. The vMF probability density function is continuous. Figure 21 shows that the vMF mixture component distributions are rotationally symmetric and differ in their location and concentration.

4.4.2 *Dirichlet compound multinomial*

Inspired from short text modeling, e.g., *bag-of-words* (BoW) of Twitter messages, *Dirichlet compound multinomial* (DCM) distributions can be considered in order to model sparse BoF vectors that are distributed on the surface of the simplex, cf. [MKE05; Elk06]. This is possible by using the conjugate prior of the multinomial distribution, i.e., the Dirichlet distribution [Bis06, Sec. 2.2.1]. The Dirichlet distribution can model the generation of multinomial parameters $\mathbf{p}$, thus, it is defined on the simplex. For this purpose, the Dirichlet distribution is parameterized with concentration parameters $\boldsymbol{\alpha} \in \mathbb{R}^V_{>0}$. In contrast to the multinomial distribution, cf. Section 2.4, the Dirichlet distribution al-

lows for distributing probability density on the edges of the simplex if $\alpha_v \ll 1$. However, since the Dirichlet distribution is continuous, it is not suitable for modeling discrete data directly [MKE05]. The generation of sparse BoF vectors can be modeled if the Dirichlet distribution and the multinomial distribution are used in a compound distribution as shown in Equation 16.

$$p(\mathbf{x}_t \mid \boldsymbol{\alpha}) = \int_{\mathbf{p}} p(\mathbf{x}_t \mid \mathbf{p})\, p(\mathbf{p} \mid \boldsymbol{\alpha}) \mathrm{d}\mathbf{p} \tag{16}$$

The Dirichlet probability density function $p(\mathbf{p} \mid \boldsymbol{\alpha})$ represents a distribution over probability distributions. In combination with the multinomial probability mass function $p(\mathbf{x}_t \mid \mathbf{p})$ this results in a distribution over parameters. By integrating over all visual word probability vectors $\mathbf{p}$, the Dirichlet distribution models plausible visual word configurations. In the generative process this can be understood as an urn model where the ball being drawn from the urn is not only returned but a second ball with the same color is added as well. Therefore, the visual word distribution sharpens with each visual word that is generated for the BoF vector. This is known as *Polya's urn model*, cf. [JKB97, Chap. 40].

In natural language processing, the DCM is used in order to model word burstiness, i.e., if a word appears once it is likely to appear again [MKE05]. Thus, the DCM rather models word occurrences than word frequencies.

When substituting the Dirichlet probability density function $p(\mathbf{p} \mid \boldsymbol{\alpha})$ and the multinomial probability mass function $p(\mathbf{x}_t \mid \mathbf{p})$ in Equation 16, the DCM probability mass function for $\mathbf{x}_t \in \mathbb{N}_{\geqslant 0}^V$ is obtained[3] in Equation 17. $\Gamma(\cdot)$ is the Gamma function, i.e., a generalization of the factorial function to real and complex values, cf. [OLB+10, Chap. 5].

$$p(\mathbf{x}_t \mid \boldsymbol{\alpha}) = \frac{\|\mathbf{x}_t\|_1!}{\prod_{v=0}^{V-1} x_{tv}!} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}_t\|_1)} \prod_{v=0}^{V-1} \frac{\Gamma(x_{tv} + \alpha_v)}{\Gamma(\alpha_v)} \tag{17}$$

In the context of a mixture model, $2MV$ evaluations of Gamma functions are required in the product. Since even a single evaluation of the Gamma function can be considered as computationally expensive, the application of the DCM is very time consuming if M and V are large [Elk06]. Furthermore, the product over visual words $v \in \Omega_v$ is mostly evaluated for visual word frequencies that are zero.

For this a reason, an approximation of the DCM is proposed in [Elk06] that is valid for sparse BoW representations. It is referred to as EDCM distribution because it belongs to the exponential family. Assuming that the sample vectors are sparse, probability mass is mostly distributed on the surface of the simplex. Consequently, the concentration parameters are small, i.e., mostly $\alpha_v \ll 1$. Based on this assumption, a linear approximation of the ratio of Gamma functions, cf.

3 The derivation of Equation 17 can be found in Section B.1.

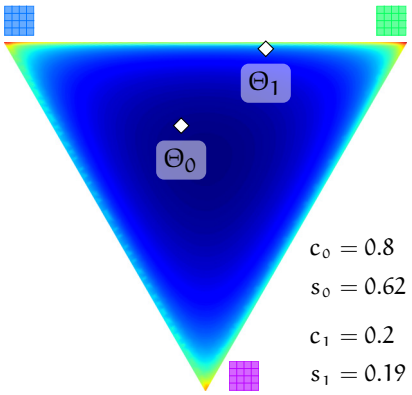


Figure 22: Simplex visualization of the EDCM mixture model. Probability mass is indicated with blue to red colors in the logarithmic domain. The colors have been scaled to the minimum and maximum probabilities. Diamond markers indicate visual word probabilities for the mixture components Θ_k with $k \in \{0, 1\}$. The visual word probabilities are given by the multinomial parameters which are represented by Dirichlet distributions in the EDCM mixture components. The multinomial parameters $\mathbf{p}_k$ are obtained as the expected values $\frac{\boldsymbol{\beta}_k}{s_k}$ of the Dirichlet distributions. $s_k = \|\boldsymbol{\beta}_k\|_1$ is referred to as the *precision* of an EDCM component distribution and measures its overall concentration, cf. Equation 19. EDCM component precision values s_k and mixture weights c_k are specified next to the simplex.

Equation 17, at $\alpha_v = 0$ is used. Equation 18 shows the approximation. The linear approximation is formally explained in Section B.2.

$$\frac{\Gamma(x_{tv} + \alpha_v)}{\Gamma(\alpha_v)} \approx \Gamma(x_{tv})\alpha_v \quad \text{for} \quad x_{tv} \geqslant 1,\ x_{tv} \in \mathbb{N}_{\geqslant 0} \tag{18}$$

For $x_{tv} = 0$ the ratio evaluates to one and does not influence the product in Equation 17. Thus, only the case $x_{tv} \geqslant 1$ has to be considered. Since $x_{tv} \in \mathbb{N}_{\geqslant 0}$ and $\Gamma(x_{tv}) = (x_{tv} - 1)!$, the factorial cancels out with the factorial in the denominator of the multinomial coefficient if the approximation in Equation 18 is substituted in Equation 17. The result is presented in Equation 19. A detailed derivation of Equation 19 can be found in Section B.2. In order to distinguish the EDCM from the DCM, the EDCM concentration parameters are referred to as $\boldsymbol{\beta}$.

$$p(\mathbf{x}_t \mid \boldsymbol{\beta}) = \frac{\|\mathbf{x}_t\|_1!}{\prod_{v:x_{tv} \geqslant 1} x_{tv}} \frac{\Gamma(\|\boldsymbol{\beta}\|_1)}{\Gamma(\|\boldsymbol{\beta}\|_1 + \|\mathbf{x}_t\|_1)} \prod_{v:x_{tv} \geqslant 1} \beta_v \tag{19}$$

For BoF vector sequences, Equation 19 can be computed efficiently by storing normalization factors in look-up tables. Due to the sparse BoF-vector characteristic, the look-up tables are typically small. All computations are performed in the logarithmic domain.

Figure 22 shows a visualization of an EDCM mixture model. Probability mass is distributed on the edges and vanishes towards the

center of the simplex. This is due to the linear approximation in Equation 18. Probabilities are underestimated for BoF vectors that are *not sparse*. The model's suitability for representing the samples shown in Figure 20 is, therefore, only limited.

The EDCM mixture model estimation closely follows the EM algorithm described in [Elk06]. The model is initialized by fitting a single EDCM distribution to the entire training dataset. M initial mixture components Θ_k are obtained by adding random noise to the concentration parameters, i.e., $\beta_{kv} \leftarrow \max(\beta_v + \mathcal{E}, \epsilon_{\text{low}})$ where $\mathcal{E}$ is random variable representing a uniform distribution over the interval $[-0.05, 0.05]$. The lower bound $\epsilon_{\text{low}} = 10^{-12}$ is required in order to ensure that none of the factors in Equation 19 becomes zero.

Based on an initial model, the posterior probabilities $p(\mathcal{M} = k \,|\, \mathbf{x}_t, \Theta)$ are computed for all $k \in \Omega_{\mathcal{M}}$ in the *expectation* step [Elk06, Equ. 7]. It has to be noted that a *deterministic annealing* procedure is applied in order to be less sensitive to local optima [UN98]. For this purpose, the likelihoods $p(\mathbf{x}_t \,|\, \mathcal{M} = k, \Theta)$ in the computation of the posteriors are smoothed with temperature meta-parameter $\tau \in \mathbb{N}_{\geqslant 1}$, cf. [Elk06].

$$p(\mathcal{M} = k \,|\, \mathbf{x}_t, \Theta) = \frac{p(\mathcal{M} = k \,|\, \Theta) \sqrt[\tau]{p(\mathbf{x}_t \,|\, \mathcal{M} = k, \Theta)}}{\sum_{l=0}^{M-1} p(\mathcal{M} = l \,|\, \Theta) \sqrt[\tau]{p(\mathbf{x}_t \,|\, \mathcal{M} = l, \Theta)}} \tag{20}$$

The root function changes the difference between low and high probabilities. For high values of τ, samples with smaller likelihoods are taken into account stronger than for smaller values of τ. Consequently, mixture components do not specialize on specific visual word configurations as much at high temperatures as at lower temperatures.

The update of the concentration parameters in the *maximization* step is based on the partial derivatives of the EDCM log-likelihood function, i.e., $\log p(\mathbf{x}_t \,|\, \boldsymbol{\beta})$, see Equation 19. The partial derivate for β_v is given in [Elk06, Equ. 5]. Together with the posteriors from the expectation step, the complete-data log-likelihood is obtained. Deriving the complete-data log-likelihood for model parameters Θ yields the parameter updates. Finally, it must be ensured that $\beta_{kv} > 0$ for all $k \in \Omega_{\mathcal{M}}$ and $v \in \Omega_v$. For this purpose, the components are smoothed by adding a small offset to all concentration parameters. The offset is dynamic and depends on the smallest non-zero concentration γ_k that has been estimated for each component [MKE05]. The meta parameter $\rho \in \mathbb{R}_{>0}$ with $\rho < 1$ adjusts the influence of the offset.

$$\beta_{kv} \leftarrow \beta_{kv} + \rho\,\gamma_k \quad \text{with} \quad \gamma_k = \min_{v \in \Omega_v} \{\beta_{kv} \,|\, \beta_{kv} > 0\} \tag{21}$$

The optimization is performed in a deterministic annealing procedure where the EM algorithm is run until the complete-data log-likelihood convergences for a given temperature. The process is repeated for each temperature in an annealing schedule. The model parameters are refined from temperature to temperature. The annealing schedule is defined heuristically, see Table 10 in Section 5.3.4. Typically, it starts with a high temperature and finishes with temperature $\tau = 1$ [Elk06].

4.4.3 *Visual words*

The output mixture models presented in in Section 4.4.1 and 4.4.2 model the generation of entire BoF vectors $\mathbf{x}_t$. The *visual words* model follows a different approach by modeling the BoF vectors in terms of *individual* visual word *occurrences*. For this purpose, the mixture model in Equation 14 is constrained to have exactly the same number of mixture components as visual words. Thus, mixture components directly refer to visual words. Component parameters Θ_v only encode the visual word index v. In analogy to random variable $\mathcal{M}$ that represents a distribution over mixture component indices, random variable $\mathcal{V}$ represents a distribution over visual word indices. For V different visual words, the corresponding event space is denoted by $\Omega_v = \{0, \ldots, V-1\}$. By replacing random variable $\mathcal{M}$ with random variable $\mathcal{V}$ in the mixture model in Equation 14, $p(\mathbf{x}_t \mid \Theta)$ can be expressed by marginalizing over visual words, see Equation 22 and 23.

$$p(\mathbf{x}_t \mid \Theta) = \sum_{v=0}^{V-1} p(\mathcal{V} = v \mid \Theta)\, p(\mathbf{x}_t \mid \mathcal{V} = v, \Theta) \tag{22}$$

$$= \sum_{v=0}^{V-1} p(\mathbf{x}_t, \mathcal{V} = v \mid \Theta) \tag{23}$$

In order to model *individual* visual word occurrences, the likelihoods for vector $\mathbf{x}_t$ must be replaced in Equation 22 and 23. For this purpose, $\mathbf{x}_t$ defines a distribution over visual words. Let $\boldsymbol{\mathcal{X}}_t$ be a multivariate random variable that represents independently distributed discrete random variables $(\mathcal{X}_{t0}, \ldots, \mathcal{X}_{t,V-1})^\top$ over the same event space $\Omega_x = \{0, \ldots, N\}$. The events correspond to *absolute* visual word frequencies x_{tv}, cf. Equation 13. Based on $\mathbf{x}_t$, $p(\mathcal{X}_{tv} > 0 \mid \mathbf{x}_t)$ is the probability that the absolute frequency of visual word v in BoF vector $\mathbf{x}_t$ is greater than zero. In Equation 24, the *logical disjunction* is considered in order to model the probability for *any* visual words that are occurring in $\mathbf{x}_t$ given the visual word probabilities that are represented by model Θ.

$$p\left(\bigvee_{\mathcal{X}_{tv} \in \boldsymbol{\mathcal{X}}_t} \mathcal{X}_{tv} > 0 \,\middle|\, \mathbf{x}_t, \Theta\right) = \sum_{v=0}^{V-1} p(\mathcal{V} = v \mid \Theta)\, p(\mathcal{X}_{tv} > 0 \mid \mathcal{V} = v, \mathbf{x}_t) \tag{24}$$

$$= \sum_{v=0}^{V-1} p(\mathcal{X}_{tv} > 0, \mathcal{V} = v \mid \mathbf{x}_t, \Theta) \tag{25}$$

Equation 24 is obtained in analogy to Equation 22 by noticing that $\mathcal{V}$ is statistically independent of BoF vector $\mathbf{x}_t$ and $\mathcal{X}_{tv}$ is statistically independent of model Θ. Consequently, the joint probability $p(\mathcal{X}_{tv} > 0, \mathcal{V} = v \mid \mathbf{x}_t, \Theta)$ can be interpreted as probability for visual word v in model Θ *and* the occurrence of visual word v in BoF vector $\mathbf{x}_t$. Due to

the marginalization, the joint probabilities over all visual words are taken into account.

Equation 25 is based on Equation 26 and 27 and follows from the *additivity* of the probabilities for mutually exclusive events, cf. e.g., [DHS00, Sec. A.4.1], here for any $k \in \Omega_v$ and any $l \in \Omega_v$ with $k \neq l$.

$$p(\mathcal{X}_{tk} > 0 \wedge \mathcal{X}_{tl} > 0, \mathcal{V} = k \wedge \mathcal{V} = l \,|\, \mathbf{x}_t, \Theta) = 0 \tag{26}$$

$$\begin{aligned} \Leftrightarrow\ & p(\mathcal{X}_{tk} > 0 \vee \mathcal{X}_{tl} > 0, \mathcal{V} = k \vee \mathcal{V} = l \,|\, \mathbf{x}_t, \Theta) = \\ & p(\mathcal{X}_{tk} > 0, \mathcal{V} = k \,|\, \mathbf{x}_t, \Theta) + p(\mathcal{X}_{tl} > 0, \mathcal{V} = l \,|\, \mathbf{x}_t, \Theta) \end{aligned} \tag{27}$$

For this purpose, Equation 26 formalizes that the occurrence of visual words k and l (with $k \neq l$) is mutually exclusive under the assumptions of the visual-word model mixture. Therefore, the occurrence probabilities for *any* of these visual words are additive as stated in Equation 27. In order to prove the premise in Equation 26, it is sufficient to show that $p(\mathcal{V} = k \wedge \mathcal{V} = l \,|\, \Theta) = 0$ with $k \neq l$, see Equation 28 to 31.

$$p(\mathcal{V} = k \wedge \mathcal{V} = l \,|\, \Theta) = 0 \tag{28}$$

$$\Leftrightarrow p(\{v \in \Omega_v \,|\, v = k\} \cap \{v \in \Omega_v \,|\, v = l\} \,|\, \Theta) = 0 \tag{29}$$

$$\Leftrightarrow p(\{k\} \cap \{l\} \,|\, \Theta) = 0 \tag{30}$$

$$\Leftrightarrow p(\emptyset \,|\, \Theta) = 0 \tag{31}$$

The evaluation of $p(\bigvee_{\mathcal{X}_{tv} \in \boldsymbol{x}_t} \mathcal{X}_{tv} > 0 \,|\, \mathbf{x}_t, \Theta)$ is based on decomposing the joint probability $p(\mathcal{X}_{tv} > 0, \mathcal{V} = v \,|\, \mathbf{x}_t, \Theta)$ in Equation 25 as shown in Equation 24. As a result, $p(\mathcal{V} = v \,|\, \Theta)$ are model parameters which are estimated from sample data. However, in contrast to the mixture models presented in Section 4.4.1 and 4.4.2, the component priors are not involved in the component parameter estimation. For the visual-word mixture model, components $p(\mathcal{X}_{tv} > 0 \,|\, \mathcal{V} = v, \mathbf{x}_t)$ are directly computed from BoF vectors $\mathbf{x}_t$. For this reason, $\mathcal{X}_{tv} > 0$ is conditioned on $\mathbf{x}_t$ but not on Θ. Therefore, the observed BoF vector $\mathbf{x}_t$ can be seen as parameterization of the discrete distributions that are represented by $\mathcal{X}_{tv} > 0$ for all $v \in \Omega_v$. In Equation 32, probability $p(\mathcal{X}_{tv} > 0 \,|\, \mathcal{V} = v, \mathbf{x}_t)$ is defined by the *relative* visual word frequency for visual word v in BoF vector $\mathbf{x}_t$.

$$p(\mathcal{X}_{tv} > 0 \,|\, \mathcal{V} = v, \mathbf{x}_t) = \frac{x_{tv}}{\|\mathbf{x}_t\|_1} \tag{32}$$

The use of relative frequencies follows from the *Laplace principle* where all events are assumed to be equally likely.

Equation 32 can be considered as a soft visual-word observation model. Instead of modeling a single observation (with probability one), the probability mass is distributed to all visual words that have been observed in frame t. The principle can be exemplified in Equation 24 if the probability vector

$$\Big(p(\mathcal{X}_{t0} > 0 \,|\, \mathcal{V} = 0, \mathbf{x}_t), \ldots, p(\mathcal{X}_{t,V-1} > 0 \,|\, \mathcal{V} = V-1, \mathbf{x}_t)\Big)^\top \tag{33}$$

just has a single non-zero entry, i.e., if it models the observation of a single visual word. In this case it simply *selects* the corresponding probability $p(\mathcal{V} = v|\Theta)$ from model Θ and Equation 24 represents *only* a discrete probability distribution. Due to the presence of multiple observations in frame t with probabilities $p(\mathcal{X}_{tv} > 0|\mathcal{V} = v, \mathbf{x}_t)$, the visual-word mixture model can be considered as *pseudo-discrete*.

Therefore, it is the soft observation model that extends the discrete model to a hierarchical model with two stages, i.e., the visual-word *mixture model*. The first stage generates a visual word index. The second stage generates the binary BoF vector component that corresponds to the visual word that was generated in the first stage, cf. Equation 24. In terms of urn models, this can be understood as having an urn for the visual words represented by model Θ and an urn for the visual words represented by BoF vector $\mathbf{x}_t$. Both urns are defined by proportions of visual words that are given by $p(\mathcal{V} = v|\Theta)$ and $p(\mathcal{X}_{tv} > 0|\mathcal{V} = v, \mathbf{x}_t)$, respectively. Thus, $p(\mathcal{X}_{tv} > 0, \mathcal{V} = v|\mathbf{x}_t, \Theta)$ is the probability for drawing a visual word with index v from the first urn and also drawing a visual word with the same index from the second urn. Hence, the first stage follows the urn scheme for a categorical distribution and the second stage follows the urn scheme for a Bernoulli distribution, cf. [Bis06, Sec. 2.1]. The categorical distribution can be seen as a special case of a multinomial distribution with a single observation, cf. [Bis06, Sec. 2.2]. Therefore, the generative process models the generation of a single visual word and not a *bag-of-visual-words*. Contrary to this model assumption, different visual words will typically be observed in a BoF vector. Thus, the visual-word mixture model does not represent a *conjunction* of visual words, as do multinomial models, but a *disjunction* of the visual words that have been observed in the bag-of-visual-words.

In summary, the visual-word mixture model makes the following *assumptions* that are not in accordance with the properties of BoF vectors extracted from the document regions:

- BoF can be represented by a disjunction of visual words (Equation 25).
- Visual words are mutually exclusive (Equation 26).
- Visual words are equally likely (Equation 32).

For word spotting, these assumptions lead to very good generalization capabilities. This is due to the abstraction from the actual BoF vectors. By interpreting BoF vectors as distributions, probabilistic similarity with respect to the model's visual word distribution is computed by cross-correlation. However, due to the strong abstraction, the approach lacks robustness in special cases. A degenerated case arises if visual words are non-discriminative. If non-discriminative visual words exist, these visual words occur in many frames and, therefore, also in the query model. Due to their mostly high probability in

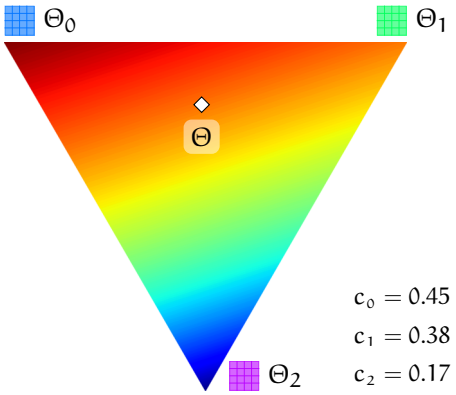


Figure 23: Simplex visualization of a visual-word mixture model. Simplex vertices are labeled with visual words. This is indicated by Θ_v where $v \in \{0, 1, 2\}$ is the visual word index. The Θ_v labels are shown next to the visual word descriptors that are colored in blue, green and pink. The labels emphasize that mixture components Θ_v correspond to visual words in the visual-word mixture model. Probability mass is indicated with blue to red colors in the logarithmic domain. The color range has been scaled to the minimum and maximum probability in the simplex. The visual word model Θ is indicated with a white diamond marker. The model's visual word probabilities $c_v = p(\mathcal{V} = v, \Theta)$ determine the marker position on the simplex. The rounded visual word probabilities are specified along with the plot. Labels Θ_v relate these probabilities with visual word vertices.

the visual word distributions, they tend to dominate the probabilistic similarity. Consequently, the model over-generalizes and looses its ability to discriminate. In document images, non-discriminative visual words typically lie in regions with uniform image intensities. A simple heuristic is to increase the descriptor size. Descriptors within annotated document image regions are less likely to be non-discriminative this way. For modeling BoW with multinomial models, the same problem is typically addressed with stop word filtering [BR11, Sec. 6.6]. However, due to the specific modeling of visual word configurations, multinomial approaches are more robust in this regard. This aspect is also interesting for weighting visual word frequencies in Equation 32. Instead of assuming an uniform visual word distribution, typical BoW weighting schemes can be applied, e.g., inverse document frequency [BR11, Sec. 3.2.4]. However, as BoF vectors are very sparse in this scenario, the visual word occurrence is more important than (weighted) visual word frequency, see Table 12 in Section 5.3.4.

Further, a degenerated case arises if the model's visual word distribution $p(\mathcal{V} = v|\Theta)$ is uniform. If all visual words are equally likely, the model looses its ability to discriminate any BoF representations. For word spotting, this case is, however, less relevant because query word models tend to be very specific.

Another important aspect concerns model estimation. The mixture components in the visual-word mixture model directly correspond to visual words. Therefore, only the visual vocabulary is required and an additional EM-style estimation as for the mixture models in Section 4.4.1 and 4.4.2 can be avoided.

Figure 23 shows a visualization of the visual-word mixture model that has been estimated from the three-dimensional example presented in Figure 20. The distribution is entirely defined by the model's visual word proportions, i.e., $c_v = p(\mathcal{V} = v \,|\, \Theta)$. The probability mass is distributed in form of a ramp that generalizes over specific visual word configurations. It models which visual words have generally been observed. The distribution reaches its extrema on the edges of the simplex.

4.4.4 *Hidden Markov model integration*

The HMM integration of the BoF output models that have been presented in Section 4.4.1 to 4.4.3 follows the standard approach for SC-HMMs, cf. [Fin14, Sec. 7.3]. Instead of using the mixture component likelihoods $p(\boldsymbol{x}_t \,|\, \mathcal{M}_t = k, \lambda)$ directly, cf. Equation 5, the likelihoods are replaced with *approximations* of mixture component posteriors $p(\mathcal{M}_t = k \,|\, \boldsymbol{x}_t, \lambda)$. Provided that HMM λ contains mixture component parameters Θ_k for all $k \in \Omega_{\mathcal{M}}$, the probability for generating BoF vector $\boldsymbol{x}_t$ in state j according to the *standard* mixture model integration is shown in Equation 34. Random variables $\mathcal{S}_t$ and $\mathcal{M}_t$ are specific to time t in the statistical HMM process. They represent distributions over the event spaces $\Omega_{\mathcal{S}}$ and $\Omega_{\mathcal{M}}$, respectively. The set $\Omega_{\mathcal{S}} = \{0, \ldots, S-1\}$ contains HMM state indices.

$$b_j(\boldsymbol{x}_t) = \sum_{k=0}^{M-1} p(\mathcal{M}_t = k \,|\, \mathcal{S}_t = j, \lambda)\, p(\boldsymbol{x}_t \,|\, \mathcal{M}_t = k, \lambda) \tag{34}$$

The component posterior *approximations* are obtained in Equation 36 by dividing the *likelihood* $p(\boldsymbol{x}_t \,|\, \mathcal{M}_t = k, \lambda)$ through the *evidence* $p(\boldsymbol{x}_t \,|\, \lambda)$, see Equation 35. The evidence is approximated by marginalizing over the mixture components of model Θ. This is an approximation because the mixture model represents the distribution of feature vectors with a finite number of mixture components, cf. [Fin14, p. 139]. Furthermore, the distribution of mixture component priors $p(\mathcal{M} = k \,|\, \Theta)$ is assumed to be uniform.

$$\frac{p(\boldsymbol{x}_t \,|\, \mathcal{M}_t = k, \lambda)}{p(\boldsymbol{x}_t \,|\, \lambda)} \approx \frac{p(\boldsymbol{x}_t \,|\, \mathcal{M}_t = k, \lambda)}{\sum_{l=0}^{M-1} p(\boldsymbol{x}_t \,|\, \mathcal{M}_t = l, \lambda)} \tag{35}$$

$$\approx p(\mathcal{M}_t = k \,|\, \boldsymbol{x}_t, \lambda) \tag{36}$$

The output probability for feature vector $\boldsymbol{x}_t$ in HMM state j that uses component posterior approximations is denoted as $b'_j(\boldsymbol{x}_t)$, see

Equation 37. Equation 37 is obtained by replacing the likelihoods $p(\mathbf{x}_t \,|\, \mathcal{M}_t = k, \lambda)$ in Equation 34 with the component posteriors from Equation 36. By using component posteriors, likelihoods are rescaled, thus, normalizing their dynamic range. This has advantages for multiplying many and potentially very small values [Fin14, p. 133]. Due to the different dynamic ranges of different output models, this normalization also simplifies the integration of the different BoF output models.

$$b'_j(\mathbf{x}_t) = \sum_{k=0}^{M-1} p(\mathcal{M}_t = k \,|\, \mathcal{S}_t = j, \lambda)\, p(\mathcal{M}_t = k \,|\, \mathbf{x}_t, \lambda) \tag{37}$$

In order to limit the number of non-zero component posteriors in Equation 37, a *beam pruning* strategy, cf. [Fin14, Sec. 10.2.1], is applied to the likelihoods in Equation 35. The basic idea is to define an interval $[zb, z]$ where z denotes the largest likelihood and $b \in [0, 1]$ is a meta parameter, the so-called *beam factor*. The interval is referred to as the beam. Likelihoods that are *not* falling into the beam are set to zero as shown in Equation 38. Typically, b is a very small number, e.g., $b = \exp(-14)$, where $\exp(\cdot)$ is the natural exponential function.

$$p(\mathbf{x}_t \,|\, \mathcal{M}_t = k, \lambda) \leftarrow \begin{cases} p(\mathbf{x}_t \,|\, \mathcal{M}_t = k, \lambda) & : p(\mathbf{x}_t \,|\, \mathcal{M}_t = k, \lambda) \geqslant zb \\ 0 & : p(\mathbf{x}_t \,|\, \mathcal{M}_t = k, \lambda) < zb \end{cases}$$
$$\text{with} \quad z = \max_{l \in \Omega_{\mathcal{M}}} p(\mathbf{x}_t \,|\, \mathcal{M}_t = l, \lambda) \tag{38}$$

It should be noted that beam pruning is applied to *all* likelihoods in the numerator *and* in the denominator of Equation 35. Beam pruning leads to sparse probabilistic representations that are important for efficient model decoding.

Furthermore, computing the sum of very small likelihoods in the denominator of Equation 36 is not without problems. In practice, the likelihoods are represented in the logarithmic domain. If the logarithmic values were simply transformed into the linear domain in order to evaluate the sum, the summation of very small values would cause numeric difficulties, cf. [Fin14, p. 135]. For this reason, all likelihoods are scaled with the inverse of the largest likelihood before they are transformed to the linear domain in Equation 39, cf. [Elk06, Equ. 7]. Since the operation is performed in the numerator and in the denominator, the scaling does not change the ratio. It has to be noted that the same procedure is applied during output model estimation (Section 4.4.1 and 4.4.2) for the computation of posterior probabilities in the expectation step, e.g., in Equation 20.

$$p(\mathcal{M}_t = k \,|\, \mathbf{x}_t, \lambda) \approx \frac{\exp(\log p(\mathbf{x}_t \,|\, \mathcal{M}_t = k, \lambda) - z)}{\sum_{l=0}^{M-1} \exp(\log p(\mathbf{x}_t \,|\, \mathcal{M}_t = l, \lambda) - z)}$$
$$\text{with} \quad z = \max_{l \in \Omega_{\mathcal{M}}} \log p(\mathbf{x}_t \,|\, \mathcal{M}_t = l, \lambda) \tag{39}$$

In case of the visual-word output model, cf. Section 4.4.3, the output probibility $b'_j(\mathbf{x}_t)$ is defined in terms of visual words $v \in \Omega_v$ with $\Omega_v = \{0, \ldots, V-1\}$ as shown in Equation 40. Component posteriors $p(\mathcal{M}_t = k|\mathbf{x}_t, \lambda)$ are replaced with visual word probabilities $p(\mathcal{V}_t = v|\mathbf{x}_t, \lambda)$ for this purpose. The random variable $\mathcal{V}_t$ is specific to time t and represents a distribution over the event space Ω_v. The state-dependent mixture weights directly correspond to state-dependent visual word probabilities.

$$b'_j(\mathbf{x}_t) = \sum_{v=0}^{V-1} p(\mathcal{V}_t = v|\mathcal{S}_t = j, \lambda)\, p(\mathcal{V}_t = v|\mathbf{x}_t, \lambda) \tag{40}$$

The visual word probabilities $p(\mathcal{V}_t = v|\mathbf{x}_t, \lambda)$ are defined in Equation 41. The random variable $\mathcal{X}_{tv}$ represents a distribution over absolute visual word frequencies. More specifically, $\mathcal{X}_{tv} > 0$ represents the *occurrence* of visual word v in BoF vector $\mathbf{x}_t$ at time t.

$$p(\mathcal{V}_t = v|\mathbf{x}_t, \lambda) = \frac{p(\mathcal{X}_{tv} > 0|\mathcal{V}_t = v, \mathbf{x}_t)}{\sum_{w=0}^{V-1} p(\mathcal{X}_{tw} > 0|\mathcal{V}_t = w, \mathbf{x}_t)} \tag{41}$$

$$= p(\mathcal{X}_{tv} > 0|\mathcal{V}_t = v, \mathbf{x}_t) = \frac{x_{tv}}{\|\mathbf{x}_t\|_1} \tag{42}$$

$p(\mathcal{X}_{tv} > 0|\mathcal{V}_t = v, \mathbf{x}_t)$ is equal to $p(\mathcal{V}_t = v|\mathbf{x}_t, \lambda)$ since the denominator in Equation 41 is equal to one. This is due to the definition of $p(\mathcal{X}_{tv} > 0|\mathcal{V}_t = v, \mathbf{x}_t)$ in Equation 42. Consequently, it is not required to improve the numeric stability as in Equation 36.

In the following, the BoF output model will generically be referred to in the terminology of Equation 37. Component posterior probabilities are denoted as $m_{tk} = p(\mathcal{M}_t = k|\mathbf{x}_t, \lambda)$ and state-dependent mixture component prior probabilities are denoted as $c_{jk} = p(\mathcal{M}_t = k|\mathcal{S}_t = j, \lambda)$. Mixture component posteriors will be used in two different scenarios. In the first scenario, posteriors m_{tk} are computed for specific document image regions in order to estimate HMM state-dependent mixture components c_{jk}. The number of BoF vectors within these regions will generally be referred to as T_q where q is an identifier for the region, thus $0 \leqslant t < T_q$. In the second scenario, mixture component posteriors are computed for line hypotheses in order to search occurrences of the query word. In this regard it is important to refer to a specific line hypothesis $l \in \Lambda$ where Λ is the set of line position indices that have been obtained for a line height on a document image. For this purpose, component posteriors are referred to as $m^{[l]}_{tk}$. BoF vectors extracted from line hypothesis l are referred to as $\mathbf{x}^{[l]}_t$. The number of BoF vectors is the same for all line hypotheses in a document and is referred to as T_l such that $0 \leqslant t < T_l$. In analogy to q, l refers to line hypotheses in a document. Equation 43 defines the output probability for BoF vector $\mathbf{x}^{[l]}_t$ in state j in terms of c_{jk} and $m^{[l]}_{tk}$.

$$b'_j(\mathbf{x}^{[l]}_t) = \sum_{k=0}^{M-1} c_{jk}\, m^{[l]}_{tk} \tag{43}$$

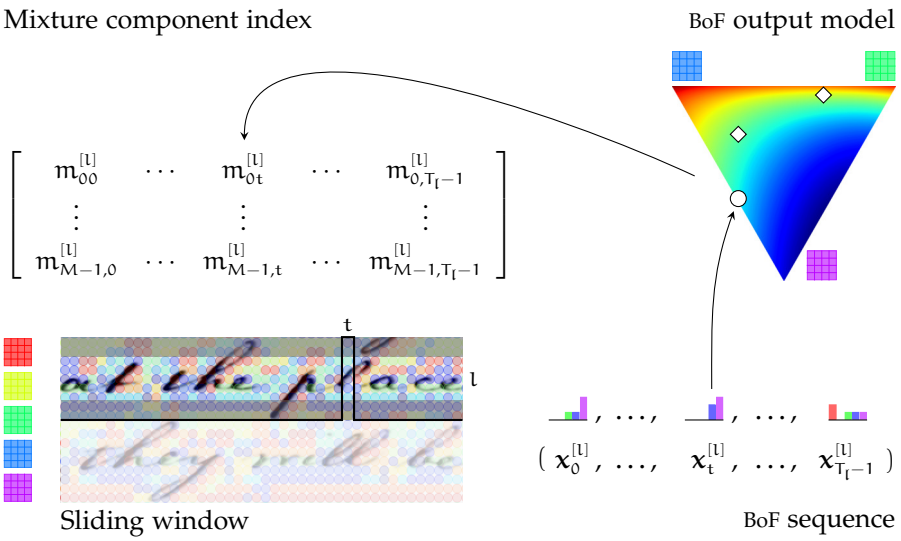


Figure 24: Mixture component indexing for decoding BoF vector sequences. The figure visualizes the process from BoF sequence extraction to BoF vector output modeling and indexing. For this purpose, a BoF output model is shown on the right. The diamond markers refer to the mixture components. The mixture component index is shown on the left. The BoF vector $\mathbf{x}_t^{[l]}$ from line hypothesis l is indicated as a white dot. It lies on the edge of the simplex due to the vector's sparsity. For HMM decoding, the BoF vector is represented by mixture component posteriors $(m_{t0}^{[l]}, \ldots, m_{t,M-1}^{[l]})^\top$. The relation between BoF vector $\mathbf{x}_t^{[l]}$, the BoF output model and the entry in the look-up table is emphasized by arrows. The columns in posterior probability matrix $[m_{kt}]$ mimic the sequential structure of corresponding BoF vectors in the line hypothesis.

For searching words efficiently, it is essential to avoid computations at query time. In the given scenario, the evaluation of the BoF output model can be precomputed for all line hypotheses which are relevant for a given query. This is a considerable advantage since the output model evaluation takes a large amount of time in the entire HMM decoding process [Fin14, Sec. 10.1]. This is particularly the case for the models presented in Section 4.4.1 and 4.4.2.

For all line hypotheses $l \in \Lambda$, look-up tables store component posteriors $\mathbf{m}_t^{[l]} = (m_{t0}^{[l]}, \ldots, m_{t,M-1}^{[l]})^\top$ with $m_{tk}^{[l]} > \epsilon_{\text{low}}$ for all $k \in \Omega_M$. This is sufficient since only posteriors $\mathbf{m}_t^{[l]}$ depend on the BoF vectors $\mathbf{x}_t^{[l]}$, cf. Equation 43. In order to limit the size of the look-up tables, sparse representations are used that only store sufficiently large posterior probabilities ($\epsilon_{\text{low}} = 10^{-12}$). While the probabilistic representations of the visual-word output model are intrinsically sparse, beam pruning is required in order to enforce sparsity otherwise. The concept of indexing component posteriors in a look-up table is visualized for a section of a line hypothesis in Figure 24. The figure combines results from the previous sections, i.e., a region hypothesis, the BoF sequence and the BoF output model.

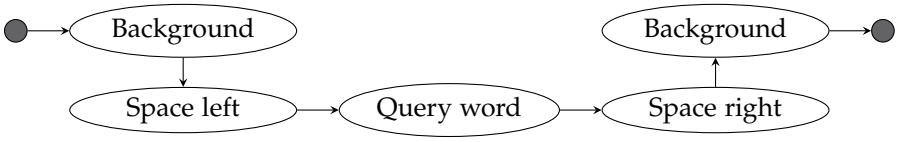


Figure 25: The query HMM is a compound HMM consisting of a model for the *query word* and the *context* in which the query word is expected to appear in the document image. Ellipses visualize the individual HMMs. The arrows specify the structure of the compound HMM. In this regard, gray dots indicate the start-node and the end-node.

4.5 QUERY MODELING

The query model is a compound HMM that consists of models for the *query word* and the query word *context*. The query word can be provided by-example (Section 4.5.2) or by-string (Section 4.5.3). The context consists of a *background* model and *whitespace* models (Section 4.5.4). The query word model and the whitespace models are estimated from annotated document regions (Section 4.5.1). The background model is based on the mixture component prior distribution.

Figure 25 shows a schematic visualization of the compound query HMM. The HMM is modeling expected occurrences of the query word within a patch in the patch-based framework. Patches that are *accurately* positioned over relevant words should receive higher similarity scores than patches that overlap but are not centered over relevant words. For this reason, the query word is enclosed by whitespace models that are specific to the left and right side context, respectively. In order to model arbitrary document image content that might be occurring at the patch boundaries to the left and to the right, the compound HMM contains a background model. It should be noted, that the model design differs from traditional approaches to line-based word spotting with HMMs where the context is optional, e.g., [FKF+12]. In these scenarios, the objective is to decode a single most likely occurrence of the query word in a segmented text line which might be located at the beginning or end of the line. Here, however, the search space is *not* constrained to a single line but it encompasses the entire document image. Similarity scores for several patches are taken into account in order to decode an occurrence of the query word.

4.5.1 *Estimation*

Model estimation follows the standard approach for HMMs with the Baum-Welch algorithm, cf. [Fin14, Sec. 5.7.4]. This excludes the output model which is independent of the query word. The output model is estimated in an unsupervised manner as described in Section 4.4.1 to 4.4.3. For estimating the query model, the corresponding HMM is

denoted as λ and has to be initialized. The initialization depends on the characteristics of the models, i.e., word models (Section 4.5.2), character models (Section 4.5.3) or context models (Section 4.5.4) and will be addressed in the corresponding sections.

Based on the initialized model, state-dependent model parameters are iteratively optimized. This includes transition probabilities and most importantly mixture weights for the mixture components of the output model. The state-independent parameters of the output model are not subject to optimization during query model estimation. Start probabilities are not modeled explicitly. The first state of any state sequence always corresponds to the background model due to the structure of the compound query HMM.

Given the BoF vector sequence $\mathbf{O} = (\mathbf{x}_0, \ldots, \mathbf{x}_{T_q-1})$ of a single annotated document image region, the BoF output model is evaluated in order to obtain mixture component posterior probabilities $m_{tk} > \epsilon_{\text{low}}$ for all $k \in \Omega_{\mathcal{M}}$ and for $0 \leqslant t < T_q$ according to Equation 39 or 42. Very small posteriors ($\epsilon_{\text{low}} = 10^{-12}$) are neglected in order to be consistent with the line region representations, cf. Section 4.4.4. Based on the current model λ, the updated model $\hat{\lambda}$ contains optimized state-dependent mixture-component prior-probabilities $\hat{c}_{jk} = p(\mathcal{M}_t = k \,|\, \mathcal{S}_t = j, \hat{\lambda})$.

$$\hat{c}_{jk} = \frac{\sum_{t=0}^{T_q-1} p(\mathcal{M}_t = k, \mathcal{S}_t = j \,|\, \mathbf{O}, \lambda)}{\sum_{t=0}^{T_q-1} p(\mathcal{S}_t = j \,|\, \mathbf{O}, \lambda)} \tag{44}$$

$$p(\mathcal{M}_t = k, \mathcal{S}_t = j \,|\, \mathbf{O}, \lambda) = \frac{\sum_{i=0}^{S-1} \alpha_{t-1}(i) a_{ij} c_{jk} m_{tk} \beta_t(j)}{p(\mathbf{O} \,|\, \lambda)} \tag{45}$$

Equation 44 shows the standard approach to estimating state-dependent mixture component priors [Fin14, Equ. 5.21]. In order to avoid zero-probabilities, the estimates are limited to $\epsilon_{\text{low}} = 10^{-12}$ such that $\hat{c}_{jk} \leftarrow \max(\hat{c}_{jk}, \epsilon_{\text{low}})$. For each state, the component priors are normalized in order to ensure that they constitute a probability distribution, i.e., $\sum_{k=0}^{M-1} c_{jk} = 1$ for all $j \in \Omega_{\mathcal{S}}$.

The estimation is part of the *maximization* step of the Baum-Welch algorithm, cf. [Fin14, Sec. 5.7.4]. It contains posterior probabilities $p(\mathcal{M}_t = k, \mathcal{S}_t = j \,|\, \mathbf{O}, \lambda)$ which are obtained for all $(j, k) \in \Omega_{\mathcal{S}} \times \Omega_{\mathcal{M}}$ and all BoF vectors of the training dataset in the *expectation* step. The posterior is defined in Equation 45 and represents the probability for selecting mixture component k in state j in order to generate feature vector $\mathbf{x}_t$. For this purpose, the generation is represented in the context of the total output probability such that the computation in the numerator of Equation 45 is restricted to component k and state j at time t, i.e., $c_{jk}\, m_{tk}$. The probability for transitioning from the preceding state i at time $t-1$ to state j at time t is denoted by $a_{ij} = p(\mathcal{S}_t = j \,|\, \mathcal{S}_{t-1} = i, \lambda)$ for all $i \in \Omega_{\mathcal{S}}$. The *forward variable* $\alpha_{t-1}(i)$ represents the total output probability until time $t-1$ ending in state i. The *backward variable* $\beta_t(j)$ represents the total output probability

from time $t+1$ starting in state j at time t. The posterior is obtained by dividing through the total output probability for generating the *entire* feature vector sequence in the denominator.

The transition probabilities are independent of the output model. Therefore, the standard estimates for discrete HMMs can be used, cf. [Fin14, Equ. 5.17]. In case the HMM is estimated from more than a single document region, the parameter estimation statistics, e.g., in Equation 44, have to be accumulated over all BoF vectors from all regions, cf. [Fin14, Sec. 5.7.7].

4.5.2 *Word models*

Word HMMs are used in the query-by-example scenario and represent individual word images. A word image is directly provided as bounding box annotation by the user. The bounding box coordinates are quantized with respect to the visual word gird. For representing the document region with an HMM, the sequence of T_q BoF vectors is extracted. As for the line hypotheses, descriptors are incorporated that do not overlap with the top and bottom boundary of the word region. However, descriptors that overlap to the left and to the right, help to improve the specificity of the model. They represent the direct word context which is important in order to distinguish occurrences of short words from the occurrences of the same words as parts of longer words. If the query word HMM is estimated from segmented word images, no context information is available. Descriptors that overlap with the word boundaries cannot be considered in this case.

Each BoF vector is represented in terms of the BoF output model with mixture component posteriors. The number of HMM states S is obtained as a *percentage* of the number of BoF vectors in the document image region. The HMM uses a linear topology, thus, allowing for transitions to the active state and to the next state, see Figure 26.

For *initializing* the query word HMM, the BoF vectors are linearly aligned with the states [Fin14, Sec. 9.3]. For this purpose, the state sequence $\mathbf{S} = (s_0, \ldots, s_{T_q-1})$ denotes the initial alignment such that $s_t \in \Omega_s$ and $0 \leqslant t < T_q$. The state at index t is defined in Equation 46.

$$s_t = \left\lfloor \frac{t}{T_q - 1} (S - 1) + 0.5 \right\rfloor \tag{46}$$

Given frame index t, Equation 46 computes the state s_t based on the relative position of frame t in the feature vector sequence. Adding 0.5 and using the integer part of the result corresponds to rounding to the nearest integer.

For each state, initial model parameter estimates are based on the feature vectors that have been aligned with the corresponding state. Essentially, this corresponds to Viterbi training, cf. [Fin14, Sec. 5.7.5]. The basic idea for Viterbi training is to use the *optimal* output probability $p(\mathbf{O}, \mathbf{S}^* \mid \lambda)$ as optimization criterion instead of the *total* output

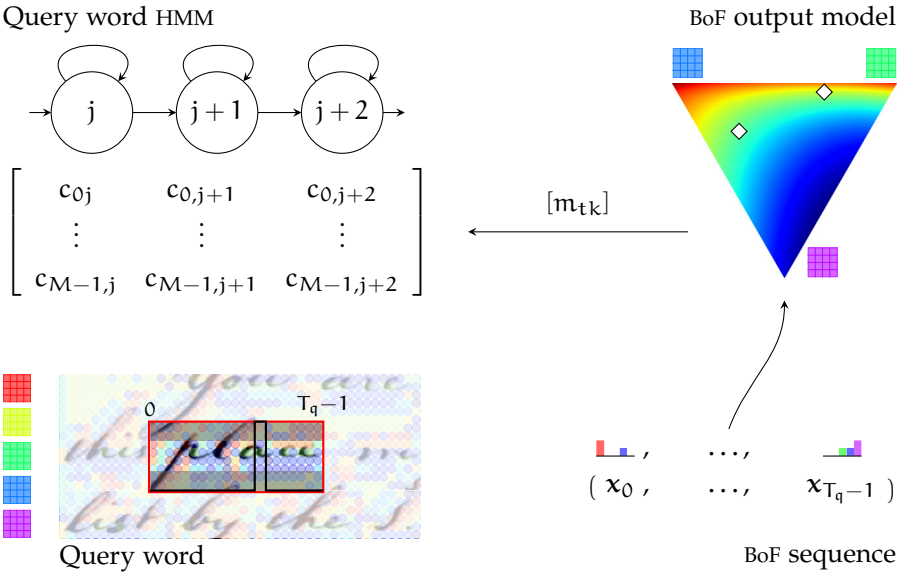


Figure 26: Query word HMM for query-by-example word spotting. The figure visualizes the process from the annotated query word region to query word HMM estimation. For this purpose, the visualization focusses on the BoF output model. The query word region is shown with a red bounding box. BoF vectors $\mathbf{x}_t$ with $0 \leqslant t < T_q$ are extracted and represented with mixture component posteriors $m_{tk} > \epsilon_{\text{low}}$ for all $k \in \Omega_M$. For each state $j \in \Omega_S$, mixture component priors $(c_{j0}, \ldots, c_{j,M-1})^\top$ are estimated. These state-dependent component probabilities are shown in matrix $[c_{kj}]$ such that the matrix columns are aligned with the corresponding states. Transition probabilities are neglected in the visualization.

probability $p(\mathbf{O}\,|\,\lambda)$ in the Baum-Welch algorithm. For this purpose, an optimal state-based alignment $\mathbf{S}^*$ is computed with the Viterbi algorithm for a training vector sequence $\mathbf{O}$, cf. Equation 8. For initialization, a linear alignment is provided according to Equation 46.

State-dependent mixture component priors are initialized by averaging the mixture component posteriors m_{tk} that have been observed in the corresponding states, see Equation 47. For this purpose, $\delta : \Omega_S \times \Omega_S \to \{0, 1\}$ serves as a function that indicates whether frame t from the training vector sequence has been aligned with state j that is considered for parameter initialization.

$$\hat{c}_{jk} = \frac{\sum_{t=0}^{T_q-1} \delta(s_t, j)\, m_{tk}}{\sum_{t=0}^{T_q-1} \delta(s_t, j)} \quad \text{with} \quad \delta(s_t, j) = \begin{cases} 1 & : s_t = j \\ 0 & : s_t \neq j \end{cases} \tag{47}$$

A lower bound of $\epsilon_{\text{low}} = 10^{-12}$ is applied to the estimates in the same way as to the estimates obtained in Equation 44. The initialization of the transition probabilities follows the standard approach in the Viterbi training [Fin14, Equ. 5.27]. The initialized model can be refined with Baum-Welch training as described in Section 4.5.1. The patch size, which is required in the patch-based decoding framework,

is directly obtained from the query word bounding box. Figure 26 shows the process from BoF vector extraction to the query word HMM.

Estimating an SC-HMM from a single document image region is only feasible because the output model parameters are not optimized along with the state-dependent parameters, cf. [PF05]. For query-by-example word spotting this approach has already been successfully applied in [RP09b]. Given that the HMM has less states than the document region has frames, the states model which mixture components occur within which section of the BoF vector sequence. Therefore, the query word HMM can be interpreted as a dynamic probabilistic extension of a spatial pyramid. Since the number of states defines the specificity of this model, the percentage of BoF vectors (see above) is a meta parameter that is very important for the word spotting accuracy and the efficiency, see Section 5.3.5.

During Baum-Welch training the probability for generating the BoF vector sequence with the HMM is optimized. Therefore, the state-dependent mixture component priors c_{jk} focus more and more on the mixture components that the BoF vectors have in common. The dynamic spatial pyramid is becoming increasingly specific with each Baum-Welch iteration. Although this effect is desired if a large number of training samples is available, an important question is whether this also applies if the mixture component priors c_{jk} are estimated with the BoF vectors from a *single* document region.

4.5.3 *Character models*

In case that a larger dataset of annotated document regions is available, HMMs can be estimated on character level. Afterwards, queries are given by-string and query word models are dynamically created as compound HMMs based on the character models. In addition, the decoding patch size is required in the patch-based framework.

For this purpose, the character models consist of an *appearance model*, i.e., the character HMM, and a *size model* that represents the expected width and height of the character. This is visualized in Figure 27. Both models are estimated from annotated training samples. Character HMMs are optimized with the Baum-Welch algorithm. Based on an uniform initialization, the training procedure is similar to the typical strategy for HMM estimation in handwriting recognition [PF09] and word spotting [FKF+12], cf. Section 4.5.1. It is adapted in order to allow for an integration in a unified framework for query-by-example and query-by-string word spotting, cf. Section 4.1.

For character model estimation, the elementary modeling units, the number of states per model and the model topology must be defined, see Section 5.3.6. The elementary units are based on the characters in the annotation. For this purpose, models can be used that capture the local character context. Given that the task is *word* spotting, this helps

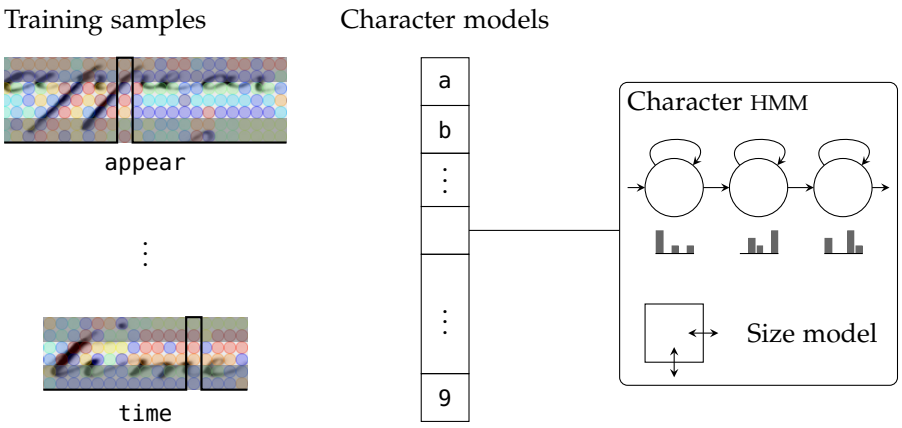


Figure 27: Character models for query-by-string word spotting are estimated using annotated training samples. On the left, the figure shows annotated word images. The estimation with annotations on line level works in analogy. The annotated document regions are represented with BoF vector sequences. Each character model consist of an appearance model and a size model as shown on the right side of the figure. The appearance is modeled with character HMMs. The distribution of mixture component priors is indicated below the HMM states. The size model represents the expected character width and character height and is obtained with a forced alignment of the training samples. It should be noted that the model inventory is not limited to characters but context-dependent models are used in addition.

to capture the visual appearance of parts-of-words. On feature level, this is supported by descriptors that typically represent more than a single character. The context-dependent models are inspired by *triphones* from speech recognition, cf. [Fin14, Sec. 8.2.2]. For this purpose, the model is defined by the characters appearing before and after the character that is supposed to be modeled. For example, a query word HMM for the query word `place` can be represented as a compound HMM of context-dependent models as shown in Equation 48.

$$\lambda(\texttt{place}) = \lambda(\texttt{\_/p/l}) \circ \lambda(\texttt{p/l/a}) \circ \lambda(\texttt{l/a/c}) \circ \lambda(\texttt{a/c/e}) \circ \lambda(\texttt{c/e/\_}) \tag{48}$$

The underscores in the first and last context-dependent model denote the word beginning and word ending. These models represent specific characters in the context of whitespace and are different from the whitespace models, cf. Section 4.5.4, which are obtained from whitespace hypotheses, cf. Section 4.2.3.

The context-dependent model inventory is derived from the training annotations. If context-dependent models are not available in order to represent specific characters in a query, the corresponding context-*independent* character models are used instead. The training process consists of two steps, cf. [Fin14, Sec. 9.3]. Based on the uni-

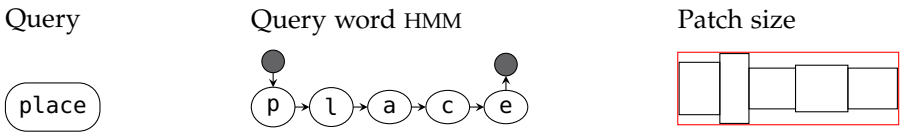


Figure 28: Query word model for query-by-string word spotting. Given a query word, a *query word HMM* and the decoding patch size must be obtained. Based on the character models, the query word HMM is given as a compound of corresponding character HMMs. The patch width is obtained by adding the individual character width estimates. The patch height is obtained as the maximum height estimate of the character models used in order to represent the query. The figure shows this exemplarily for the query `place`.

form initialization, context-*independent* models with a fixed number of states and a linear topology are estimated. Afterwards, these models are used in order to segment the training data with a forced alignment[4]. Context-dependent models are estimated based on the segmentation of the training data. The number of states per context-dependent model is chosen such that the shortest segment in the training data that was aligned with the corresponding context-independent model can still be generated. Therefore, the number of states of the context-independent models is a lower bound for the length of these segments. Using a Bakis topology, cf. [Fin14, Sec. 8.1], for the context-dependent models increases the flexibility with respect to the estimated number of states. Thus, the number of states of the context-independent models is an important meta parameter. During Baum-Welch training, the parameters of the context-independent models are tied to the parameters of the corresponding context-dependent models, cf. [Fin14, p. 172]. As for query-by-example, only the state-dependent parameters are subject to optimization.

The size model requires width and height estimates for every elementary model. The width estimates are obtained with a forced alignment of the training samples. For each elementary model, the average width is recorded. The height estimation requires more heuristics. Provided that the annotations are given on word level, as shown in Figure 27, all samples that are relevant for a particular model are recorded. The height estimate is obtained at a lower percentile of the word height distribution, see Section 5.3.6. As larger characters define the height of the word bounding box, the estimate is less accurate for small characters. The lower percentile is chosen in order to select a small word height that still contains the character. In case the annotations are given on line level, a forced alignment on word level is required. Afterwards, word bounding box heights have to be refined, e.g., using the region refinement approach presented in Section 4.6.5.

4 The alignment is referred to as *forced* because the model sequence is given during Viterbi decoding and only the segmentation has to be derived.

Once the character models are available, query word models can be obtained dynamically. The patch size is derived by adding character width estimates of the elementary models that are relevant for the query. The height is obtained as the maximum over the relevant model height estimates. Similar to the query-by-example scenario, cf. Section 4.5.2, the patch size is quantized such that width and height are multiples of the visual-word grid sampling step. Figure 28 shows an overview of the query word model definition.

The word spotting method presented in this chapter is primarily designed for the query-by-example scenario where no annotated training samples are available besides the query word image. Therefore, the support for query-by-string word spotting can be seen as an *extension* of this query-by-example scenario. The very large number of context-dependent character models as well as the optimization of only state-dependent parameters in the Baum-Welch training, require that the visual variability in the document collection is limited.

4.5.4 *Context models*

The context of a query word in a patch is represented by whitespace models as well as a background model, cf. Figure 25. The whitespace HMMs are estimated based on whitespace regions that have been obtained for the left-side as well as the right-side context of text, cf. Section 4.2.3. Since whitespace regions have a small width by design, see Figure 18, the HMMs consist of a single state each. Otherwise, the model configuration and estimation follows the procedure described for the query word HMM, cf. Section 4.5.2.

The background model represents arbitrary document image content. It is a single-state HMM which models the distribution of mixture components in the document collection, i.e., the background [RP08b]. It is given as distribution of mixture component priors $p(\mathcal{M} = k|\Theta)$ in the BoF output model Θ, cf. Section 4.4. The transition probabilities are set heuristically such that any probability for transitioning out of the background model is set to the floor probability $\epsilon_{\text{floor}} = 10^{-5}$. For example, the self-transition probability and the probability for transitioning to the next state are given by $(1 - \epsilon_{\text{floor}}, \epsilon_{\text{floor}})$ in case of a linear topology. Neither the whitespace HMMs nor the background model require their own size estimates.

The context models are required for handling word size variabilities and for decoding more accurate query word locations in the patch-based framework (Section 4.6.2). The partial output probabilities that are obtained for the context models are not considered for the patch-similarity scores. The explicit modeling of whitespace increases the specificity of the compound query HMM. This helps if the whitespace HMMs represent the context of an instance of the query word better than the background HMM, see Section 5.3.7.

4.6 RETRIEVAL

Retrieval is performed in a patch-based framework. For this purpose, patches are sampled from the document images (Section 4.6.1). Each patch receives a score that indicates similarity to the query. In order to reduce the computational complexity of evaluating a large number of patches with the Viterbi algorithm (Section 4.6.2), a coarse and a fine analysis stage is proposed. In the coarse stage, the BoF-HMM is evaluated with a voting scheme that mimics Viterbi decoding (Section 4.6.3). Only the document image regions that are potentially relevant to the query are analyzed in the fine stage. The key idea for the integration of the two stages is to re-rank all local optima identified in the first stage (Section 4.6.4). This allows for a trade-off between retrieval accuracy and efficiency. Finally, the retrieved patches can be refined by snapping patch coordinates to text hypotheses (Section 4.6.5). This allows for highly accurate detections in the document images.

4.6.1 *Patch sampling*

Patches are sampled from document images in a regular grid. The patch size is given by the query model. In order to limit the huge amount of patches, the grid resolution is dynamically adapted to the patch size in horizontal and vertical direction. Thus, more patches are extracted for smaller query words than for larger query words.

In order to align the patches with the visual word grid, the patch size has been quantized to a multiple of the visual-word grid sampling step during query model estimation. Furthermore, patch representations are based on line hypothesis representations (Section 4.6.2). For this purpose, the patch height is quantized with respect to heights of line hypotheses that have been extracted in the document image.

In the following, $\mathfrak{p}$, $\mathfrak{r}$, $\mathfrak{o}$, $\mathfrak{g}$, $\mathfrak{z}$ and $\mathfrak{d}$ refer to lengths in image pixels. The subscripts $\mathfrak{w}$ and $\mathfrak{h}$ indicate *width* and *height*. The subscripts $\mathfrak{x}$ and $\mathfrak{y}$ indicate *horizontal* and *vertical* direction. The following derivations are specific to a given document and a given patch size. Illustrations of the *patch size* and the *patch sampling step* can be found in Figure 29.

Provided that the quantized patch size is given by $(\mathfrak{p}_\mathfrak{w}, \mathfrak{p}_\mathfrak{h})$, the patch sampling steps $(\mathfrak{r}_\mathfrak{x}, \mathfrak{r}_\mathfrak{y})$ are defined in Equation 49. Horizontal and vertical offsets that have to be considered in order to exclude *invalid* descriptors, cf. Section 4.5.2, are represented by $(\mathfrak{o}_\mathfrak{w}, \mathfrak{o}_\mathfrak{h})$. For a given patch size, the offsets are chosen dynamically such that the adaptation does not reduce the patch size below a minimum number of visual-word grid rows and grid columns. The patch size $(\mathfrak{p}_\mathfrak{w}, \mathfrak{p}_\mathfrak{h})$ is not increased in this way, i.e., $\mathfrak{o}_\mathfrak{w} \geqslant 0$ and $\mathfrak{o}_\mathfrak{h} \geqslant 0$, see Figure 29 (left). The offsets are multiples of the visual-word grid sampling step.

The patch sampling steps are based on scaling the patch size by $\frac{1}{\upsilon}$. The patch sampling rate $\upsilon \in \mathbb{N}_{>0}$ is a meta parameter that has to be

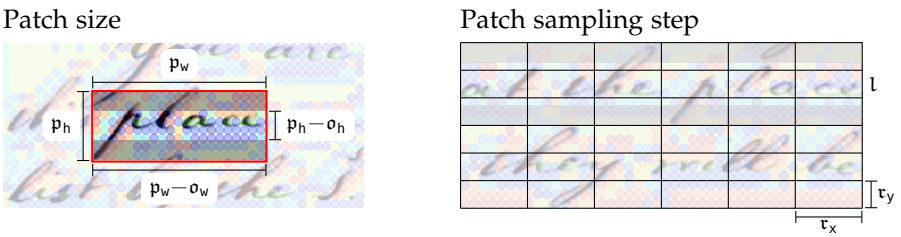


Figure 29: Patch sampling. On the left, the figure indicates patch dimensions. The patch size (p_w, p_h) is obtained from the query word model. For extracting the BoF vector sequence, the patch size is reduced according to offsets (o_w, o_h) in order to exclude invalid descriptors in the dense grid of visual words. Descriptors are invalid if they overlap with the patch boundaries. In the given example, only those descriptors are excluded that overlap with the upper and lower patch boundaries, i.e., $o_w = 0$. On the right, the figure illustrates the patch sampling step (r_x, r_y). In the example, the sampling steps are larger than in practice for a better visualization. Patches are regularly sampled as indicated by the dense grid. Patches will be processed if there exists a line hypothesis with a line height that corresponds to the patch height. A single line hypothesis with index l is indicated due to this reason.

sufficiently large, e.g., $\upsilon \geqslant 4$, cf. Table 18 in Section 5.3.8. In order to ensure that the patch positions are aligned with the visual word grid, function $q : \mathbb{Z} \times \mathbb{N}_{>0} \to \mathbb{N}_{>0}$ quantizes the horizontal and vertical sampling steps, see Equation 49. The minimum sampling step is the visual-word grid sampling step g. The mod operator computes the remainder of the integer division between the operands.

$$r_x = q\left(\left\lfloor \frac{p_w - o_w}{\upsilon} \right\rfloor, g\right), \quad r_y = q\left(\left\lfloor \frac{p_h - o_h}{\upsilon} \right\rfloor, g\right) \qquad (49)$$
$$\text{with} \quad q(z, g) = \max(z - (z \bmod g),\, g)$$

The individual sampling steps allow for sampling as many patches as are required in order to achieve accurate detections in each direction. With this trade-off, the number of possible patch positions can be reduced substantially compared to a full search, i.e., from the order of hundred thousands to the order of ten thousands.

In order to specify the number of patches, the document dimensions are required. Width and height are given by (d_w, d_h) where d_w and d_h are obtained based on the difference of the maximum and minimum visual word image coordinates in horizontal and vertical directions. Then, the number of patches in horizontal and vertical directions are given in Equation 50, provided that $d_w \geqslant (p_w - o_w)$ and $d_h \geqslant (p_h - o_h)$.

$$R = \left\lfloor \frac{d_w - (p_w - o_w)}{r_x} \right\rfloor + 1\,, \quad Q = \left\lfloor \frac{d_h - (p_h - o_h)}{r_y} \right\rfloor + 1 \qquad (50)$$

Patches are sampled from the document such that all patches lie within the document boundaries.

4.6.2 *Viterbi decoding*

The basic idea for searching document images with the compound query HMM, see Section 4.5, is to represent patches with sequences of BoF vectors. Viterbi decoding yields the optimal probability for generating the sequences with the HMM. In order to improve retrieval speed, the evaluation of the BoF output model is precomputed and stored in look-up tables for line hypotheses.

For each document, line hypothesis positions are grouped according to line hypothesis heights in an index, cf. Section 4.2.2. Since the patch height has been quantized to the index heights, line hypotheses are obtained that fit the decoding patch height. Based on this set Λ of relevant line hypothesis positions, the regular grid of patches is filtered such that only those patches will be processed that lie within line hypotheses that are relevant for the patch height.

In the Viterbi algorithm, the output probabilities $b'_j(\mathbf{x}_t^{[l]}) = \mathbf{c}_j^\top \mathbf{m}_t^{[l]}$ are required, cf. Equation 43. For line hypothesis l, the mixture component posteriors $\mathbf{m}_t^{[l]}$ can efficiently be obtained from the mixture component index, cf. Figure 24. The state-dependent vector of mixture component priors $\mathbf{c}_j$ is given by the model.

A patch representation is defined as a subsequence of the corresponding line representation. Provided that line l is represented with vectors $\mathbf{x}_t^{[l]}$ with $0 \leqslant t < T_l$, then a patch with F BoF vectors is defined with sub-indices $t + f$ with $0 \leqslant t \leqslant T_l - F$ where $F \leqslant T_l$ and $0 \leqslant f < F$. Based on offset t and patch sequence index f, the corresponding mixture component posterior vector $\mathbf{m}_{t+f}^{[l]}$ is retrieved from the look-up table. Equation 51 defines the number of BoF vectors per line and per patch, also cf. Section 4.6.1.

$$T_l = \frac{\vartheta_w}{g} + 1\,, \quad F = \frac{p_w - o_w}{g} + 1 \tag{51}$$

Figure 30 visualizes the concept. For this purpose, it is indicated that patches are sampled from the document. For each patch position, the BoF vector sequence is obtained from the look-up table. This is shown for the patch in line hypothesis l at BoF vector index t.

For a given patch size, patches are arranged in a regular $Q \times R$ grid, also see Figure 29 (right), where Q is the number of rows and R is the number of columns, as defined in Equation 50. Individual patches can be addressed with tuple $(q, r) \in \{0, \ldots, Q-1\} \times \{0, \ldots, R-1\}$. Corresponding lines can be addressed based on the patch row indices $l \in \Delta$ with $\Delta = \Lambda \cap \{0, \ldots, Q-1\}$. For a patch column index r, the BoF vector offset t in the line hypothesis is computed with function $g : \{0, \ldots, R-1\} \to \{0, \ldots, T_l - F\}$ based on the patch sampling step r_x and the visual-word grid sampling step g. Equation 52 defines the sequence of F BoF vectors $\mathbf{O}_r^{[l]}$ for patch (l, r).

$$\mathbf{O}_r^{[l]} = \left(\mathbf{x}_{g(r)+0}^{[l]}, \ldots, \mathbf{x}_{g(r)+F-1}^{[l]}\right) \quad \text{with} \quad g(r) = r\,\frac{r_x}{g} \tag{52}$$

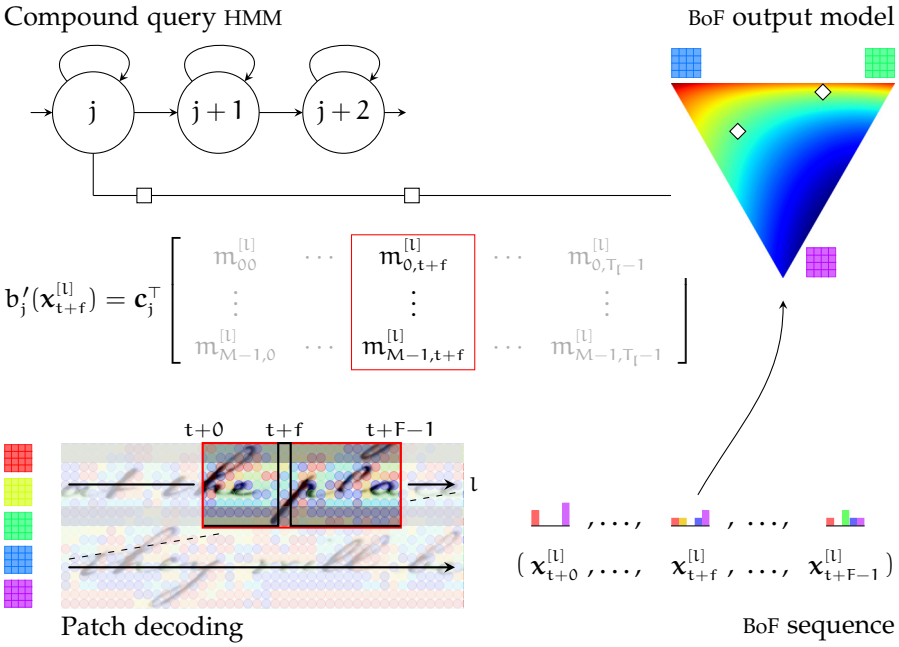


Figure 30: Patch-based decoding of the compound query HMM. The figure shows how the patch-based framework is integrated with line hypotheses. For this purpose, sequential patch representations are obtained as subsequences of the line representations as indicated by offset t and patch index f within line hypothesis l. Mixture component posterior probability vectors $\mathbf{m}^{[l]}_{t+f} = (m^{[l]}_{t+f,0}, \ldots, m^{[l]}_{t+f,M-1})^\top$ can directly be obtained from the mixture component index. Given the state-dependent mixture component prior vectors $\mathbf{c}_j = (c_{j0}, \ldots, c_{j,M-1})^\top$ of the compound query HMM, the output probability $b'_j(\mathbf{x}^{[l]}_{t+f}) = \mathbf{c}_j^\top \mathbf{m}^{[l]}_{t+f}$ is computed with a scalar product. This is indicated by the arrow with the two square markers. Further, the vector $\mathbf{m}^{[l]}_{t+f}$ is highlighted by a red frame. The arrow is reaching from HMM state j towards the visual word simplex in order to emphasize that the output probability depends on the distribution represented by the BoF output model.

The Viterbi algorithm computes $p(\mathbf{O}^{[l]}_r, \mathbf{S}^{[l]*}_r \,|\, \lambda)$, i.e., the optimal output probability for BoF vector sequence $\mathbf{O}^{[l]}_r$ and the optimal state sequence $\mathbf{S}^{[l]*}_r$. For this purpose, the beam search algorithm [Fin14, Sec. 10.2.1] is applied with a negative-logarithmic beam offset of 200 and a floor probability [Fin14, Sec. 7.2] of $\epsilon_{\text{floor}} = 10^{-5}$ for any output probability, i.e., $b'_j(\mathbf{x}^{[l]}_t) \leftarrow \max\left(b'_j(\mathbf{x}^{[l]}_t), \epsilon_{\text{floor}}\right)$. The beam parameter is chosen such that an optimal output probability can be computed for most of the patches. Since the probabilities along state sequences are based on the product of output probabilities, the lower bound ensures that paths are not excluded due to individually occurring very small output probabilities.

Based on the optimal alignment $\mathbf{S}^{[l]*}_r$, the optimal output probability $p\left(\mathbf{O}^{[l]}_r, \mathbf{S}^{[l]*}_r \,|\, \lambda\right)$ can be expressed in terms of the individual align-

ments with the states of the models in the compound query HMM, cf. Figure 25. For this purpose, Equation 53 defines subsequences of BoF vectors and states. The notation for a sequence at row index l and column index r, i.e., $\mathbf{O}_r^{[l]}$, is extended such that the column index r can be *conditioned* on the subsequence for the corresponding models of the compound HMM. The subsequence that corresponds to models which are representing the *left side context* in a patch, i.e., left side background and left side whitespace, is denoted as u. In analogy, the *right side context* is denoted as v. The subsequence that has been aligned with the *query word* HMM is denoted as q. The definition of the subsequences is based on the estimated frame offset $\hat{f}_q$ for the query word model and the estimated number of frames $\widehat{F}_q$, that have been aligned with the query word model within a patch.

$$\begin{aligned}
\mathbf{O}_{r|u}^{[l]} &= \left(\boldsymbol{x}_{g(r)+0}^{[l]}, \ldots, \boldsymbol{x}_{g(r)+\hat{f}_q-1}^{[l]}\right), \quad \mathbf{S}_{r|u}^{[l]*} = \left(s_{g(r)+0}^{[l]*}, \ldots, s_{g(r)+\hat{f}_q-1}^{[l]*}\right) \\
\mathbf{O}_{r|q}^{[l]} &= \left(\boldsymbol{x}_{g(r)+\hat{f}_q}^{[l]}, \ldots, \boldsymbol{x}_{g(r)+\hat{f}_q+\widehat{F}_q-1}^{[l]}\right), \mathbf{S}_{r|q}^{[l]*} = \left(s_{g(r)+\hat{f}_q}^{[l]*}, \ldots, s_{g(r)+\hat{f}_q+\widehat{F}_q-1}^{[l]*}\right) \\
\mathbf{O}_{r|v}^{[l]} &= \left(\boldsymbol{x}_{g(r)+\hat{f}_q+\widehat{F}_q}^{[l]}, \ldots, \boldsymbol{x}_{g(r)+F-1}^{[l]}\right), \ \mathbf{S}_{r|v}^{[l]*} = \left(s_{g(r)+\hat{f}_q+\widehat{F}_q}^{[l]*}, \ldots, s_{g(r)+F-1}^{[l]*}\right)
\end{aligned} \tag{53}$$

Due to the independence assumptions in the HMM[5], $p\left(\mathbf{O}_r^{[l]}, \mathbf{S}_r^{[l]*} \mid \lambda\right)$ can be expressed in terms of the *partial* alignments of the BoF vector sequence with the states in the compound query HMM as shown in Equation 54. The *partial* output probability for the query word model is obtained in Equation 55. Equation 55 is *equivalent* to Equation 54. Therefore, the ratio in Equation 55 does *not* correspond to a filler-score normalization. The denominator does *not* represent the *entire* observation sequence $\mathbf{O}_r^{[l]}$.

$$p\left(\mathbf{O}_r^{[l]}, \mathbf{S}_r^{[l]*} \mid \lambda\right) = p\left(\mathbf{O}_{r|u}^{[l]}, \mathbf{S}_{r|u}^{[l]*} \mid \lambda\right) p\left(\mathbf{O}_{r|q}^{[l]}, \mathbf{S}_{r|q}^{[l]*} \mid \lambda\right) p\left(\mathbf{O}_{r|v}^{[l]}, \mathbf{S}_{r|v}^{[l]*} \mid \lambda\right) \tag{54}$$

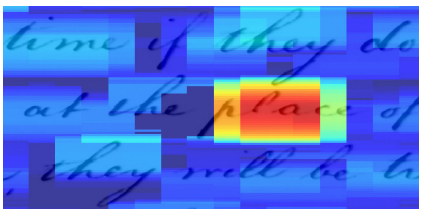
$$\Leftrightarrow p\left(\mathbf{O}_{r|q}^{[l]}, \mathbf{S}_{r|q}^{[l]*} \mid \lambda\right) = \frac{p\left(\mathbf{O}_r^{[l]}, \mathbf{S}_r^{[l]*} \mid \lambda\right)}{p\left(\mathbf{O}_{r|u}^{[l]}, \mathbf{S}_{r|u}^{[l]*} \mid \lambda\right) p\left(\mathbf{O}_{r|v}^{[l]}, \mathbf{S}_{r|v}^{[l]*} \mid \lambda\right)} \tag{55}$$

Since the partial output probabilities depend on the number of frames $\widehat{F}_q$ that have been aligned with the query word model, the patch scores are obtained after length-normalization, see Equation 56.

The patch similarity scores are stored in the $Q \times R$ matrix $[v_{qr}]$. Equation 56 defines the similarity score for the patch at row index q and column index r. The length-normalization is performed by the geometric mean because the output probability over a sequence of observations and states is represented as a product. Thus, the patch similarity score corresponds to the average output probability for the path through the query word model. Patches that are not processed with the Viterbi algorithm as well as patches that do not obtain a

5 Provided that transition probabilities between the context HMMs and the query word HMM are neglected.

Patch similarity scores



Patch-based alignment

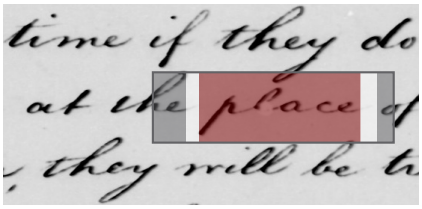


Figure 31: Patch-based retrieval for query `place`, cf. Figure 26. On the left, the figure shows patch similarity scores for a section of a document image. Each patch center is indicated as a rectangular cell. The cell color visualizes the similarity to the query from blue to red in the logarithmic domain. The cell width and cell height correspond to the patch sampling rates. On the right, the figure shows the patch that corresponds to the strongest local optimum within the patch similarity scores. The patch size has been increased by 50%, in order to align the BoF vectors with the *background* (gray), *whitespace* (white) and *query* (red) models. The flexibility of the approach is emphasized by the differently sized background and whitespace regions to the left and to the right of the query word.

similarity score due to beam pruning are assigned the floor probability $\epsilon_{\text{floor}} = 10^{-5}$. The floor probability is the same as for output probabilities in the Viterbi algorithm and can, therefore, be used as a lower bound after length-normalization of the patch scores. The lower bound for any patch score is required in order to be able to represent patch similarity scores in the logarithmic domain.

$$\nu_{qr} = \begin{cases} \max\left(\sqrt[\hat{F}_q]{p\left(\mathbf{O}_{r|q}^{[q]}, \mathbf{S}_{r|q}^{[q]*} \,|\, \lambda\right)}, \epsilon_{\text{floor}} \right) & : \quad q \in \Lambda \\ \epsilon_{\text{floor}} & : \quad q \notin \Lambda \end{cases} \tag{56}$$

The computational effort for computing matrix $[\nu_{qr}]$ depends on the number of patches and the computational complexity of the Viterbi algorithm. Provided that F is the number of BoF vectors in a patch, S is the number of states in the query model and M is the number of mixture components in the BoF output model, the complexity for evaluating a single patch is $O(FMS^2)$. By exploiting the model topology in the beam search algorithm, cf. [Fin14, Sec. 10.2.1], and by exploiting the sparsity of mixture component posterior vectors $\mathbf{m}_t^{[l]}$, cf. Section 4.4.4, the computational complexity reduces to $O(F\hat{M}\hat{S}^2)$ where $\hat{M} \ll M$ and $\hat{S} \leqslant 3$ for a Bakis topology or $\hat{S} \leqslant 2$ for a linear topology. Provided that $L = |\Delta|$ line hypothesis have to be considered for the evaluation of an entire document image and R patches are extracted per line, the computational complexity for obtaining matrix $[\nu_{qr}]$ is $O(LRF\hat{M}\hat{S}^2)$.

Figure 31 (left) visualizes patch scores for a section of a document image in the logarithmic domain. Due to the high overlap of neighboring patches, the score distribution is relatively smooth. Consequently,

the similarity scores start raising as soon as patches start to overlap with document image regions that are visually similar to the query. The similarity scores reach local maxima for the patches that are centered over document image regions that are most similar to the query.

This effect is exploited for retrieving relevant patches with *non-maximum suppression* (NMS). Since the patch scores are stored in the *matrix* $[v_{qr}]$, locally optimal scores can be obtained after applying a maximum rank-order filter, cf. [GW02, p. 124]. All scores that did not change after the filter operation are locally optimal. The size of the local neighborhood is defined by the size of the maximum-filter mask. For each patch, the overlapping patches can be determined based on the patch size and the patch sampling steps. Given that υ patches are sampled in horizontal direction within the bounds of a patch, $\upsilon - 1$ patches have their left corners and $\upsilon - 1$ patches have their right corners within these bounds. Based on this observation, the filter mask has width $2\upsilon - 1$. Equation 57 specifies the size of the mask in horizontal and vertical direction with respect to matrix $[v_{qr}]$. The number of rows is given by G_y and G_x denotes the number of columns.

$$G_x = \left\lfloor \frac{p_w - o_w}{r_x} \right\rfloor 2 - 1 \,, \quad G_y = \left\lfloor \frac{p_h - o_h}{r_y} \right\rfloor 2 - 1 \tag{57}$$

G_y and G_x are not expressed in terms of υ directly due to possible quantization artifacts, cf. Equation 49. Figure 31 (right) shows a single patch which corresponds to the local optimum with the highest similarity score.

An important assumption of patch-based word spotting systems is that the patch size is similar to the size of words that are relevant to the query. However, in documents with larger word size variability this can be a considerable limitation. Due to the dynamic probabilistic properties of the compound query HMM, word size variability can be handled better than in patch-based approaches using holistic representations. The estimated bounds of the query word within a patch are directly given by the partial BoF vector sequence $\mathbf{O}^{[q]}_{r|q}$ that has been aligned with the states of the query word model. In order to improve the flexibility with respect to occurrences of the query that are larger than the patch size, the patch width can be increased according to $p_w \leftarrow p_w + \psi p_w$ with meta parameter $\psi \in \mathbb{R}_{\geqslant 0}$. The sampling steps in Equation 49 are adapted accordingly. Figure 31 (right) indicates the alignment of the patch with background, whitespace and query word models. The patch size has been increased by 50%, i.e., $\psi = 0.5$.

The improved word size flexibility comes at the cost of reduced specificity of the scores in the patch score matrix $[v_{qr}]$. The Viterbi algorithm computes the optimal path for aligning the states in the compound query HMM with the frames in a patch. In this regard, the number of states of the context models is fixed and the number of states in the query word model depends on the expected width of the query word. Therefore, the number of possibilities for aligning

the query word model with a subsequence of frames depends on the difference between the number of frames in a patch and the number of states in the query word HMM. If the difference is small, the patches are mostly represented by the query word model which makes the scores very specific. If the difference grows, the Viterbi algorithm has more possibilities for computing an alignment for the query word model. The more possibilities for obtaining the path with *maximum* output probability are available, the less specific will the similarity scores be with respect to the spatial location of the patches in the document image. Therefore, the resolution of the similarity scores decreases with increasing values of ψ. This is particularly important for retrieving occurrences of the query that are *not* represented well by the query word model.

Furthermore, it should be noted that the patch similarity scores also represent the probabilities of the observations. In contrast, a query posterior is conditioned on the observations due to the filler normalization [PTV15a]. Observations that are likely obtain higher probabilities than unlikely observation. Therefore, the filler normalization can be seen as a writing style normalization. However, if only very limited amounts of annotated samples are available, the generalization over *very different* writing styles is not the targeted scenario. For this reason, the normalization with a filler model or a background model is ommited in favor of better retrieval accuracy for spotting within document collections with limited writing style variability, see Section 3.3.2 and Table 19 in Section 5.3.8.

4.6.3 *Mixture component voting*

Computing similarity scores in a patch-based framework with the Viterbi algorithm is only feasible due to patch sampling and mixture component indexing based on line hypotheses. However, in real world scenarios where users expect retrieval results instantly, this is insufficient. The computational complexity $O(LRF\hat{M}\hat{S}^2)$ is still dominated by the large number of patches per document image, i.e., typically $LR > 10,000$. Principally, this is a challenge for any patch-based word spotting system. In this regard, the major difference between HMMs and holistic representations is that the HMM query model is not indexable with standard approaches, cf. Section 3.3.3.

Mixture component voting allows for obtaining potentially relevant patches *without evaluating each patch individually*. The approach is inspired by the application of the *generalized Hough transform* [Bal81] to object detection with SIFT descriptors [Low04, Sec. 7]. For this purpose, local features from the model are matched with local features from the document image. The model defines a reference point in order to obtain a relative displacement for each match. The matching features vote for an object hypothesis by increasing an accumulator in

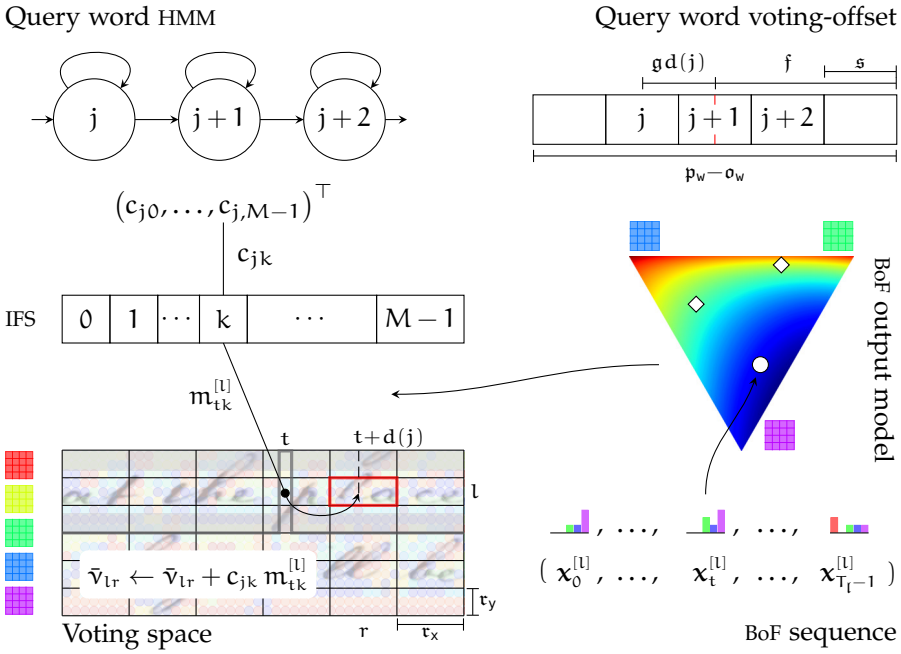


Figure 32: Mixture component voting. The figure shows how BoF vectors are indexed with respect to mixture components in an IFS. A Hough-style voting scheme allows for retrieving document regions rapidly that are potentially relevant to the query. For this purpose, component posteriors $m_{tk}^{[l]}$ are computed for BoF vectors $x_t^{[l]}$ where l and t denote line and frame indices. The frame position at index t is indicated in gray and the frame center is shown as a black dot. For each mixture component, the index stores tuples $\left(m_{tk}^{[l]}, (l,t)\right)$ with $m_{tk}^{[l]} > \epsilon_{\text{low}}$ for all frames. The voting process is indicated for HMM state j. For each component k with $c_{jk} > \epsilon_{\text{low}}$, all relevant frames are retrieved from the index. The figure shows a single match for state j and frame (l,t). The voting mass is added to elements in $Q \times R$ matrix $[\bar{v}_{qr}]$. Each matrix element corresponds to an accumulator cell in the document image. Cells correspond to patches and their size equals to the patch sampling steps (τ_x, τ_y). They are visualized in a regular grid over the document image. It can be seen that the center-coordinate of frame (l,t) is located in cell $(l, r-1)$. However, in the given example the voting mass $c_{jk} m_{tk}^{[l]}$ is accumulated in cell (l,r), which is highlighted in red. This is due to frame *voting-offset* $d(j)$ which incorporates the sequential HMM structure. The computation of the voting-offset uses the patch width $p_w - o_w$ and the number of HMM states S. The entire width is subdivided in S sections of uniform size s. The offset is based on the displacement between section centers and the patch reference point at the horizontal patch center. The horizontal patch center is indicated in red with two vertical marks. Numerically, the reference point is referred to as f which corresponds to the distance between the horizontal patch center and either of the horizontal patch bounds as shown for the right bound in the figure. For state j, the displacement with respect to the reference point in image pixels is indicated as $gd(j)$. The frame index $t + d(j)$ must be transformed to cell column index r according to $r = g^\star(t + d(j))$, see Equation 65.

the Hough voting space. The voting space is represented by a regular grid of accumulator cells over the image. In this voting process, the cell coordinates are obtained based on the coordinates of the matching features in the document image and their relative displacement. The key idea for integrating mixture component voting in the patch-based framework is that each accumulator cell directly corresponds to a patch. The accumulator cells do not overlap and the cell size is defined by the patch sampling steps, see Section 4.6.1. Figure 32 shows a schematic visualization of the entire mixture component voting approach. The visualization of the BoF vector sequence in line hypothesis l, the BoF output model and the query word HMM are known from the previous figures. The visualization of the accumulator cells and the cell size (r_x, r_y) corresponds to the visualization of the patch sampling steps in Figure 29. The figure caption provides a summary of the method. The details will be presented in the following.

Mixture component voting is performed with the *query word* HMM from the *compound* query HMM. The context HMMs are *not* used in the process. Given the query word model, mixture components with prior probabilities c_{jk} for all $j \in \Omega_S$ and all $k \in \Omega_{\mathcal{M}}$ are considered as local features. The features are local since they are specific to state j in the state *sequence* of the query word model. In the same way, mixture components from the document image are represented with mixture component posteriors $m_{tk}^{[l]}$ for all $l \in \Delta$ with $0 \leqslant t < T_l$. These can be considered as *local* features as well because they are specific to line l and frame t. The key idea is to match all state-dependent prior probabilities with all posteriors in the document image for all components $k \in \Omega_{\mathcal{M}}$. Instead of a binary match indicator, matches are *soft*. This is represented by the voting mass in Equation 58.

$$c_{jk}\, m_{tk}^{[l]} = p(\mathcal{M}_t = k \,|\, \mathcal{S}_t = j, \lambda)\, p(\mathcal{M}_t = k \,|\, \mathbf{x}_t^{[l]}, \lambda) \tag{58}$$

The voting mass is the joint probability for mixture component k given BoF vector $\mathbf{x}_t^{[l]}$ and mixture component k given state j in model λ. Thus, the voting mass is zero if any of the probabilities is zero. The local features can be considered as *non-matching* in this case.

Matches can be retrieved rapidly by extending mixture component look-up tables, cf. Figure 24, with inverted indices. For this purpose the IFS is defined as Υ in Equation 59. For each mixture component $k \in \Omega_{\mathcal{M}}$, Υ_k stores line-frame indices (l, t) of all frames in all line hypotheses that are represented with a *non-zero* posterior probability ($\epsilon_{\text{low}} = 10^{-12}$), see Equation 60. Further, the posteriors are stored along with the indices as weights.

$$\Upsilon = \left\{ \Upsilon_k \,|\, 0 \leqslant k < M \right\} \tag{59}$$

$$\Upsilon_k = \left\{ \left(m_{tk}^{[l]}, (l, t) \right) \,|\, \forall (l, t) \in \Delta \times \{0, \ldots, T_l - 1\} : m_{tk}^{[l]} > \epsilon_{\text{low}} \right\} \tag{60}$$

Figure 32 shows a single IFS entry for component posterior $m_{tk}^{[l]}$ of BoF vector $\mathbf{x}_t^{[l]}$ which has been extracted from frame t in line l.

For each match, the relative displacement with respect to a reference point in the query word model is required. The displacement is based on the patch width, the number of HMM states and the index of the corresponding HMM state. The HMM states represent the visual appearance of the query word in horizontal direction. The spatial extent of each state with respect to the patch width can be approximated with s for any state, see Equation 61, where $p_w - o_w$ is the patch width and S is the number of HMM states. Thus, the patch is subdivided in S *sections* of uniform width s. The reference point f is the horizontal patch center.

$$s = \frac{p_w - o_w}{S}, \quad f = \frac{p_w - o_w}{2} \tag{61}$$

Equation 62 defines the relative horizontal displacement between a state j and the reference point in terms of frame indices. The result of function $d(j)$ is referred to as the *voting-offset*. Mapping from state indices to frame indices is convenient for the integration of the IFS and the voting space in the voting algorithm (see below).

The definition of function $d(j)$ in Equation 62 is based on the displacement in image pixels between the section that corresponds to state j and the reference point. Since the reference point is the patch *center*, the displacement is defined with respect to the section centers. The left bound of the section that corresponds to state j is given by js. The horizontal section center is obtained after adding $\frac{s}{2}$. The difference with respect to f results in the displacement in image coordinates such that a positive displacement is obtained for states representing the beginning of the query word and a negative displacement is obtained for states representing the end of the query word. Thus, the displacement is relative with respect to the reference point.

$$d : \left\{0, \ldots, S-1\right\} \rightarrow \left\{-\left\lfloor \frac{T_l}{2} \right\rfloor, \ldots, \left\lfloor \frac{T_l}{2} \right\rfloor\right\}, \; d(j) = \left\lfloor \frac{f - \left(j\,s + \frac{s}{2}\right)}{g} \right\rfloor \tag{62}$$

Function $d(j)$ maps a state index to a horizontal displacement in terms of frame positions by dividing by grid sampling step g. Consequently, the co-domain is bounded based on the number of frames T_l in a line. Figure 32 visualizes the horizontal displacement in image pixels $g\,d(j)$ for state j. This involves the *voting-offset* $d(j)$ which is used in the voting algorithm. In this regard, the application of $d(j)$ is shown in the *voting space*.

The Hough voting space over a document image is quantized in accumulator cells of width and height (r_x, r_y) such that the center-coordinates of the cells are aligned with the center-coordinates of the patches, cf. Section 4.6.1. Since the cell size equals to the patch sampling step, the cells are non-overlapping and there exists exactly one cell per patch. This correspondence allows for obtaining similarity

scores for patches based on the accumulated voting scores of the cells. In the same way as Viterbi-based patch scores are stored in $Q \times R$ matrix $[v_{qr}]$, cf. Equation 56, votes are accumulated in matrix $[\bar{v}_{qr}]$. The accumulator values are initialized to ϵ_{floor}, see Equation 63. The floor probability $\epsilon_{\text{floor}} = 10^{-5}$ is used in analogy to Equation 56. Figure 32 shows the voting space as a regular grid of cells over the document image.

The voting algorithm can be defined for *non-zero* state-dependent prior probabilities $c_{jk} > \epsilon_{\text{low}}$ for all $j \in \Omega_S$ and all $k \in \Omega_M$ from the query word HMM. All corresponding non-zero mixture posteriors in a document image, i.e., $\forall\big(m_{tk}^{[l]}, (l,t)\big) \in \Upsilon_k$, are retrieved from the IFS. Each element votes for a patch by adding the voting mass $c_{jk}\, m_{tk}^{[l]}$ to the accumulator in the voting space. The corresponding accumulator-cell index is determined based on line-frame index (l,t) *and* voting-offset $d(j)$, see Equation 64. Index l is a valid patch row index since $l \in \Delta \subseteq \{0,\dots,Q-1\}$. Frame index $t + d(j)$ is mapped to the corresponding patch column index by function $g^\star(t + d(j))$, see Equation 65.

$$\bar{v}_{qr} = \epsilon_{\text{floor}} \quad \forall (q,r) \in \{0,\dots,Q-1\} \times \{0,\dots,R-1\} \tag{63}$$

$$\begin{aligned} &\forall (j,k) \in \Omega_S \times \Omega_M : c_{jk} > \epsilon_{\text{low}},\ \forall\big(m_{tk}^{[l]}, (l,t)\big) \in \Upsilon_k\ . \\ &\qquad \bar{v}_{l,g^\star(t+d(j))} \leftarrow \bar{v}_{l,g^\star(t+d(j))} + c_{jk}\, m_{tk}^{[l]} \end{aligned} \tag{64}$$

Function $g^\star(t)$ is related to the inverse of function $g(r)$, cf. Equation 52. The difference lies in the domain definitions which is important for handling document boundaries. In this regard it has to be noted that an accumulator is only increased in Equation 64, if $t + d(j)$ is in the domain of function $g^\star$. This handling of document boundaries is required because patches are entirely located *within* a document. Thus, frame indices outside the domain would not be mapped to a valid patch column index. Further, $g^\star(t)$ rounds down in order to map a range of frame indices to a single patch column index.

$$g^\star : \left\{0,\dots,T_l - \left\lceil \frac{F}{2} \right\rceil - 1\right\} \rightarrow \left\{0,\dots,R-1\right\},\ g^\star(t) = \left\lfloor t\, \frac{g}{r_x} \right\rfloor \tag{65}$$

The mixture component voting algorithm can be interpreted as a coarse evaluation of output probabilities $b'_j\big(x_t^{[l]}\big)$. For each state and each BoF vector, the output probabilities are computed componentwise and added to the cell that corresponds to the patch at line $l \in \Delta$ and frame $t + d(j)$, if $t + d(j)$ can be mapped to a valid patch column index. Assuming that the query word HMM has just a single state, the voting scores correspond to the HMM output probabilities. For multiple states, the *sums are extended* to output probabilities for frames in the horizontal neighborhood that vote for the same cell. The voting-offset reflects the model structure due to its dependence on state j. The output probability for *any* BoF vector in a cell, i.e., vectors that are

consistent with the sequential model structure, is considered. Similarity scores $[\bar{v}_{qr}]$ are, therefore, sensitive to individual, high output probabilities. This leads to high recall at the cost of a high false positive rate.

Figure 32 indicates the voting process for state-dependent component prior c_{jk} and component posterior $m_{tk}^{[l]}$ which is obtained through the IFS. The voting mass is accumulated in cell (l, r) that contains frame $t + d(j)$. Figure 33 (top, left) visualizes patch similarity scores for a section of a document image in the logarithmic domain. Patches can be retrieved according to similarity to the query word HMM after applying an NMS filter to the matrix $[\bar{v}_{qr}]$, see Figure 33 (top, right). The size of the filter mask is based on the patch size, cf. Equation 57, as for patch-based retrieval in Section 4.6.2.

The computational complexity is $O(S\hat{M}_{HMM}\hat{M}_{IFS})$ according to Equation 64. In this regard, S is the number of HMM states, $\hat{M}_{HMM}$ is the expected number of non-zero state-dependent component priors per state and $\hat{M}_{IFS}$ is the expected number of IFS entries per component. In comparison to the computational complexity for Viterbi decoding, it is important to note that the computational complexity for mixture component voting is independent of the number of patches. This is a considerable advantage for providing retrieval results fast. However, this requires that at least the look-up table and the corresponding inverted index, which is required for processing the current query, can be stored in memory.

Memory requirements mostly depend on the number of line hypotheses, the number of frames per line and the expected number of mixture components per frame. Provided that the look-up tables and the IFS are stored independently, a factor of two has to be incorporated for sparse representations in the look-up tables. A factor of three is incorporated for the IFS in order to store posteriors and line-frame indices. Therefore, the memory complexity per line-height and document is $2 \cdot O(LT_l\hat{M}) + 3 \cdot O(LT_l\hat{M}) = 5 \cdot O(LT_l\hat{M})$.

In case a very large volume of fast disk storage is available, indices can be loaded from a hard drive dynamically at query time. It should be noted that this can dominate retrieval times of the mixture component voting approach severely. The delay can be considered as less critical if patches are re-ranked with Viterbi decoding. This is due to the overall increased computational complexity.

4.6.4 *Two-stage integration*

The patch-based Viterbi decoding approach computes accurate retrieval results at the cost of low computational efficiency, cf. Section 4.6.2. The mixture component voting approach computes retrieval results efficiently at the cost of accuracy, cf. Section 4.6.3. However, accuracy is mostly affected with respect to the ranking of the retrieval

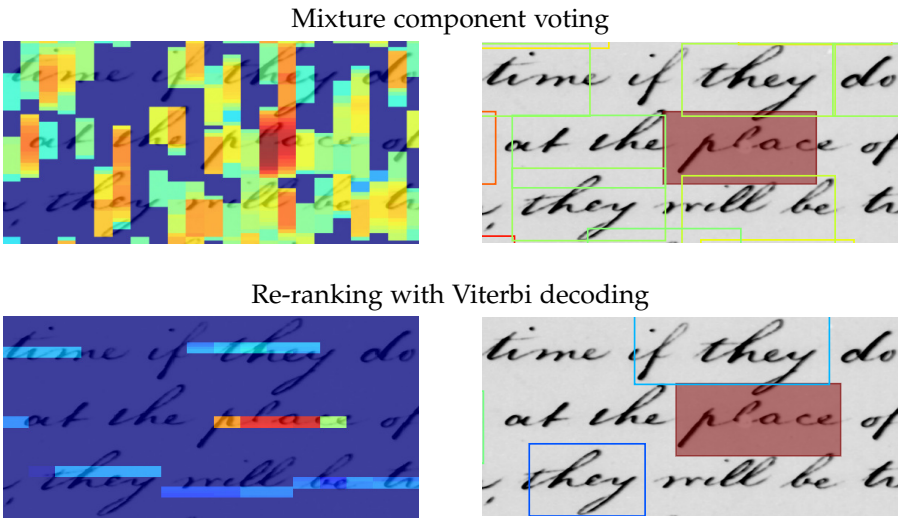


Figure 33: Two-stage integration. The figure shows patch-based similarity scores in the logarithmic domain for mixture component voting (top left) and re-ranking with the Viterbi algorithm (bottom left). For this purpose, a local neighborhood of 5×5 patches is considered. Patch similarity scores are indicated with blue to red colors. The corresponding regions that have been retrieved after NMS are show on the right. The *soft* NMS scheme has been applied for retrieving patches after mixture component voting (top right). After re-ranking, the horizontal localization of the query word has been improved (bottom right). It can be seen how the regions have different width which is due to decoding the most likely occurrence of the query word in each patch.

list. Recall, i.e., the completeness of the retrieval list with respect to relevant results, is not affected as much. Since both approaches are built on the same patch-based framework, cf. Section 4.6.1, they can directly be integrated in a two-stage method, also see Section 5.3.9.

The two-stage integration is achieved by obtaining potentially relevant patches with mixture component voting and re-ranking these patches with the Viterbi algorithm. In order to benefit from Viterbi decoding the most, re-ranking is applied to the top-200 patches *per document* and not only to the top results in the retrieval list, e.g., as in [AGF+14b]. Re-ranking a high number of patches *per document* decreases the risk of suppressing relevant patches that did not obtain a high similarity score in the first decoding stage. The number 200 is a heuristic upper bound for the expected number of relevant patches per document.

Since mixture component voting is sensitive to document image regions that are partially similar to the query word, an extension to patch-based NMS is proposed. This is required, since these high-ranked patches might occur in close proximity to each other, cf. Figure 33, and only the patch with the best score will be considered. The naive solution is to reduce the size of the maximum filter mask in

Equation 57. However, this increases the number of patches that have to be re-ranked considerably. In order to compromise between the number of patches and re-ranking mostly all relevant patches, similarity scores $[\bar{v}_{qr}]$ are smoothed with a discrete anisotropic Gaussian filter, cf. [GSW03], before applying NMS. The key idea for this *soft* NMS is that the filter mask for the Gaussian is larger than the mask for the maximum filter. For this purpose, Equation 57 is extended with parameter $\zeta \in \{1, 2\}$ as shown in Equation 66. The size of the Gaussian filter is obtained with $\zeta = 2$ which corresponds to the spatial extent of a patch. The size for the maximum filter is obtained with $\zeta = 1$, thus, patches are allowed to overlap up to 50% with a higher ranked patch without being suppressed.

$$G_x = \left\lfloor \frac{p_w - o_w}{r_x} \right\rfloor \zeta - 1 \,, \quad G_y = \left\lfloor \frac{p_h - o_h}{r_y} \right\rfloor \zeta - 1 \tag{66}$$

The application of the Gaussian emphasizes document regions where many patches have high similarity scores, while document image regions with just few high-ranked patches are weighted down. The number of local optima is reduced due to the large filter mask. *Soft* NMS makes NMS filtering sensitive to the similarity score distribution and allows to detect strong local optima in close proximity to each other while weak local optima in close proximity to a strong local optimum will be suppressed, cf. Table 22 and 23 in Section 5.3.9.

As long as the vertical positioning of patches that have been retrieved with mixture component voting is accurate, small displacements in horizontal direction can be compensated with Viterbi-based decoding. This is due to the compound structure of the query HMM that allows for inferring the most likely occurrence of the query word *within* the patch. Achieving accurate vertical localization is easier than achieving accurate horizontal localization, because distinguishing adjacent lines is easier than distinguishing adjacent words.

Further improvements can be achieved if more than just a single locally optimal patch is re-ranked with the Viterbi algorithm. For this purpose, the locally optimal patches are indicated as ones in a binary $Q \times R$ matrix. Using a binary dilation operation, cf. [GW02, Sec. 9.2.1], the ones can be extended into the local neighborhood according to a structuring element. Afterwards, NMS is applied to the re-ranked patches with $\zeta = 2$ in order to only retrieve patches that do not overlap with each other. Provided that P patches are selected for re-ranking with the Viterbi algorithm, the computational complexity for two-stage decoding with BoF-HMMs is $O(S\hat{M}_{HMM}\hat{M}_{IFS} + PF\hat{M}\hat{S}^2)$. Due to soft NMS and depending on the size of the re-ranking mask, $P \ll LR$ where LR is the number of patches in the patch-based framework, cf. Section 4.6.2. Thus, with respect to computational efficiency the two-stage decoding approach can be seen as a trade-off that makes highly accurate decoding with the Viterbi algorithm feasible.

In order to obtain document image regions for the retrieval list, the indices for locally optimal patches in the matrix $[\nu_{qr}]$ are transformed to image coordinates based on the patch sampling steps and the image coordinate offsets. The offsets are defined by the patch with minimum image coordinates on the document image. The patch height is given by $\mathfrak{p}_h$. The width is based on the BoF vector indices that are associated with the states from the *query word* HMM in the compound query model. Figure 33 visualizes the two-stage integration from patches retrieved with mixture component voting to regions retrieved after re-ranking with Viterbi-based query word decoding.

It can be noted that Viterbi decoding produces very similar scores for patches that horizontally overlap with document image regions that are visually similar to the query. This is due to the scores that represent the most likely occurrence of the query within the patch and not the entire patch. Retrieval with NMS is only unambiguous if the maximum patch score within the maximum filter mask is unique. Redundant detections can be avoided by applying soft NMS to patch score matrix $[\nu_{qr}]$. In contrast to the application to matrix $[\bar{\nu}_{qr}]$, the Gaussian filter is considerably smaller, i.e., the same size as the dilation filter mask.

The meta parameters affecting the two-stage decoding process include the visual-word grid sampling step $\mathfrak{g}$, the number of mixture components M in the BoF output model and the number of HMM states S in the query model, cf. Section 5.3.9. The patch sampling rate υ, introduced in Section 4.6.1, needs to be sufficiently high such that there exist patches that are approximately vertically centered over the words in the document image. In horizontal direction the sampling rate is not as important since the location of the query word within the patch is inferred with patch-based Viterbi decoding. For this purpose, the patch width is extended according to ψ such that also larger occurrences of the query word can be detected, cf. Section 5.3.8. The probability threshold ϵ_{low} is required for keeping representations sparse, thus, avoiding very small probabilities that affect the computations only marginally. Threshold ϵ_{floor} is mostly important during Viterbi decoding. Otherwise, it is useful for visualizing scores in the logarithmic domain. This way, users obtain a more detailed impression of similarity in addition to the order of regions in the retrieval list, cf. Figure 31 and also 33.

4.6.5 *Region snapping*

Setting the patch size implies an assumption about the size of words that are relevant to the query. While word spotting is most accurate if the assumption is valid, good generalization capabilities of the BoF representations allow for detecting words that are smaller as well as larger than the patch size. However, the localization of the relevant

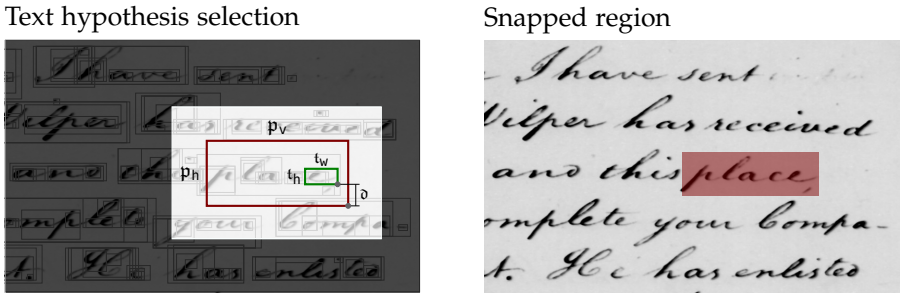


Figure 34: Region snapping. The figure visualizes the refinement of regions that are obtained after patch-based Viterbi decoding. Region bounds are snapped to text hypothesis bounds for this purpose. On the left, region p is shown in red. Text hypotheses that are relevant for snapping are completely localized within the search area with size (a_w, a_h), which is shown as a white overlay centered around p. Each bound of region p is snapped to the corresponding bound of a relevant text hypothesis. The bound selection is based on a size similarity indicator and a proximity indicator. For a visualization of the indicators, text hypothesis t is shown in green. Size similarity is indicated by labeling width p_w and height p_h of region p as well as width t_w and height t_h of text hypothesis t. Bound proximity is indicated by distance ∂ between the y-coordinates of the lower-right corners of p and t. It is important to note that the lower bound of p will only be snapped to the lower bound of a relevant text hypothesis. On the right, the result after region snapping is shown. The region fits the word boundary tightly.

document image regions cannot be expected to be as accurate anymore. Due to Viterbi-based decoding of the query word within the patch, this is much more critical in vertical direction than in horizontal direction. If the word spotting application requires regions that fit the relevant words tightly, the regions obtained after Viterbi decoding can be snapped to text hypotheses, cf. Section 4.2.1.

For this purpose, the region that should be refined will be denoted as p in the following. Region p defines a search space in the document image in order to obtain text hypotheses that are relevant for snapping. Provided that p_v is the width after Viterbi decoding and p_h is the patch height, the search space is a rectangular area centered around p with width and height (a_w, a_h) as defined in Equation 67.

$$a_w = p_v + \eta_{\text{size}}\, p_h \quad \text{and} \quad a_h = p_h + \eta_{\text{size}}\, p_h \tag{67}$$

Thus, the size of this search area is relative to the patch height scaled by parameter $\eta_{\text{size}} \in \mathbb{R}_{>0}$. In order to retrieve hypotheses fast, the upper-left and lower-right corners of all text hypotheses are indexed with respect to their (x, y) coordinates, cf. [ZE02]. For each coordinate, the index contains a list of coordinate values sorted in ascending order. Search intervals for x and y coordinates are defined by the upper-left and lower-right corners of the search area. Text hypothe-

ses that have their upper-left or lower-right corners within the search intervals for the corresponding coordinates are retrieved efficiently with binary searches. Text hypotheses that are relevant for snapping have *all* corner coordinates within the search intervals. Figure 34 (left) shows region $\mathfrak{p}$ that is to be snapped to text hypotheses. It is centered in the search area which is used to obtain text hypotheses for snapping.

Each bound, i.e., left, right, top and bottom, of region $\mathfrak{p}$ is snapped to the corresponding bound of a relevant text hypothesis. Corresponding means that, for example, the left bound of region $\mathfrak{p}$ is only snapped to a *left* bound of a text hypothesis. It is important to note that each bound may be snapped to a bound of a different hypothesis. The text hypothesis bounds are selected based on a size similarity indicator and a bound proximity indicator. While the size indicators are computed for relevant hypothesis, the proximity indicators are computed for relevant hypothesis bounds. Both indicators have to be represented in the same value range such that they can be incorporated in a single value measure.

Since text hypotheses are based on contrast, cf. Section 4.2.1, they might represent arbitrary text components and also background clutter. Therefore, text hypotheses that represent (parts of) the query word should have a width that is similar to $\mathfrak{p}_v$ or a height that is similar to $\mathfrak{p}_h$ or both. In order to measure size similarity, relative width and height differences are recorded separately. Similarity values are scaled differences between 0 and 1 such that the largest width difference and the largest height difference are represented by similarity value 0. For each relevant text hypothesis a single size similarity indicator is obtained as the maximum of the width and height similarity values. The size similarity indicator will be denoted as $\varphi_{\text{size}} \in [0, 1]$. Figure 34 (left) shows text hypothesis $\mathfrak{t}$. The size similarity criterion is indicated by labeling width and height of region $\mathfrak{p}$ and text hypothesis $\mathfrak{t}$.

Proximity indicators are computed for each bound of region $\mathfrak{p}$ with respect to all corresponding bounds of relevant text hypotheses. The four bounds are represented by the (x, y) coordinates of the upper-left corner and the (x, y) coordinates of the lower-right corner. For each of the four coordinates, the distances between the coordinate value for region $\mathfrak{p}$ and the values of corresponding coordinates for all relevant text hypotheses are computed. The proximity indicators are scaled distances such that the smallest distance has the largest proximity value 1 and the largest distance is represented by 0. The proximity indicator will be denoted as $\varphi_{\text{prox}} \in [0, 1]$. Figure 34 (left) indicates distance $\mathfrak{d}$ between the lower-right y-coordinates of region $\mathfrak{p}$ and text hypothesis $\mathfrak{t}$.

Based on size indicator φ_{size} that is obtained for each relevant text hypothesis and proximity indicator φ_{prox} that is obtained for each rel-

evant hypothesis bound, a joint indicator is computed. The joint indicator is used in order to select the text hypothesis bounds that the region bounds will be snapped to. For this purpose, the F-score between φ_{size} and φ_{prox} is considered. In order to parameterize the measure with respect to the influence of the indicators, the generalized $F_{\eta_{\text{weight}}}$-score, cf. [BR11, p. 328], is defined in Equation 68. For $\eta_{\text{weight}} < 1$, meta parameter $\eta_{\text{weight}} \in \mathbb{R}_{>0}$ emphasizes φ_{prox}. For $\eta_{\text{weight}} > 1$, η_{weight} emphasizes φ_{size}, see Section 5.3.10.

$$F_{\eta_{\text{weight}}} = (1 + \eta_{\text{weight}}^2) \frac{\varphi_{\text{prox}}\, \varphi_{\text{size}}}{\eta_{\text{weight}}^2 \varphi_{\text{prox}} + \varphi_{\text{size}}} \tag{68}$$

Essentially, the $F_{\eta_{\text{weight}}}$-score is a generalized form of the harmonic mean between two numbers. In the given scenario the measure is useful because it is dominated by the smaller of the two values. This is due to the product in the numerator of Equation 68. Thus, text hypothesis bounds are preferred that have high indicator values for proximity *and* size. After a hypothesis bound has been selected for each region bound, region $\mathfrak{p}$ can be refined to new coordinates as shown for the occurrence of query word `place` in Figure 34 (right).

The meta parameter η_{size}, which controls the size $(\mathfrak{a}_w, \mathfrak{a}_h)$ of the search area around $\mathfrak{p}$, and the meta parameter η_{weight} have to be adjusted according to the expected size variability of the words in the document. If the size variability is large, η_{size} has to be adjusted according to the maximally expected deviation, e.g., $\eta_{\text{size}} = 1$ covers word occurrences that are three times as high as the patch height, see Equation 67. Furthermore, size indicator φ_{size} becomes more important with increasing values of η_{size}. This is due to the increasing influence of background clutter and text components that are not part of the query word. The influence of φ_{size} can be adjusted with η_{weight}.

4.7 SUMMARY

Segmentation-free word spotting with BoF-HMMs allows for supporting the analysis of document images in order to explore a document collection. This is achieved by combining different information sources that are available during the exploration process.

Starting with the document images, BoF representations are adapted to the document collection in an unsupervised manner. These powerful representations allow for high word spotting accuracy even if only a single annotated example is available in the query-by-example scenario. Given an exemplary occurrence of the query word, the document region is represented with a sequence of BoF vectors. Similarity is based on an optimal alignment of the sequence with the compound query model. This is particularly important for coping with typical writing variabilities. The BoF-HMM can be seen as a dynamic probabilistic extension of a spatial pyramid.

Different approaches for modeling BoF representations in the stochastic HMM process have been presented:

- The vMF mixture model is suitable for representing *directional* data. It is relevant due to the wide use of cosine similarity with BoF representations.
- The EDCM mixture model is suitable for representing *sparse and high dimensional* BoW vectors and has been used for document clustering and classification. It is relevant due to the similar characteristics of BoF vectors in the given scenario.
- The visual words mixture model interprets BoF vectors as *distributions* over visual words. Therefore, it models the occurrence of individual visual words instead of the occurrence of a BoF vector. It is relevant because the relaxed modeling allows for very good *generalization* capabilities for word spotting.

In addition to the visual appearance of the query word, its size is used for defining the patch size in the patch-based framework. Although size is an important indicator for spotting words, word size variabilities cannot be handled well if the patch size is fixed for a query. Extending the search to different patch sizes per query quickly becomes infeasible due to the large number of patches [RAT+15a].

By integrating the patch-based framework with text hypotheses, the presented approach combines advantages of word spotting methods that are either based on patches, line segments or word regions. For this purpose, text hypotheses are used in order to obtain line hypotheses and whitespace hypotheses in a bottom-up manner. By using ERs for text detection, the dependence on a single binarization threshold is avoided. This improves the robustness for processing document images with degradations such as varying contrast. Line region representations can be precomputed since they are independent of the query. The number of possible line regions is considerably lower than the number of possible patches. If patches are decoded within the line hypotheses, the search can be constrained to document image regions that contain text hypotheses with a height that is similar to the height of the query. Even more importantly, the approach is related to HMM-based word spotting on line level. Instead of modelling the occurrence of the query in a line, the approach models the occurrence of the query in a patch. For this purpose, the context is represented with a background HMM that represents the statistical feature distribution as well as whitespace HMMs. The patch width is adapted to the potential width of the query. The patch similarity score is given by the geometric mean of the partial path probability that has been aligned with the query word model. This way, word size variabilities can be handled while the scores are specific to the spatial patch locations at the same time. If the localization of the potentially relevant words needs to be refined further, regions can be

snapped to text hypotheses. For these reasons, the presented method has advantageous properties of:

- patch-based approaches, because it uses *size* information and refines patches based on similarity with the query,
- line-based approaches, because it decodes the most likely occurrence of the query *within* a patch with a sequence model,
- approaches using word regions, because text, line and whitespace hypotheses are obtained *independently* of the query.

Furthermore, any annotated document regions that become available during the exploration of a document collection, can directly be incorporated within model estimation. In this regard, HMMs offer the advantage that regions do not have to be annotated at the level of the elementary modeling units. Therefore, the extension to query-by-string is possible with standard approaches while the decoding framework remains the same as for query-by-example word spotting. This allows for an easy transition from query-by-example to query-by-string without changing the word spotting method in general. However, using the same decoding framework for query-by-example and query-by-string requires that the characteristics of the document collection are similar in both cases. This refers to the visual variability in the appearance of text.

- The use of a patch-based framework restricts the search to a single patch height.
- The output probability represents the visual appearance of the document image regions and *not* a query posterior probability.
- The output model is not optimized jointly with the HMM.

These aspects *cannot* be considered as limitations if only a single word image is available for model estimation. However, due to these aspects, the writing style variability that BoF-HMMs can cope with will be limited with respect to the writing style variability that might be covered in a large dataset of annotated training samples. Therefore, query-by-string word spotting is intended as an extension to the query-by-example scenario where no annotated training material is available besides the query word image.

Finally, computational efficiency is an aspect that all segmentation-free word spotting methods are concerned with. Standard indexing approaches and approximate similarity measures are not directly applicable to HMMs. For HMM-based word spotting, graph representations have been used in order to retrieve words from precomputed n-best recognition results fast. Mixture component voting is rather inspired from feature matching approaches that are efficient due to indexing with inverted indices. Since BoF-HMMs are semi-continuous

Table 3: BoF-HMM characteristics

Characteristic	BoF-HMM
Output model	BoF mixture model
Retrieval	Partial output probability
Score normalization	Query length
Segmentation	None
Hypotheses	Line hypotheses, patches and mixture component voting
Efficiency	Mixture component voting based on IFS
Selection	NMS of patch similarity scores
Refinement	Patch-based alignment and region snapping

(SC-HMMs), the output model is independent of the query model and can be precomputed for all frames in all line hypotheses. Frames are represented by mixture components. By indexing the frames according to mixture component indices, the frames that are relevant for the query model can be retrieved rapidly through an IFS. Mixture component voting considers the sequential state structure of the query word HMM. Matching components vote with respect to a reference point which is defined with respect to the query word model. The accumulator cells in the voting space represent sums for the evaluation of the mixture model under the voting scheme. This way, the method is sensitive to document regions that are visually similar to the query, allowing for high recall at the cost of reduced precision. This is compensated by re-ranking patches with the computationally more expensive Viterbi decoding.

Table 3 summarizes the important characteristics of the proposed method. This also includes the aspects that have been discussed with respect to segmentation-free word spotting (Table 1) and HMM-based word spotting (Table 2) in Section 3.4. The comparison of the three tables highlights the novelties of word spotting with BoF-HMMs:

- No other HMM-based word spotting method has been applied on document-level without a segmentation on word-level or on line-level.
- The study of the BoF output models for SC-HMMs is new.
- Decoding the most likely occurrence of the query *within a patch* is new.
- The mixture component voting algorithm for SC-HMMs is new.

That these novelties also lead to outstanding word spotting performance will be evaluated in Chapter 5.

5 EVALUATION

Segmentation-free word spotting with BoF-HMMs is evaluated on different word spotting benchmarks in order to analyze the components of the method in detail. Furthermore, the method's performance is compared to the performance of related methods from the literature. For this purpose, the following sections present:

- performance measures for the quantitative analysis of word spotting methods which will be used in the word spotting benchmarks (Section 5.1),
- benchmark datasets of handwritten historic document images including evaluation protocols for query-by-example and query-by-string word spotting (Section 5.2),
- a detailed optimization of the meta parameters that shows how the different components of the method affect retrieval performance and computational efficiency (Section 5.3),
- a comparison to a baseline method which uses a temporal adaptation of a spatial pyramid in a patch-based word spotting framework. A comparison to the related methods from the literature shows that the proposed method outperforms all other methods by a large margin in the query-by-example scenario where no annotated training material is available besides the query word image (Section 5.4).

Retrieval efficiency is evaluated in terms of computational complexity and memory requirements. Runtime measurements are performed in order to demonstrate the applicability in practice. For this purpose, the word spotting method is implemented in an integrated software framework in *Python*. Runtime critical components are implemented in *C++*. The most important software libraries include the HMM toolkit *ESMERALDA* [FP08] and the computer vision toolkit *VLFeat* [VF08].

In order to demonstrate the applicability of the method to different datasets, query-by-example results are reported for five benchmarks that have been used in the word spotting community before. Query-by-string performance is reported for three benchmarks that are defined on datasets which are used in the query-by-example scenario as well. Furthermore, the applicability of the method to different scripts is demonstrated on one out of these three datasets. The documents in this dataset are written in German *Kurrent* script while the documents from all other datasets are written in English *Latin* script.

5.1 PERFORMANCE MEASURES

The evaluation of segmentation-free word spotting with BoF-HMMs, see Chapter 4, is based on a qualitative and a quantitative analysis. The qualitative analysis gives an *impression* of the performance by considering individual example cases. The quantitative analysis measures the *average case* performance by considering a larger number of examples. Since the *impression* that is received in the qualitative analysis is strongly dependent on the selection of the example cases, a comparison between different systems, i.e., different methods or different meta parameterizations of the same method, should be made with *quantitative* performance measures. In order to draw conclusions that lead to optimal performance in practice, it is required that the diversity of the examples in the quantitative analysis is *representative* for the real world scenario. For the quantitative evaluation of BoF-HMMs for segmentation-free word spotting, a single value measure is used (Section 5.1.1). Whether the performance difference, that is observed between different method parameterizations, can be considered as *significant* is measured with a statistical test (Section 5.1.2).

5.1.1 *Mean average precision*

The quantitative analysis of a word spotting method typically measures if all occurrences of the query word are present in the retrieval list as well as the order of the relevant detections in the list. The most common measure which incorporates both of these criteria into a single value is the average precision. The *mean average precision* (mAP) is obtained as the mean of the *average precision* values for a set of queries, cf. [BR11, Sec. 4.3.2]. For this purpose, the set of queries and an annotated set of document images is required. For an evaluation on word-level, the annotations consist of bounding boxes and labels which indicate each occurrence of a word from the query set, cf. Section 5.2.

Given a query, a segmentation-free word spotting system automatically detects *potentially relevant* document regions and *ranks* these regions in a retrieval list. In order to measure the performance, the *relevance* of these regions must be determined. In the segmentation-free scenario, an element is considered as relevant if the detected document image region overlaps with a relevant bounding box from the dataset annotations by more than a given threshold, see Figure 35. A bounding box annotation is relevant to the query if it is labeled with the query word.

For this purpose, the retrieval list with K elements is given as an answer set $\Phi = \{\Phi_k \mid 0 \leqslant k < K\}$ where Φ_k denotes the set of image pixels of the document region at retrieval list index k. In analogy, the set of R document image regions that are *relevant* to the corre-

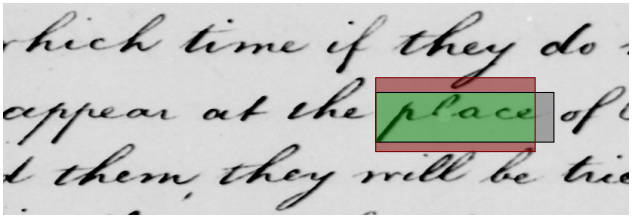


Figure 35: Intersection over union (IoU). The detected region is shown with a red frame and a region from the dataset annotations is shown with a black frame. The intersection of the regions is shown in green. The union of the regions includes the areas shown in green, red and gray. The IoU is the fraction of the sizes of the intersection area and the union area. A detection from the retrieval list is considered as relevant with respect to the query if the IoU exceeds a given threshold.

sponding query for *all* document images in the dataset is given by $\Psi = \{\Psi_r \,|\, 0 \leqslant r < R\}$. Thus, according to the dataset annotations, R is the *total* number of relevant regions for the query in the dataset. The relevance of the detected document image region at retrieval list index k can then be expressed by function $\varphi : \{0, \ldots, K-1\} \to \{0, 1\}$ as defined in Equation 69. The overlap threshold ι is based on the *intersection over union* (IoU) of two document image regions, cf. Figure 35, typically $\iota = 0.5$, cf. e.g., [PZG+16; GSG+17].

$$\varphi(k) = \begin{cases} 1 & : \quad \exists \Psi_r \in \Psi : \frac{|\Phi_k \cap \Psi_r|}{|\Phi_k \cup \Psi_r|} > \iota \\ 0 & : \quad \text{otherwise} \end{cases} \tag{69}$$

Based on the relevance function $\varphi(k)$ the average precision can be defined. The average precision corresponds to the area under the precision-recall curve. Precision is the proportion of relevant and non-relevant elements in the retrieval list. Recall is the proportion of relevant elements in the retrieval list and the total number of relevant elements in the dataset. The key idea for evaluating the ranking of relevant elements in the retrieval list is to consider the precision for different recall levels. This incorporates the ranking since different recall levels are obtained for different lengths of the retrieval list. For this purpose, Equation 70 defines precision $\pi : \{1, \ldots, K\} \to [0, 1]$ and recall $\rho : \{1, \ldots, K\} \to [0, 1]$ as functions of the number of top-n elements of the retrieval list, cf. [PTV15b].

$$\pi(n) = \frac{\sum_{k=0}^{n-1} \varphi(k)}{n}, \quad \rho(n) = \frac{\sum_{k=0}^{n-1} \varphi(k)}{R} \tag{70}$$

It can be noted that for increasing lengths n of the retrieval list, $\rho(n)$ changes with every additional *relevant* element. Since the *average precision* (AP) averages precision values at different recall levels, Equation 71 defines AP such that the precision at list length $n = k + 1$ is incorporated at every *relevant* element with index k, thus, whenever

the recall changes. By normalizing the sum with R, i.e., with the total number of relevant elements in the dataset, all relevant elements that are *not* included in the retrieval list contribute to the average with precision zero. In Equation 71, it should be noted that $\pi(n)$ is a function of the list *length* and $\varphi(k)$ is a function of the list *index*.

$$\mathrm{AP} = \frac{1}{R} \sum_{k=0}^{K-1} \pi(k+1)\,\varphi(k) \tag{71}$$

Thus, average precision measures the *ranking* and the *completeness* of the retrieval list with respect to relevant occurrences of the query word. Both criteria are important for using word spotting in practice. Within an evaluation over a larger set of queries, the average precision values AP_q obtained for each query q are averaged according to Equation 72 resulting in *mean average precision* (mAP).

$$\mathrm{mAP} = \frac{1}{Q} \sum_{q=0}^{Q-1} \mathrm{AP}_q \tag{72}$$

In addition to evaluating word spotting performance with a single value measure, the *interpolated precision-recall curve*, cf. [BR11, Sec. 4.3.1], allows for analyzing the precision at different recall levels specifically. The interpolation is required in order to compute a single precision-recall curve for a larger set of queries. The interpolated precision $\overline{\pi} : \{0.0, 0.1, \ldots, 1.0\} \to [0, 1]$ allows for obtaining precision values at 11 standardized recall levels, cf. [BR11, p. 137], which are specified by the domain of function $\overline{\pi}(l)$. The *maximum* interpolation over all *larger* recall values, as shown in Equation 73, is useful in order to allow for an interpolated recall level 0.0 [BR11, p. 136]. The mean interpolated precision-recall curve $\mathrm{mIP}(l)$ over all queries is obtained in analogy to Equation 72.

$$\mathrm{mIP}(l) = \frac{1}{Q} \sum_{q=0}^{Q-1} \overline{\pi}_q(l), \quad \overline{\pi}(l) = \max_{m:\, l \leqslant \rho(m)} \pi(m) \tag{73}$$

The maximum interpolation can also be used for defining *average interpolated precision* (AIP), cf. [PTV15b]. AIP can be seen as an approximation of the area under the *interpolated* precision-recall curve. For this purpose, function $\widetilde{\pi} : \{1, \ldots, K\} \to [0, 1]$ is defined in Equation 74. In contrast to function $\overline{\pi}(l)$, the function domain is based on the *length* of the retrieval list and not on the recall level. The recall levels result from the retrieval list directly, as for AP in Equation 71. In analogy to the mAP, the mAP_{ip} is defined in Equation 75.

$$\mathrm{AIP} = \frac{1}{R} \sum_{k=0}^{K-1} \widetilde{\pi}(k+1)\,\varphi(k), \quad \widetilde{\pi}(n) = \max_{m:\, \rho(n) \leqslant \rho(m)} \pi(m) \tag{74}$$

$$\mathrm{mAP}_{ip} = \frac{1}{Q} \sum_{q=0}^{Q-1} \mathrm{AIP}_q \tag{75}$$

In the following, mAP will be the main performance measure. The mIP curve will be used in order to analyze precision at different recall levels. This is useful for distinguishing the performance at the beginning or at the end of the retrieval list. Finally, mAP_{ip} is required for comparing performance with other methods from the literature. It should be noted that the difference between mAP and mAP_{ip} is typically small and $\text{mAP} \leqslant \text{mAP}_{\text{ip}}$.

5.1.2 *Permutation test*

For a comparison of two systems, the mAP is evaluated for both systems on the same benchmark, i.e., using the same set of queries, the same document images and the same evaluation protocol, cf. Section 5.2. However, since the mAP is an average case performance measure, it is unclear whether the difference in the average case is due to large differences in the performance of few queries or if the differences can be observed for the majority of the queries. In the latter case, the difference is more likely to be due to the characteristics of the systems rather than due to the characteristics of the benchmark. In order to support this interpretation of the results, significance tests help with a statistical analysis of the differences.

The permutation test, cf. [ET98, Chap. 15], is a significance test which is suitable for the analysis of retrieval systems [SAC07] and has also been used for analyzing word spotting performance [SF18]. In [SAC07], the permutation test was compared to different significance tests that can be used for measuring the mAP difference of two information retrieval systems. Among these tests is the *paired t-test* which is generally common for testing the difference of means, cf. [Coh95, Sec. 4.4]. The permutation test and the paired t-test showed similar performance in the experiments in [SAC07]. Nevertheless, the permutation test is preferred since it is non-parametric and avoids the assumptions that come along with parametric models, such as the normal distribution which is involved in the t-test. This makes the permutation test more robust with respect to outliers, especially if the number of samples is small [SAC07], also cf. [Coh95, Sec. 5.5]. According to the motivation for applying a significance test in the following, such a scenario can be considered as very important.

The permutation test requires a test statistic, a null hypothesis and a significance level. Provided that the performance measure is mAP, the *test statistic* for a two-sided test is given by the absolute difference of the means of the average precision values obtained for the two word spotting systems [SAC07], as defined in Equation 77.

The *null hypothesis* is that the systems are identical. No other assumptions are required, e.g., about the distribution of the test statistic. If the null hypothesis is true, it is assumed that the absolute difference of the means is *not* representative for the *typical* absolute difference

of the average precisions of the queries. In practice this means that the two word spotting systems do not behave significantly differently for the benchmark considered. Consequently, the difference in mAP is not suitable for drawing any *definitive* conclusions with respect to the performances on the benchmark. For example, whether to choose one meta parameter configuration over the other.

The *significance level* is directly given as the probability for obtaining an absolute mAP difference that is greater than or equal to the *observed* absolute mAP difference *under the null hypothesis*. Intuitively, the test estimates how likely the absolute difference between the mAP values is under the assumption that the two systems are identical. This probability is known as the *p-value*. If the p-value falls below (or is equal to) a threshold, typically $\alpha = 0.05$, the null hypothesis is rejected and it is assumed that the benchmark results differ significantly. The smaller the value of α the less likely are *false positives* where the null hypothesis is rejected although it is actually true.

In order to test the null hypothesis, it is assumed that all 2Q average precision values that have been obtained for Q queries and two word spotting systems, could also have been computed by a *single* system [SAC07]. For this purpose, the values of the two systems are written in a vector $\mathbf{a} \in \mathbb{R}^{2Q}$ where AP_{0q} denotes an average precision value of the first system and AP_{1q} denotes an average precision value of the second system for $q \in \{0, \ldots, Q-1\}$ as shown in Equation 76.

$$\mathbf{a} = (\mathrm{AP}_{00}, \ldots, \mathrm{AP}_{0,Q-1}, \mathrm{AP}_{10}, \ldots, \mathrm{AP}_{1,Q-1})^\top \tag{76}$$

Based on vector $\mathbf{a}$, the absolute mAP difference that has originally been observed between the two systems can be computed with function $d_{\mathrm{mAP}} : \mathbb{R}^{2Q} \to \mathbb{R}_{\geqslant 0}$, see Equation 77. This is the test statistic.

$$d_{\mathrm{mAP}}(\mathbf{a}) = \left| \frac{1}{Q} \sum_{r=0}^{Q-1} a_r - \frac{1}{Q} \sum_{s=Q}^{2Q-1} a_s \right| \tag{77}$$

Under the null hypothesis and since the *mean* of the average precisions is considered in the test statistic, the values in the first half of vector $\mathbf{a}$ can arbitrarily be exchanged with the values in the second half of vector $\mathbf{a}$ [ET98, Sec. 15.2]. If the absolute mAP differences obtained for the permuted versions of $\mathbf{a}$ will typically be smaller than the absolute difference for the original vector $\mathbf{a}$, this is an indication that the majority of the query results for the two systems are typically different to each other. Exchanging numbers from two sequences with very different means will make the means more similar. Furthermore, the larger the absolute mAP difference for the original vector $\mathbf{a}$, the less likely it is to obtain even lager absolute mAP differences if values are exchanged between the sequences.

Statistically this is modelled with the help of set $\mathfrak{P}$ which contains $\binom{2Q}{Q}$ permutations of vector $\mathbf{a}$ [ET98, p. 208]. The number of permu-

tations considered is the number of possibilities for drawing Q elements from $\mathbf{a}$ without replacement while the order of the elements in $\mathbf{a}$ is disregarded. According to the null hypothesis, all permutations are equally likely [SAC07]. Consequently, the p-value can be computed as the proportion of permutations $\mathbf{a}_{\text{perm}} \in \mathfrak{P}$ with an absolute mAP difference $d_{\text{mAP}}(\mathbf{a}_{\text{perm}})$ which is greater than or equal to the original absolute mAP difference and the total number of permutations [ET98, p. 208], as shown in Equation 78.

$$p = \frac{\left|\left\{\mathbf{a}_{\text{perm}} \in \mathfrak{P} \mid d_{\text{mAP}}(\mathbf{a}) \leqslant d_{\text{mAP}}(\mathbf{a}_{\text{perm}})\right\}\right|}{|\mathfrak{P}|} \tag{78}$$

Due to the large size of $\mathfrak{P}$, the exact p-value will be approximated in practice. For this purpose, only a random subset $\mathfrak{P}' \subset \mathfrak{P}$ is considered. The computational effort and the accuracy of the estimate $\hat{p}$ can be controlled through the size of the subset. A measure for the accuracy is based on the variance of the estimator [ET98, pp. 208–210].

Provided that $B = |\mathfrak{P}'|$ denotes the size of the subset and considering the estimator $\hat{p}$ as a random variable, $B \cdot \hat{p}$ follows a binomial distribution with B independent experiments and success probability p, [ET98, p. 208]. In this regard, it is important to note that the success probability of the binomial distribution is the *exact* p-value. This can be seen when substituting $\mathfrak{P}'$ for $\mathfrak{P}$ in order to obtain the definition of $\hat{p}$ in Equation 78. The size of the set in the numerator still depends on the *exact* success probability since the success for the elements in $\mathfrak{P}$ only depends on the original absolute mAP difference $d_{\text{mAP}}(\mathbf{a})$ and not on the other permutations. Based on the definition of the *expected value* $E[B\hat{p}] = Bp$ and the *variance* $\text{Var}[B\hat{p}] = Bp(1-p)$ of the binomial distribution, cf. [Bis06, Sec. 2.1], the variance of the estimator $\text{Var}[\hat{p}]$ is derived in Equation 79 and 80.

$$\begin{aligned} \text{Var}[B\hat{p}] &= E[(B\hat{p})^2] - E[B\hat{p}]^2 \\ &= B^2 E[\hat{p}^2] - \left(B\,E[\hat{p}]\right)^2 = B^2 \text{Var}[\hat{p}] \end{aligned} \tag{79}$$

$$\Leftrightarrow \text{Var}[\hat{p}] = \frac{\text{Var}[B\hat{p}]}{B^2} \tag{80}$$

Equation 79 follows from the properties of the expected value, cf. [DHS00, Sec. A.4.2]. After substituting the definition of the variance for a binomial distribution in Equation 80, the number of permutations which is required for a given accuracy of the estimated p-value is obtained in Equation 81.

$$\text{Var}[\hat{p}] = \frac{p(1-p)}{B} \Leftrightarrow B = \frac{p(1-p)}{\text{Var}[\hat{p}]} \tag{81}$$

Since the *exact* p-value is unknown in practice, a worst case estimate for the number of permutations is computed for $p = 0.5$. Following [SF18], a highly accurate estimator with a variance of $\text{Var}[\hat{p}] \leqslant 10^{-6}$, or a standard deviation of $\sigma_{\hat{p}} \leqslant 0.001$, is obtained for $B = 250,000$ random permutations.

(a) G. Washington (GW20) (b) Bentham 2014 (BT50) (c) Bentham 2015 (BT70)

(d) Konzilsprotokolle (KP20) (e) Botany (BO20)

Figure 36: Benchmark datasets. The George Washington dataset (a) is widely used in the word spotting community. The visual variability in the 20 document images in the benchmark is limited. Two document collections of the Bentham dataset, (b) and (c), have been considered in the word spotting competitions at the *ICFHR* 2014 and *ICDAR* 2015. Both collections exhibit large word size and writing style variabilities. The 2014 benchmark contains 50 pages and the 2015 benchmark contains 70 pages. The Konzilsprotokolle (d) and Botany (e) datasets have been used for the word spotting competition at the *ICFHR* 2016. Each benchmark contains 20 pages. While the writing style in the Konzilsprotokolle benchmark is rather homogeneous, the document images from the Botany benchmark contain the largest writing style variabilities among the benchmarks considered.

5.2 BENCHMARK DATASETS

Word spotting benchmarks are defined by a set of document images, a set of annotated document regions including occurrences of the query words and an evaluation protocol. The evaluation protocol describes which query words are searched on which document images and how the quantitative performance is measured, cf. Section 5.1.

The targeted scenario for the word spotting method presented in Chapter 4 is the exploration of a document collection where no or only little annotated training material is available. This is a typical scenario for historians that start to explore a document collection. In order to simulate the exploration process, a set of document images for searching query words is defined. The occurrence of each query word is annotated for the quantitative evaluation. These queries are given by a set of query word images for the query-by-example scenario. The query-by-string scenario requires a set of training annotations and a list of query strings. The characteristics of the document

images and the selection of queries and annotations is crucial for the word spotting performance that will be measured. For this reason, a quantitative performance measure can only be interpreted *in comparison* to other methods, or to parameterizations of the same method, that have been evaluated on the *same* benchmark. Thus, benchmark datasets should have been considered in the word spotting literature before. The benchmarks that will be presented in Section 5.2.1 to 5.2.3 have been selected according to both criteria, i.e., relevance with respect to the targeted scenario and utilization in the word spotting community. Figure 36 shows an overview of the five datasets.

5.2.1 *George Washington letters*

The George Washington dataset is the most widely used benchmark dataset for evaluating word spotting methods, cf. [GSG+17]. The document images originate from the *George Washington papers* collection at the *Library of Congress*, Washington DC, USA [Was54]. The entire collection contains over 65,000 documents and is divided into 9 series. The document images in the word spotting benchmark come from *Series 2, Letterbooks 1754 to 1799: Letterbook 1*. The letterbooks contain copies of Washington's mail and have been written by George Washington and his secretaries in the 18th century. *Letterbook 1* is not an original but a later re-copy of Washington's correspondences, cf. [SF18]. Thus, the writing style in the document images from the benchmark is very homogeneous. Small document image sections from this dataset have been used throughout the previous chapters. A larger document section is shown in Figure 36a.

In the context of word spotting, the George Washington letters have been described in [KLP01] for the first time. Twenty document images along with 4,860 bounding box annotations on word level have been used in [LRM04] and are available at the *University of Massachusetts*[1]. The protocol that will be used for evaluating segmentation-free query-by-example word spotting without annotated training material has been described in [RAT+11]. All annotations are converted to lowercase characters and all punctuation marks are removed. Each of the 4,860 word image regions is used as a query and all occurrences of the query are searched on all 20 pages. Therefore, the query image region will typically be the first element in the retrieval list.

For evaluating the query-by-string word spotting performance on the George Washington dataset, a cross validation is performed in analogy to [ART+13]. For this purpose, the 20 document images are split in four cross-validation folds of five pages such that the first fold contains the pages one to five, the second fold contains the pages six to ten and so on. For each fold, the query set is given by the words

1 `http://ciir.cs.umass.edu/downloads/old/data_sets.html`, accessed on July 13, 2019.

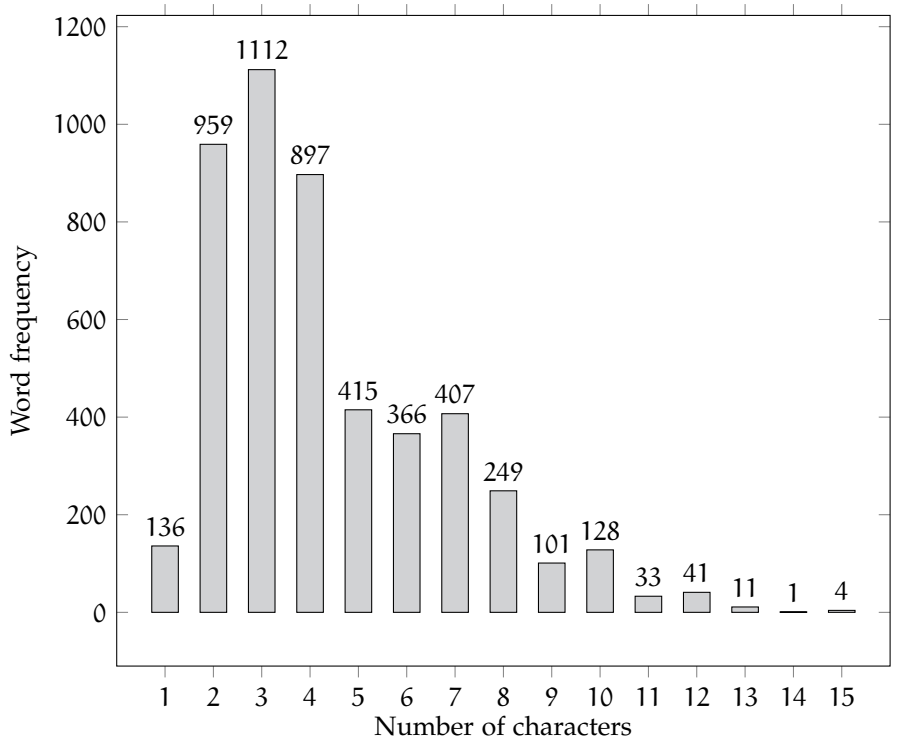


Figure 37: Word frequencies by length in the George Washington dataset. The histogram is computed over all 4,860 word instances in the dataset. The word frequencies correspond to the query word frequencies in the query-by-example scenario.

that are occurring at least once in any of the five pages in the fold. This results in 435, 424, 521 and 431 query words for the four folds. The lexicon over all 20 pages contains 1,124 words. The 15 pages that are not among the five pages in a fold constitute the training set of the fold. The training sets for the four folds contain 3,639, 3,677, 3,568 and 3,696 word-level annotations. In the segmentation-free scenario considered in the following, the five document images in a fold are searched for the occurrences of the query words. The quantitative results are given as the weighted average of the results obtained for each fold where the weight is the relative number of queries per fold. A variant of this cross-validation is considered in order to analyze the required amount of training annotations. For this purpose, only five pages are used in a training set and the remaining 15 pages are used for retrieval. In analogy to the *15-5* cross-validation, this *5-15* cross-validation uses 1,221, 1,183, 1,292 and 1,164 word-level training annotations per fold. The numbers of query words per fold are 965, 961, 846 and 965.

In the query-by-example scenario and in the query-by-string scenario, a detection is considered as relevant if it overlaps with a bounding box which is annotated with the query word by more than 50% IoU ($\iota = 0.5$). The main performance measure is mAP, cf. Section 5.1.1.

The major challenges for segmentation-free word spotting on the George Washington benchmark are considerable contrast variations of the pen stroke, cf. Figure 36a, and short words in the query sets. The contrast variations are most problematic for methods that rely on binarization. The short query words are a challenge in general. The histogram in Figure 37 shows that over 60% of the words have four characters or less. Particularly in the segmentation-free scenario, spotting short words is more difficult than spotting longer words due to the reduced specificity of the query word model. The shorter the word, the more likely is the occurrence of the word within a longer word. Such occurrences lead to image regions that are visually similar to the query. At the same time these regions are irrelevant to the query according to the evaluation protocol. Short words are also harder to detect according to the relevance criterion, cf. Equation 69. Since IoU is a *relative* measure, the detection overlap tolerance in absolute image pixels shrinks with the size of the word. In this regard, it is also problematic that the bounding box annotations in the George Washington benchmark are inaccurate. Particularly the small words are arbitrarily padded with document image background.

It can be questioned whether the use of a larger number of very short query words such as *a*, *in*, *the*, *them* etc., so-called *stop words*, simulates a realistic word spotting scenario. However, it avoids an arbitrary query word selection and leads to a large number of queries in total. The George Washington benchmark will be denoted as GW20.

5.2.2 *Jeremy Bentham manuscripts*

The Bentham manuscripts have been used in the word spotting competitions [PZG+14; PTV15b] at the *Int. Conference on Frontiers in Handwriting Recognition (ICFHR) 2014* and the *Int. Conference on Document Analysis and Recognition (ICDAR) 2015*. The document images originate from the *University College London*, UK [LM81]. The manuscripts contain works on law and moral philosophy and have been analyzed in the context of the project *Transcribe Bentham* where over 19,000 document images have been transcribed [CGS+18]. The documents considered in the competitions were written by Jeremy Bentham and his secretaries during a period of 60 years at the end of the 18th and the beginning of the 19th century, cf. [PZG+14]. The writing style varies considerably. However, the writing style does not change from page to page. Instead, larger groups of pages can be identified where the writing style within the groups is rather homogeneous. Hence, both benchmarks can be considered as challenging but suitable for the query-by-example scenario where no training annotations are available. Figure 36b and Figure 36c show sections of document images used in the 2014 and 2015 competitions. The word images in Figure 38 give an impression of the writing style variability.

(a) Bentham 2014 (BT50) (b) Bentham 2015 (BT70)

Figure 38: Query word images from the Bentham manuscripts. The word images give an impression of the writing style variability for the words *character* (a) and *Forgery* (b). The examples have been used in the 2014 and 2015 word spotting competitions [PZG+14; PTV15b].

It is important to note that the results which are reported in the competitions are based on non-standard relevance criteria in the segmentation-free scenario in order to emphasize the detection qualities of the methods. However, in [ZPG17] the corresponding results have been re-evaluated[2] with the standard IoU measure, cf. Equation 69, where the IoU threshold is $\iota = 0.5$ and the main performance measure is mAP. For this reason, the following evaluations for the Bentham 2014 dataset in Section 5.3 will use the evaluation protocol from [ZPG17]. A comparison to the *original* Bentham 2014 competition results [PZG+14] as well as the *original* Bentham 2015 competition [PTV15b] results can be found in Section 5.4.4 along with the comparisons to the corresponding results from [ZPG17].

The segmentation-free query-by-example benchmark that was defined for the 2014 competition[3] contains 50 document images and 290 query word images. All query words have at least seven characters and appear at least five times in the dataset [PZG+14]. Since the query word images have been cropped from the 50 document images, the query word region can be detected and will typically be the first result in the retrieval list. Three query word images that show the word *character* can be seen in Figure 38a.

The segmentation-free query-by-example benchmark that was defined for the 2015 competition[4] contains 70 document images and 1,421 query word images. Figure 38b visualizes examples for the query word *Forgery*. The 1,421 query word images show 234 different query words. These words have six to 15 characters and appear at

2 The re-evaluation was possible because the authors of [ZPG17] were involved in the organization of both competitions and had access to the original submissions.

3 `http://vc.ee.duth.gr/H-KWS2014/`, accessed on July 13, 2019.

4 `http://transcriptorium.eu/~icdar15kws/`, accessed on July 13, 2019.

least four times in the document images. Appearances of the query words with upper and lower case characters are not distinguished in the benchmark annotations. It is important to note that the query word images have been extracted from a dedicated set of document images that are *not* included in the 70 document images. Further query set statistics can be found in [PTV15b].

The major challenges for segmentation-free query-by-example word spotting in the Bentham manuscripts include writing style variabilities as well as word size variabilities. The generalization across different writing styles based on a single example of the query word is particularly difficult. The word size variabilities pose a challenge with respect to the IoU relevance criterion which requires highly accurate detections. This is especially problematic for patch-based methods that rely on the size of the query word image. The Bentham 2014 benchmark will be denoted as BT50 and the Bentham 2015 benchmark will be denoted as BT70.

5.2.3 *Konzilsprotokolle and Botany*

The Konzilsprotokolle and Botany benchmarks have been used in the word spotting competition [PZG+16] at the *Int. Conference on Frontiers in Handwriting Recognition (ICFHR) 2016*. Both benchmarks have been prepared in the European project *READ*[5] and will be used in order to evaluate segmentation-free query-by-example and query-by-string word spotting.

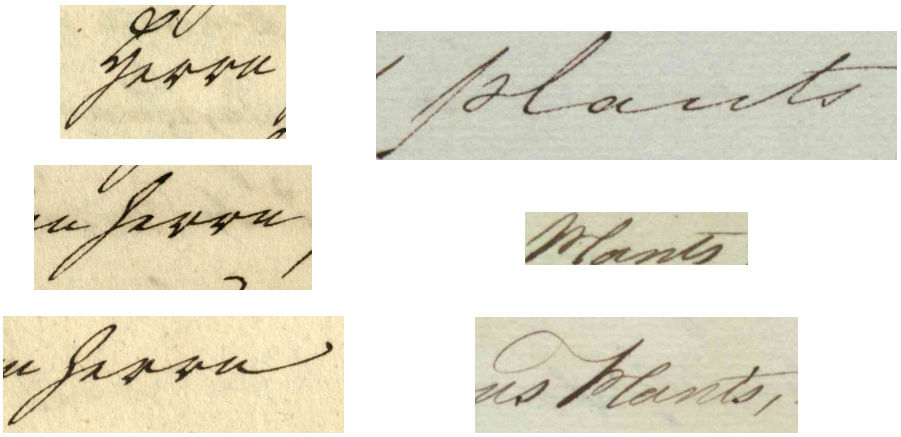
The *Konzilsprotokolle* collection contains around 18,000 documents and is archived at the *University of Greifswald*[6], Germany. The document images contain notes of formal meetings that have been written in the 18th century. In contrast to all other datasets presented in Section 5.2, the script is German *Kurrent* instead of English *Latin*. The writing style in the document images of the benchmark can be considered as rather homogeneous. Thus, the benchmark is suitable for the targeted scenario where no or only limited amounts of annotated training samples are available. Figure 36d shows a section of a document image from the Konzilsprotokolle benchmark. The word images in Figure 39a give an impression of the writing style variability.

The *Botany in British India* collection is hosted at the *British Library*[7], London, UK. The collection contains manuscripts on various botanical topics that have been written in the 19th century. The characteristics of the documents are very diverse. The document images in the benchmark contain several different writing styles and the script has a very unique visual appearance in most of the document images. For

5 `http://read.transkribus.eu`, accessed on July 13, 2019.

6 `http://www.digitale-bibliothek-mv.de/viewer/`, accessed on July 13, 2019.

7 `https://www.bl.uk/collection-guides/botany-in-british-india`, accessed on July 13, 2019.



(a) Konzilsprotokolle (KP20) (b) Botany (BO20)

Figure 39: Query word images from the Konzilsprotokolle and Botany datasets. The examples give an impression of the writing style variability for the words *Herrn* (a) and *plants* (b). Both datasets have been used in the 2016 word spotting competition [PZG+16].

these reasons, the dataset characteristics are *unsuitable* for the targeted scenario where the annotated sample data is scarce. The benchmark is included in order to demonstrate the limitations of the method that has been proposed in Chapter 4. Furthermore, the consideration of the benchmark allows for a comparison with the same methods that have been evaluated on the Konzilsprotokolle benchmark. A section of a document image from the Botany benchmark is shown in Figure 36e. The word images in Figure 39b illustrate the large word size variability and writing style variability.

The main objective of the 2016 competition[8] is the evaluation of word spotting methods on different scripts and an analysis with respect to the required amounts of training data [PZG+16]. For this purpose, the competition defines benchmarks on the two collections such that each benchmark dataset contains 20 document images. In the segmentation-free scenario, the competition reports results for two different annotation set sizes. The training sets for the Konzilsprotokolle dataset contain 1,849 and 16,919 word-level annotations. The training sets for the Botany dataset contain 1,684 and 21,981 word-level annotations. Generally, the training annotations are obtained from a dedicated set of document images. For both benchmarks, this means that the documents in the training sets and the corresponding 20 documents, which are used for word spotting, are disjoint. The same sets of training annotations are available for query-by-example and query-by-string. The query sets contain query words with at least three appearances in the corresponding 20 pages. The query lengths vary from two to 14 characters. The majority of the words has

8 `https://www.prhlt.upv.es/contests/icfhr2016-kws`, accessed on July 13, 2019.

five characters or more. Further statistics on word occurrences and queries can be found in [PZG+16].

For the query-by-example scenario, query word images are provided that have been cropped from the corresponding 20 pages. Therefore, the average precisions are typically biased by the occurrence of the query word regions at the top of the retrieval lists as in the GW20 and BT50 benchmarks. The query set for the Konzilsprotokolle dataset contains 200 query word images and the query set for the Botany dataset contains 150 query word images. Examples for query word images are shown in Figure 39. It is important to note that for evaluating query-by-example word spotting with BoF-HMMs, *no* training annotations will be used. For the query-by-string scenario, the query sets are given as the lexica of the query word image transcriptions from the corresponding query-by-example benchmarks. This makes two sets of 101 queries for the query-by-string benchmarks on the datasets. The relevance criterion is always IoU with an overlap threshold of $\iota = 0.5$. The main performance measure is mAP_{ip}.

The major challenges that are specific to the Konzilsprotokolle dataset are the characteristics of Kurrent script. The ascenders and descenders in Kurrent are typically so long that they reach into the adjacent text lines. This leads to touching pen-strokes of different words and is particularly problematic for word spotting methods that rely on connected components. Furthermore, the visual appearances of some Kurrent characters are very similar, e.g., the Kurrent character shapes for *c*, *e*, *n*, *m*, *r*, *u* and *w*, cf. [Aug96]. In this way, also the visual appearances of different words tend to be more similar to each other than in Latin script. Figure 39a illustrates similar character shapes within the word *Herrn*. The German word *Herrn* translates to *Mister* in English.

The most important challenge that is specific to the Botany dataset is the very large visual variability of the text in the document images. Challenges that apply to both datasets are document degradations, like ink bleeding through paper, and the selection of the query sets. The query sets contain words that are orthographically very similar to each other, such as *Inspection*, *Inspector* or *three*, *trees*. The Konzilsprotokolle benchmark will be denoted as KP20 and the Botany benchmark will be denoted as BO20.

5.3 OPTIMIZATION

The optimization of BoF-HMMs for segmentation-free word spotting is focussed on the query-by-example scenario where no annotated training material is available. Query-by-string word spotting is intended as an extension to this scenario. In order to allow for a seamless integration, it is required that query-by-string word spotting is compatible with the query-by-example configuration. Most importantly, this refers to line region representations as these can be precomputed.

The patch-based decoding framework should be compatible in order to limit the complexity of the approach. The more meta parameters require an adjustment for a new scenario or dataset, the less robust is the approach with respect to the risk of an inadequate configuration.

In the following, the objective is to analyze the effect of different parameterizations and architectural design choices to the performance of the method. For this purpose, experiments will be performed on the GW20 benchmarks for query-by-example and query-by-string word spotting as well as on the BT50 query-by-example benchmark. The datasets have been chosen due to their different characteristics, cf. Section 5.2. Most importantly, the GW20 benchmark contains a large number of queries, cf. Section 5.1, and the BT50 benchmark is challenging with respect to the writing style variability. In analogy to the structure of Chapter 4, the evaluation includes:

- text hypotheses (Section 5.3.2),
- BoF representations (Section 5.3.3),
- output models (Section 5.3.4),
- word models (Section 5.3.5),
- character models (Section 5.3.6),
- context models (Section 5.3.7),
- patch-based retrieval (Section 5.3.8),
- retrieval efficiency (Section 5.3.9) and
- region snapping (Section 5.3.10).

In order to analyze the effect of individual adjustments, all evaluations are based on an optimized query-by-example configuration (Section 5.3.1). Only a single meta parameter is changed at a time. Meta parameters that do no affect the optimized query-by-example configuration directly, are evaluated in the context of the corresponding methodological variant. For example, this applies to the meta parameters of the different BoF output models or the meta parameters for query-by-string word spotting. Representations that depend on a random initialization are obtained once and reused for all experiments. The influence of random initializations is discussed along with the corresponding representations.

Within each of the following tables, a single configuration is selected as the *reference* configuration either due to its performance or due to its computational efficiency. The difference between the performance achieved by the *reference* configuration and the performance achieved by a *modified* configuration is analyzed with a permutation test, see Section 5.1.2. In the following tables, this is indicated by printing the mAP performances in different fonts. The performance of the

Table 4: Optimized BoF-HMM query-by-example configuration

Meta parameter	Value
ER thresholds	12
ER descriptor size	16
SIFT descriptor size	32 (BT50) or 48 (GW20)
Grid sampling step (g)	3
Vocabulary size (V)	4096
Mixture model	Visual words
Term weighting	Frequency
HMM states	$\lfloor 0.7\, T_q \rfloor$
HMM topology	Linear
Baum-Welch iterations	0
Patch sampling rate (υ)	8
Patch width expansion (ψ)	0.5
Re-ranking mask	3×3
Region snapping area (η_{size})	1.0
Region snapping weight (η_{weight})	0.1

reference configuration is printed in a bold font. The performances for configurations that are *significantly different* ($\hat{p} \leqslant 0.05$) in comparison to the reference configuration are printed in an italic font. The results for all other experiments are printed in a regular font. All performance measurements are rounded to one decimal place.

5.3.1 *Query-by-example configuration*

In the targeted scenario, historians *start* to explore a collection of document images with query-by-example word spotting. In this exploration process, it is essential to keep the adjustments that are required for working with a new dataset to a minimum. All experiments on all dataset are based on the same *optimized* query-by-example configuration for this reason, see Table 4. The characteristic property of this optimized configuration is that *none* of the parameterizations considered in the targeted scenario, achieves a performance which is *significantly better* ($\hat{p} \leqslant 0.05$) than the performance achieved by the optimized configuration. Only a single meta parameter, i.e., the SIFT descriptor size, requires dataset specific adjustments. This is a very important result for the applicability of the method in practice.

The configuration in Table 4 has been obtained within an informal optimization. The following evaluation will confirm this optimization on the GW20 dataset and on the BT50 dataset. The optimized query-by-

Table 5: Text hypotheses evaluation

ER thresholds	ER descriptor size	GW20 mAP [%]	BT50 mAP [%]
4	16	74.5	*38.5*
12	16	**75.1**	**56.9**
20	16	75.1	57.0
12	8	75.1	56.7
12	32	75.2	55.1

example configuration achieves 75.1% mAP on GW20 and 56.9% mAP on BT50. In the following tables, the optimized query-by-example configuration is selected as the reference configuration unless the table presents an evaluation of a methodological variant, such as different BoF output models or query-by-string word spotting.

It is important to note that *no* quantitative evaluation of region snapping (Section 4.6.5) is performed on GW20 due to the inaccurate bounding box annotations. Region snapping leads to document image regions that enclose the retrieved text components tightly. Therefore, relevant detections will regularly be considered as irrelevant with 50% IoU. Instead, the application of region snapping on the GW20 dataset will be demonstrated in a qualitative evaluation.

In a similar manner, the minimum number of visual words per frame for excluding *invalid* descriptors that overlap with the vertical region bounds (Section 4.5.2 and 4.6.1) can *only* be evaluated on the GW20 benchmark. All other benchmarks do not define query image regions on the document images but as independent query word images. Thus, there are no invalid descriptors in the latter case and it is also impossible to incorporate descriptors that represent the horizontal context (Section 4.5.2).

5.3.2 *Text hypotheses*

Text hypotheses are required in order to extract line hypotheses, estimate whitespace regions and for snapping potentially relevant document image regions to text components, cf. Section 4.2. Table 5 shows how different parameterizations for generating text hypotheses affect the word spotting performance. In this regard it is important to note that the experiments on the GW20 benchmark do not incorporate region snapping. Therefore, text hypotheses have a greater influence on the BT50 benchmark than on the GW20 benchmark.

The most sensitive meta parameter for generating text hypotheses is the number of *ER thresholds* which controls the resolution of the *extremal region* (ER) tree. The higher the number of thresholds, the

more variants of potential text components are extracted. Another meta parameter is the size of the descriptors that are used in order to obtain local contrast information. The *ER descriptor size* refers to the edge length in image pixels of the square descriptor area. The larger the descriptors, the smoother is the resulting text score map. The smoother the text score map, the less details in the document image are incorporated. This reduces the number of small hypotheses and allows to distinguish pen-strokes in the text core area and between adjacent text lines, i.e., ascenders and descenders.

For segmentation-free word spotting with BoF-HMMs, it is important that mostly all text components are represented by at least a single text hypothesis. This is achieved if the number of ER thresholds is sufficiently high. Table 5 shows that the mAP drops significantly ($\hat{p} \leqslant 0.05$) if only 4 threshold values are used on the BT50 dataset. This is due to a substantial number of document images with relatively low image contrast. The small performance reduction on the GW20 dataset is not significant ($\hat{p} > 0.05$). Apart from local contrast variations, the pen-stroke has typically a strong contrast. The local contrast variations on GW20 are not as critical for extracting line hypotheses. Line hypotheses are obtained for all text hypotheses such that a missed text component can be compensated by a text component with a similar height in the same line.

With respect to the ER descriptor size, high retrieval performance is measured on both benchmarks for a value of 16 pixels. The parameter can be considered as very robust because it does not influence the mAP significantly ($\hat{p} > 0.05$) for the different descriptor size values that are evaluated in Table 5.

5.3.3 *Bag-of-features representations*

BoF sequences are used in order to represent document image regions, such as query word regions and line hypotheses, cf. Section 4.3. The BoF representations are extracted from the columns of a dense grid of visual words. Visual words are computed by quantizing SIFT descriptors with respect to a codebook, i.e., the visual vocabulary. Three meta parameters are important for defining this process. The *SIFT descriptor size* specifies the edge length of the square document image area that is represented by each descriptor. The influence of different descriptor sizes is presented in Table 6. The *grid sampling step* specifies the distance of the SIFT descriptor center points in horizontal and vertical direction. The *vocabulary size* defines the number of visual words in the codebook. Both parameters are evaluated in Table 7.

The size of the SIFT descriptors is the most sensitive and, therefore, also most important meta parameter for segmentation-free word spotting with BoF-HMMs. This is due to the fact that the SIFT descriptors represent the visual image features. Essentially, these features are the

Table 6: BoF descriptor size evaluation

SIFT descriptor size	GW20 mAP [%]	BT50 mAP [%]
16	*64.5*	*46.6*
24	*71.0*	54.5
32	*73.7*	**56.9**
40	75.0	53.7
48	**75.1**	*49.7*
56	*73.9*	*44.9*

basis for computing visual similarities. The ability of the method to generalize from the estimated visual appearance of the query to the visual appearance of document regions that are relevant to the query, largely depends on the specificity of the descriptors. The generalization capability improves if the specificity is reduced. However, if the specificity is reduced, the model *also* loses its capability to distinguish *relevant* document regions from *irrelevant* regions.

The descriptor *size* influences the specificity the most. The smaller the represented document image area, the less specific is the area to a potentially relevant instance of a word. While very small descriptors rather represent parts of characters, larger descriptors represent groups of characters. It is more likely to find a visually similar part of a character in a different relevant (or irrelevant) context than to find a visually similar group of characters in a different relevant (or irrelevant) context. Figure 40 illustrates this behavior for three different SIFT descriptor sizes on GW20. The mean interpolated precision-recall curves show that the smaller descriptors are not sufficiently specific. This is indicated by the growing differences of the curves until recall level 50%. Thus, visually similar instances of the query do not obtain high similarity scores. The mIP at recall level zero is high for all curves due to the inclusion of the query word region in the retrieval list. In general, this behavior leads to different local optima for different descriptor size values on different benchmarks.

Table 6 shows that the locally optimal result for GW20 is achieved with a descriptor size of 48 pixels while the locally optimal result for the BT50 benchmark is achieved with a descriptor size of 32 pixels. If the performance of these two descriptor sizes is compared on the GW20 benchmark *only*, the best performance of 75.1% mAP is significantly better ($\hat{p} \leqslant 0.05$) than the 73.7% mAP achieved with the parameter value 32. The effect can be observed in analogy on the BT50 benchmark. This sets the descriptor size apart from all other meta parameters considered in this chapter. In practice, finding an *optimal* SIFT descriptor size requires a validation set of annotated samples that is *representative* for the corresponding document collection. Since the

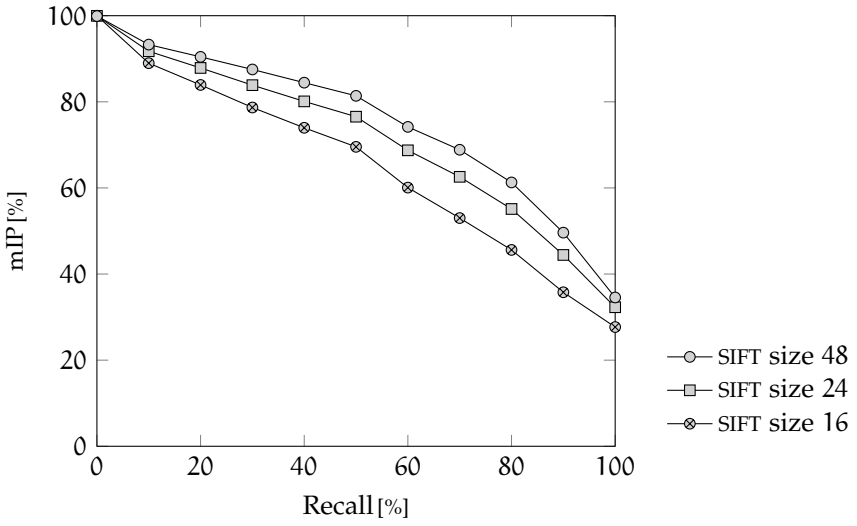


Figure 40: SIFT descriptor size evaluation on GW20.

document images in the BT50 and BT70 benchmarks originate from the same document collection, the BT50 benchmark can be seen as a validation set for BT70. However, in the targeted scenario where historians *start* the exploration of a document collection, such a validation set will typically *not* be available. Since the descriptor size is related to the typical size of the text in the document images, an estimate of the height of the text core area, cf. [FB14, Fig. 12.3], can be used as an estimate for the descriptor size, see Section 5.4.1.

Table 7 presents the evaluation for the grid sampling step and the vocabulary size. The evaluation of both parameters is consistent with the results that have been reported on BoF image representations in the word spotting literature, cf. e.g., [ART+15].

The descriptor sampling step in the dense grid has a large influence on the performance. The performance differences between sampling steps of 3 and 5 pixels are *significant* ($\hat{p} \leqslant 0.05$) on both benchmarks. However, it has to be noted that a higher grid resolution comes at the cost of reduced efficiency. While a grid step of 5 pixels results in an average of 651×400 grid rows and grid columns on the GW20 dataset, a grid step of 3 pixels leads to an average grid resolution of 1084×667 over the 20 document images. This grid resolution is already demanding with respect to computational efficiency and memory efficiency. An even smaller grid step can be considered as infeasible for these reasons. For example, a grid step of 1 pixel leads to an average resolution of 3251×1999 on the GW20 dataset. The average resolutions for the different grid steps on the BT50 dataset are similar to the corresponding resolutions on GW20.

Less critical with respect to the performance is the vocabulary size. The larger the number of visual words, the smaller is the quantization error, cf. Section 2.3, and the higher is the specificity of the visual

Table 7: BoF representation evaluation

Grid sampling step $\mathfrak{g}$	Vocabulary size V	GW20 mAP [%]	BT50 mAP [%]
3	4096	**75.1**	**56.9**
5	4096	*73.0*	*51.1*
3	512	*68.0*	53.6
3	1024	*71.0*	56.0
3	2048	*73.5*	57.3
3	6144	75.5	55.9
3	8192	75.5	55.4

words with respect to the SIFT descriptors in the document images. The experiments on the GW20 dataset show that high performance is achieved with large vocabulary sizes. The mAP starts to converge at 4096 visual words. While the smaller vocabulary sizes perform significantly worse ($\hat{p} \leqslant 0.05$), larger vocabularies do not lead to significant improvements ($\hat{p} > 0.05$). This can be explained with the relatively homogeneous visual appearance of the text in the document images on GW20. Due to the writing style variabilities in the BT50 dataset, the behavior cannot be observed in the same way. On the BT50 benchmark, the highest mAP is obtained for 2048 visual words. However, it is important to note that *none* of the differences to the 56.9% mAP of the reference configuration in Table 7 are significant ($\hat{p} > 0.05$). Since document collections with limited visual variability are the targeted application domain for the proposed method, a visual vocabulary with 4096 visual words can be considered as a robust parameterization.

An interesting question concerns the influence of the randomization in the visual vocabulary estimation. This refers to the random subset of SIFT descriptors used for Lloyd's algorithm, the random selection of initial centroids for Lloyd's algorithm and the random shuffling that is required for clustering all descriptors from all document images with MacQueen's k-means algorithm, cf. Section 4.3. Experiments on GW20 show that the influence of the randomization on the mAP is marginal. The results for ten independent clusterings that do not include the visual vocabulary of the reference experiment, achieve a mean of 75.2% mAP and a standard deviation of 0.1% mAP. The observed minimum and maximum values are 75.1% mAP and 75.3% mAP. The high robustness can be explained with the very large number of over 14 million clustered descriptors on the GW20 dataset. For comparison, over 34 million descriptors are clustered on the BT50 dataset.

5.3.4 *Bag-of-features output models*

BoF output models are required for modeling the generation of BoF vectors in the statistical HMM process. In this regard, three approaches have been presented in Section 4.4. The *von Mises-Fisher* (vMF) mixture model and the *exponential Dirichlet compound multinomial* (EDCM) mixture model, cf. Section 4.4.1 and 4.4.2, are estimated from BoF vectors in an unsupervised manner. For this purpose, BoF vectors are extracted from all line hypotheses at all line heights that have been obtained based on the text hypotheses in the document images, cf. Section 4.2.2. This results in 507,290 BoF vectors on the GW20 dataset and 1,040,826 BoF vectors on the BT50 dataset. The third model, i.e., the visual-word mixture model, is directly based on BoF vectors and does not require another estimation step, cf. Section 4.4.3. It is used in the optimized query-by-example configuration, see Table 4. In the following, the meta parameters of the output models will be discussed. It is important to note that the reference configurations for the vMF and EDCM models are selected from the corresponding vMF and EDCM parameterizations. These parameterizations are specified in the *vMF mixture model* evaluation and in the *EDCM mixture model* evaluation. Finally, all output models will be compared with each other.

vMF mixture model

Table 8 and 9 show the evaluation of the vMF mixture model. In order to obtain an initial model, M BoF vectors are randomly drawn from the sample set. Afterwards, the spherical version of the Lloyd algorithm is run for ten iterations. Based on the resulting codebook, the initial vMF mixture model is obtained. An important aspect in the estimation of a vMF mixture model on high dimensional input data is the estimation of the concentration parameters κ_k with $k \in \Omega_M$, cf. [BDG+05]. Even for the toy example in Figure 20, the mixture model shown in Figure 21 does not generalize well. For the high dimensional and sparse BoF vectors considered in Section 4.3, the concentration parameters of the individual mixture components quickly approach *infinity* while training with the EM algorithm. Due to this reason, the performance achieved with different strategies for obtaining concentration parameters is presented in Table 8.

In the *unbounded* estimation strategy, the training is directly terminated after obtaining the initial model. In this case, the κ estimates are based on the BoF vectors that have been assigned to the centroids in Lloyd's algorithm. For example, after the initialization on GW20, the mean over 512 estimated κ values is 2816 with a standard deviation of 5240. This indicates that the model is already degenerated. Also Table 8 shows that the mAP achieved with the *unbounded* estimation is *significantly* worse ($\hat{p} \leqslant 0.05$) than the performance achieved with the vMF reference configuration on both benchmarks.

Table 8: vMF concentration parameter

Concentration max κ	Beam width $-\log b$	GW20 mAP [%]	BT50 mAP [%]
unbounded	14	*64.4*	*37.5*
128	14	*66.9*	*41.5*
64	7	*68.6*	*45.8*
32	7	71.1	50.2
16	3	**71.6**	**50.6**
8	1	*70.3*	49.5
16	5	*72.9*	*45.1*
16	1	*67.9*	*43.9*

Since the concentration parameters are largely over-estimated, a simple approach is to limit the κ values to an upper bound heuristically during EM training, cf. Section 4.4.1. The EM algorithm is run for ten iterations or until the relative improvement of the complete data log-likelihood is smaller than 10^{-3}, cf. [Fin14, p. 67]. Table 8 shows how the mAP values increase with smaller limits for κ until the performance reaches a local optimum. The performance differences are mostly significant ($\hat{p} \leqslant 0.05$, indicated with an italic font in Table 8). If the limit for κ decreases, the mixture distribution gets smoother and smoother. This has two negative side effects. The first side effect is the growing number of non-zero mixture components per frame $\hat{M}$, as explained in Section 4.6.2. In order to make the application of the vMF mixture model feasible, the beam pruning meta parameter has to be adjusted for every parameterization, cf. Equation 38. Table 8 shows the required adjustments for the beam in the negative logarithmic domain, i.e., the so-called *beam width*, cf. [Fin14, p. 187]. The vMF reference configuration uses a κ limit of 16 with a beam width of 3. In this case, the average number of non-zero mixture components per frame for word spotting on the GW20 benchmark is 34. If the beam width is increased to 5, the average number of components per frame is already 120. Only 4 non-zero components per frame are obtained in average for a beam width of 1. In this regard, the second side effect can be observed. Table 8 shows that the performance on the BT50 benchmark is significantly worse ($\hat{p} \leqslant 0.05$) with a beam width of 5 in comparison to the performance achieved with a beam width of 3. This can be explained with an over-generalization of the model. Out of 512 mixture components, 167 non-zero components per frame can be observed in average on the BT50 dataset with a beam width of 5.

It is important to note that *none* of the performance differences in Table 8 are due to different random initializations. For each of the two benchmarks, a single codebook has been estimated with the spherical

Table 9: vMF mixture model

Mixture components M	Vocabulary size V	GW20 mAP [%]	BT50 mAP [%]
256	4096	71.2	48.9
512	4096	**71.6**	**50.6**
1024	4096	70.8	50.3
512	2048	*69.7*	50.3
512	8192	71.9	47.9

version of Lloyd's algorithm. The EM algorithm is always based on the same initial codebook from the corresponding benchmark dataset.

Table 9 shows the effect for different numbers of mixture components and different vocabulary sizes. The performance that is measured for the different parameterizations is consistent on both benchmarks. On GW20, only a single parameterization achieves a performance that is significantly worse ($\hat{p} \leqslant 0.05$) than the performance of the corresponding vMF reference configuration. All results are based on different initial spherical Lloyd codebooks. Therefore, a performance deviation can be expected which is due to the random initialization. Due to the considerable performance difference to the reference configuration that uses the visual-word mixture model, cf. e.g., Table 6, this deviation can be neglected. The influence of the SIFT descriptor size is consistent with the results that have been obtained for the visual-word mixture model, see Table 6. The results are reported in Table 45 which can be found in Section A.2.

EDCM mixture model

The evaluation of the EDCM mixture model is presented in Table 10 and 11. Table 10 shows different parameterizations for the estimation of the model. Table 11 shows the evaluation for different numbers of mixture components and different sizes of the visual vocabulary.

The estimation with an EM algorithm requires an initial model. The initial model is obtained by estimating a single EDCM distribution from the entire dataset in a first step. Afterwards, M random deviations of this distribution constitute the initial mixture model, cf. Section 4.4.2. Within each deterministic annealing phase (see below), the EM algorithm is run for at most ten iterations or until the complete data log-likelihood converges, cf. [Fin14, p. 67], i.e., the relative improvement is smaller than 10^{-3}. In order to make the estimation robust, the use of two heuristics is proposed in [MKE05] and [Elk06]. For scenarios where the mixture model is evaluated on data that was unknown during training, smoothing of the concentration parameters is proposed in [MKE05]. This way, a degeneration of the model is

Table 10: EDCM estimation parameters

Smoothing ρ	Det. annealing $(\tau_0, \ldots, \tau_n)$	GW20 mAP [%]	BT50 mAP [%]
0	$(2, 1)$	70.9	50.4
0.01	$(2, 1)$	**70.1**	**50.8**
1	$(2, 1)$	69.7	50.0
0.01	$(5, 2, 1)$	69.1	51.1
0.01	1	70.5	50.8

precluded that could theoretically occur due to a factor of zero in the product in the EDCM probability mass function, see Equation 19. The application of the smoothing factor is shown in Equation 21. In the proposed method, this could be relevant when document images are added for word spotting that have not been available during model estimation. Therefore, the EDCM mixture model uses a smoothing factor of $\rho = 0.01$ in the EDCM reference configuration as proposed in [MKE05]. However, since this scenario does not apply to the GW20 or the BT50 benchmark, no significant differences ($\hat{p} > 0.05$) can be observed for different values of this factor. In a similar manner, no significant performance differences ($\hat{p} > 0.05$) can be observed with the deterministic annealing strategy [UN98] which is used in [Elk06], cf. Equation 20. A scenario where deterministic annealing is required for estimating an EDCM mixture model, is the small training dataset of just 400 *bag-of-words* (BoW) representations in [Elk06]. In comparison, over 500,000 BoF vectors are used for training on GW20 and over one million BoF vectors are used for training on BT50. The EDCM reference configuration in Table 10 uses deterministic annealing because it does not affect the performance significantly but has the potential of making the estimation of the mixture model more robust.

The robustness of the model is also demonstrated in Table 11. Except for the 128 mixture components and 1024 visual word configurations on the GW20 benchmark ($\hat{p} \leqslant 0.05$), none of the other measured performances differs significantly ($\hat{p} > 0.05$) from the EDCM reference configuration. Since all models in Table 10 and 11 have been obtained with different random initializations, this shows that the model optimization is largely independent of its starting point. Significant influences on the performance with the EDCM mixture model can mostly be observed for different descriptor sizes, see Table 46 in Section A.2. The descriptor size evaluation is consistent with the results that have been obtained for the other output models, cf. Table 6 and Table 45.

With respect to the approximation of the *Dirichlet compound multinomial* (DCM) distribution through the EDCM distribution, cf. Section 4.4.2 and see Section B.2, the considered scenario is perfectly suitable. For word spotting on the GW20 benchmark, the BoF vectors contain only

Table 11: EDCM mixture model

Mixture components M	Vocabulary size V	GW20 mAP [%]	BT50 mAP [%]
128	4096	*67.1*	51.7
256	4096	69.9	50.6
512	4096	**70.1**	**50.8**
1024	4096	69.9	48.6
512	1024	*68.8*	51.2
512	2048	70.2	50.7
512	8192	69.5	48.9

16 (quantized) descriptors per frame in average. The number of *different* visual words per frame is in average 10. In the same way, the mean concentration in the mixture model of the EDCM reference configuration is 0.01 with a standard deviation of 0.05. The average number of descriptors per frame for word spotting on the BT50 benchmark is 17 where 13 different visual words per frame occur in average. The mean and standard deviation of the concentration parameters in the EDCM reference configuration for the BT50 dataset is 0.01 and 0.06. For comparison, in [Elk06] the approximation is considered as very accurate with an average concentration parameter value of 0.06.

The negative logarithmic beam width during mixture model estimation and model decoding is 14. This leads to an average of 9 non-zero EDCM components per frame on the GW20 benchmark and an average of 13 non-zero EDCM components per frame on the BT50 benchmark. All EDCM related statistics on GW20 and BT50 refer to 512 mixture components and 4096 visual words in the EDCM reference configuration. With respect to the number of visual words per frame, it is important to recall that the reference configuration uses a grid sampling step of 3.

Visual-word mixture model

The visual-word mixture model is used in the optimized query-by-example configuration which has been presented in Table 4. Since this model is directly based on BoF representations, cf. Section 4.4.3, all related meta parameters have already been discussed in Section 5.3.3.

The visual-word mixture model represents the generation of a single visual word. However, a *bag*-of-visual-words, i.e., *multiple* visual words, have been observed. Therefore, the model can be considered as *pseudo-discrete*. For this purpose, a BoF vector is understood as an estimate for the visual word probabilities given the corresponding frame. Following the Laplace principle, all visual words are assumed to be equally likely and the visual word probability estimates for

Table 12: Visual-word mixture model

Term weighting	GW20 mAP [%]	BT50 mAP [%]
Frequency	**75.1**	**56.9**
Frequency $\times\, p(\mathcal{V} = v \mid \Theta)^{-1}$	74.9	57.0
Binary	75.1	56.9

the frame are obtained as relative visual word frequencies, see Equation 32. In the following, the choice of using relative frequencies will be discussed.

For representing texts with BoW, relative frequencies are a standard approach to term weighting, cf. [BR11, Sec. 3.2.3], and estimating term probabilities, cf. [RST+03]. However, in BoW applications to retrieval and classification, an important assumption is that the terms are discriminative for the classes considered, e.g., documents that are relevant and documents that are non-relevant with respect to a query. A common heuristic is to emphasize the terms that are *overall infrequent*, since these terms are assumed to be more specific and carry more information than the terms that are overall frequent. In this spirit, the overall frequent terms are also more likely to have high frequency in a BoW representation. Term weighting schemes make use of this heuristic by combining the relative term weights with the *inverse document frequency* of the corresponding terms, resulting in term-frequency inverse-document-frequency weighting [BR11, Sec. 3.2.4] for example. In a similar manner, all terms can be considered as equally important regardless of their frequency in the so-called *boolean model*, cf. [BR11, Sec. 3.2.2]. For this purpose, a term is represented by either *zero* or *one*, depending on its occurrence in a document.

Inspired by these approaches, Table 12 shows different methods for representing visual word frequencies. The inverse document frequency is interesting because it has already been used for word spotting with spatial pyramid representations in [RAT+15a]. For this purpose, the inverse visual word probabilities are used as an approximation of the inverse frame frequencies. Finally, the boolean approach is interesting because the considered BoF representations are extremely sparse and, therefore, already similar to the characteristics of the boolean term representations. The visual word frequency representations are directly applied on the BoF vectors. The weighting schemes are used in the numerator and denominator of Equation 32, see Section 4.4.3.

The results in Table 12 show that there are only marginal differences ($\hat{p} > 0.05$) between the frequency representations for word spotting with BoF-HMMs on the GW20 and the BT50 benchmarks. Therefore, the use of relative frequencies can be seen as a design choice that nat-

Table 13: Output model comparison

Output mixture model	GW20 mAP [%]	BT50 mAP [%]
Visual words	**75.1**	**56.9**
vMF	*71.6*	*50.6*
EDCM	*70.1*	*50.8*
Multinomial	*68.1*	*49.5*

urally arises from probabilistic BoW and BoF models that have been used in the literature, e.g., [RST+03; CDF+04]. The behavior can be explained with the large number of visual words and the comparably very small number of descriptors per frame. This is consistent with the assumptions of the visual-word mixture model. The *occurrence* of individual visual words in the query model and in the frames is most important.

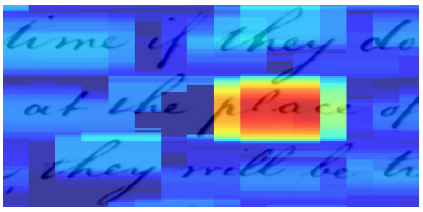
Output model comparison

The three output models that have been evaluated so far have been considered due to their very different characteristics. Table 13 shows a comparison of the models on the GW20 and the BT50 benchmarks. Additionally, the multinomial mixture model is added to the comparison as a reference. The multinomial mixture model has been discussed in Section 2.4. Table 13 shows that the visual-word mixture model *significantly* outperforms ($\hat{p} \leqslant 0.05$) the other models on the benchmarks considered. Thus, the visual-word mixture model achieves the best trade-off between generalization capabilities and specificity for the query HMM.

The consideration of the vMF mixture model is inspired by the wide use of cosine similarity for word spotting. However, the necessity of adjusting the concentration parameters manually is more than suboptimal. Since it is infeasible to adjust the concentration parameters for the mixture components individually, a global limitation for the automatic estimates is used. This value has a strong effect on word spotting performance and computational efficiency.

The EDCM mixture model is perfectly suited for the characteristics of the BoF representations. The EDCM distribution is an approximation of the *Dirichlet compound multinomial* (DCM) distribution that is specifically designed for high dimensional and sparse BoW vectors [Elk06]. While the suitability for sparse data is a characteristic of the DCM model, the main advantage of the approximation through the EDCM model is the considerable simplification of the DCM model for this scenario, see Appendix B. Since the DCM model extends the multinomial model, an important question is whether this extension is beneficial for segmentation-free word spotting with BoF-HMMs.

Visual-word mixture model



EDCM mixture model

Figure 41: Comparison of the patch-similarity scores for the visual-word mixture model and the EDCM mixture model. The colors represent the geometric mean of the output probabilities for the frames that were aligned with the query word HMM in the logarithmic domain. Blue corresponds to low similarity and red corresponds to high similarity. The colors in the two images correspond to each other.

Table 13 shows the results for both mixture models. The EDCM mixture model achieves higher mAP values on both benchmarks. This indicates that the EDCM model is better suited for representing the sparse BoF vectors than the multinomial model. It should be noted that the absolute difference of 2% mAP on GW20 is *significant* ($\hat{p} \leqslant 0.05$). In contrast, the absolute difference of 1.3% mAP on BT50 is *not* significant ($\hat{p} > 0.05$). The estimation and evaluation of the multinomial mixture model follows the same procedure that is used for the EDCM mixture model. The probability mass functions and the maximum likelihood estimates for the maximization step in the EM algorithm are adapted, cf. [NMT+00]. Further, the visual word probability distributions of the mixture components have to be smoothed in order to avoid zero probabilities, see Section A.1. The visual word probability distributions are interpolated with the visual word prior probability distribution, i.e., the background model, for this purpose. Section A.1 also presents an evaluation of the multinomial mixture model. The results are consistent with the results that have been presented for the EDCM mixture model in Table 10 and 11.

Another important question that concerns the EDCM mixture model is the significant performance difference in comparison to the visual-word mixture model. This can be explained with the better generalization capabilities of the visual-word model. An EDCM mixture component represents the generation of the entire BoF vector. In contrast, the generation of just a single visual word is represented by the visual-word mixture model. Intuitively, it is more likely to find individual occurrences of visual words in the document images (disjunction) than configurations of multiple visual words in the document images (conjunction), cf. Section 4.4.3. The visual-word mixture model mostly achieves its specificity by using sufficiently specific image descriptors. For example, on GW20 the standard deviation for the mAP measurements of the six descriptors sizes is 3.7% with the visual-word mixture model and only 2.3% with the EDCM mixture model, cf. Table 6

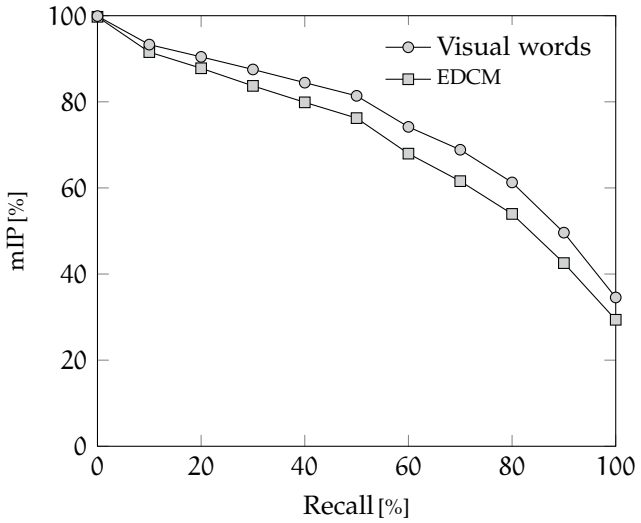


Figure 42: Mean interpolated precision-recall curves for the visual-word mixture model and the EDCM mixture model on GW20.

and Table 46 in Section A.2. Apart from the descriptor size, also the vocabulary size and the number of mixture components influence the generalization capabilities of the EDCM mixture model. Table 11 shows that a reduction of these parameter values does not achieve any significant improvements.

In order to investigate the differences between the EDCM and the visual-word models further, a qualitative analysis of the patch similarity scores is shown in Figure 41. The similarity scores obtained with the EDCM mixture model are considerable more specific to the query word model than the scores obtained with the visual-word model.

For a quantitative analysis of the two models, Figure 42 shows the mean interpolated precision recall curves on the GW20 benchmark. This allows for comparing mIP measurements at different recall levels. The EDCM mixture model is consistently outperformed.

The analysis of the *mean recall* over all queries allows for an interpretation of the specificity of the models. For each query, the recall is computed for the entire retrieval list. The recall for the top-n elements is defined in Equation 70. The visual-word mixture model achieves a mean recall of 89.7% and the EDCM mixture model achieves a mean recall of 85.8%. This *significant* ($\hat{p} \leqslant 0.05$) absolute difference of 3.9% shows that the EDCM model does not generalize as well as the visual-word mixture model. In comparison, the *over-generalizing* visual-word mixture model with a SIFT descriptor size of 24, see Figure 40, achieves a mean recall of 88.8% which is significantly better ($\hat{p} \leqslant 0.05$) than the mean recall with the EDCM mixture model.

The significance test for the *difference of mean recall values* is performed with the permutation test, see Section 5.1.2. For this purpose, the test statistic, cf. Equation 77, uses the absolute difference of mean recall values instead of the absolute difference of mAP values.

Table 14: Query word HMM meta parameters

States $\lfloor \cdot T_q \rfloor$	Iterations	Topology	GW20 mAP [%]	BT50 mAP [%]
0.5	0	Linear	74.5	54.2
0.7	0	Linear	**75.1**	**56.9**
0.9	0	Linear	*72.8*	53.0
0.7	3	Linear	74.6	54.0
0.7	5	Linear	*73.8*	*51.5*
1.0	0	Bakis	74.1	56.2

5.3.5 *Word models*

The query HMM is a compound model which consists of the query *word* HMM as well as the whitespace HMMs and the background HMM, cf. Section 4.5. In the query-by-example scenario, the query word HMM is given by a word model that represents the visual appearance of the query image region. In the following, the evaluation of these word models will be presented. The evaluation of the context models will be presented in Section 5.3.7. All experiments are based on the optimized query-by-example configuration which is summarized in Table 4 in Section 5.3.1.

The most important meta parameters for the query word HMM include the number of HMM states and the number of Baum-Welch training iterations for estimating the model. Since the query word HMM represents just a *single* document image region in the query-by-example scenario, the training process reduces to the estimation of transition probabilities and mixture component weights, cf. Section 4.5.2. Most importantly, the output model is not considered in this regard but estimated in an unsupervised manner, see Section 5.3.4. The number of states is given as a percentage of the length T_q of the BoF vector sequence that is extracted from the query word region. The model is initialized based on a linear alignment of these BoF vectors with the states, see Equation 47 in Section 4.5.2. The topology of the model is linear. Based on the initialization, the model can be refined in an iterative manner.

Table 14 shows the effect of the relative number of states, the number of training iterations and the model topology on the GW20 and the BT50 benchmarks. The locally optimal scaling factor for the number of states is 0.7 in both cases. The number of states influences the specificity of the models. The higher the relative scaling factor, the less flexible is the model with respect to spotting instances of the query word that have a smaller width than the query word image. The number of states is a lower bound for the number of observations, here

BoF vectors, that can be generated by a model with a linear topology. The lower the relative scaling factor, the less specific is the model with respect to the query word image, thus, the model over-generalizes. Table 14 shows that there is no benefit ($\hat{p} > 0.05$) in increasing the flexibility with a Bakis topology. Again, the relative scaling factor specifies the minimum number of observations that can be generated with the model. The factor is 1.0 for the Bakis experiment in Table 14. However, the model can be traversed with at least 50% of the states, due to the possibility to skip every second state. Thus, the specificity of the model with respect to the query word image can be reduced dynamically.

Furthermore, an interesting behavior can be observed for the number of training iterations in Table 14. In comparison to the performance of the reference configuration (printed in a bold font in Table 14), the measured performance is lower at three training iterations and *significantly* worse ($\hat{p} \leqslant 0.05$) at five iterations on both benchmarks. While optimizing the total output probability, the Baum-Welch algorithm focusses on the mixture components, here visual words, that most of the BoF vectors have in common according to their probabilistic alignment with the states. However, with just a single sequence there is no guarantee that these are the components that are relevant for spotting the query word.

Model estimation is based on the BoF vectors that are extracted from the query word image. In practice, the query word image is given as a region in a document image. Therefore, it is possible to exploit the context of the query in the document. Since the query is given as a bounding box, the context can be classified into *horizontal* and *vertical*. The vertical context is located above and below the query word region. The horizontal context is located to the left and to the right of the query word region. Intuitively, the vertical context should *not* be represented in the query word model. The content of the adjacent text lines is not relevant for the query. However, the context to the left and right which is typically whitespace, i.e., document image background, can increase the specificity of the query word model considerably. In this regard it is important to note, that this whitespace context is represented by the query *word* model itself while the whitespace HMMs, see Section 4.5.4, are part of the context representation that is required for localizing the query within a patch, also compare Section 5.3.7.

Provided that a document image region is represented with a sequence of BoF vectors, the influence of the horizontal and vertical context can be controlled by excluding descriptors that overlap with the corresponding region boundaries. These descriptors are referred to as *invalid* in Section 4.3, 4.5.2 and 4.6.1. Table 15 shows the measured performance for pruning *only* descriptors that overlap with the *horizontal* boundaries, pruning *only* descriptors that overlap with the *vertical* boundaries and pruning descriptors that overlap with the *hor-*

Table 15: Pruning of invalid descriptors

Pruning	Min. descriptors	GW20 mAP [%]
Horizontal	15	*65.1*
Vertical	15	**75.1**
Horizontal or vertical	15	*66.6*
Vertical	1	*72.1*
None	–	*73.2*

izontal or vertical boundaries. The results on the GW20 benchmark confirm the expected behavior. It is interesting to note that the results differ from the reference configuration significantly ($\hat{p} \leqslant 0.05$). The absolute differences are considerable, i.e., 10.0% mAP and 8.5% mAP. These large differences can be explained with the characteristics of the GW20 benchmark. Figure 37 shows that over 60% of the queries have four characters or less. Thus, the overall performance is dominated by short query words. Since query word models for short words are naturally less specific than the models for long words, short words benefit from the incorporation of horizontal context the most. Figure 43 shows mAP values per query length for the *vertical* descriptor pruning strategy and the *horizontal or vertical* descriptor pruning strategy. It can be seen that the performances differ the most for short queries. The descriptor pruning experiments are only performed on GW20. On all other benchmarks the queries are given as individual word images which corresponds to the *horizontal or vertical* pruning strategy.

Due to its importance, the effect of vertical pruning is further analyzed in Table 15. A corner case arises if the descriptor size is larger than the height of the query word region. In order to prevent the exclusion of all descriptors, descriptors are only pruned up to a minimum number of descriptors per BoF vector, cf. Section 4.6.1. In comparison to the reference value of a minimum of 15 descriptors, Table 15 also shows the results for a minimum of a single descriptor. Furthermore, *none* refers to the configuration in which no pruning is performed, neither in horizontal nor in vertical direction. Both results are *significantly* worse ($\hat{p} \leqslant 0.05$) in comparison to the performance of the reference configuration on GW20.

It is important to note that the corner case, in which the descriptor size is larger than the width or height of the query region, can also occur on the other benchmarks in which the queries are provided as individual word images. In order to obtain at least a single BoF vector and at least a single descriptor per BoF vector, the descriptor size is dynamically adapted to the minimum of the query image width and the query image height in this case.

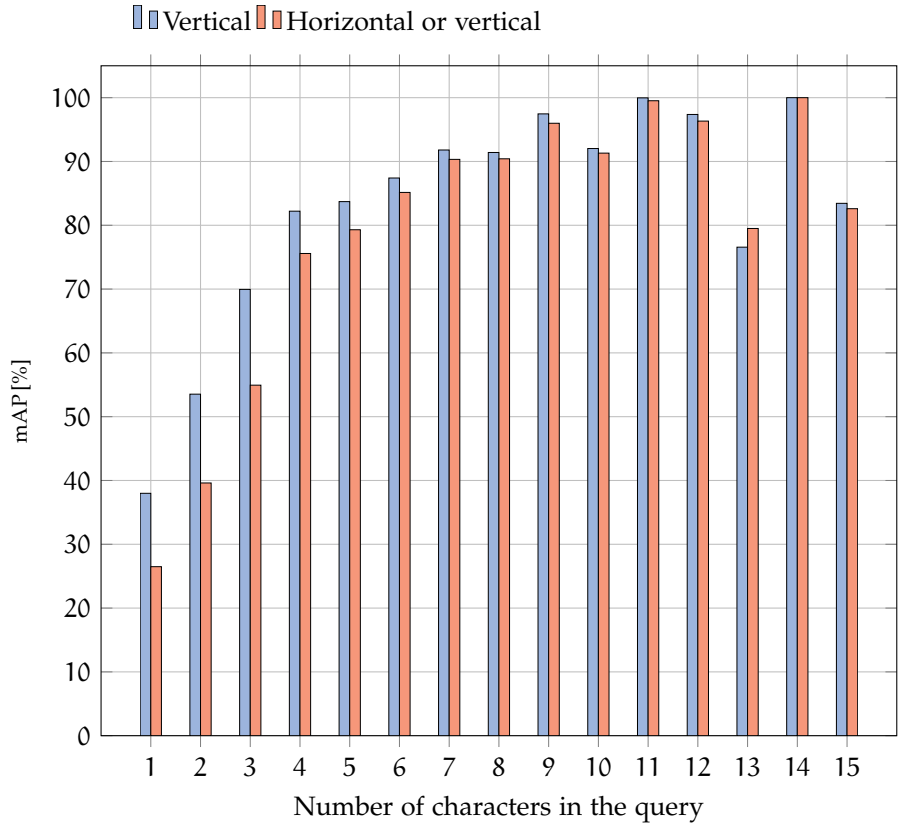


Figure 43: Effect of descriptor pruning for different query word lengths on the GW20 benchmark. It is important to note that only very few queries contribute to the mAP values for the query words with 13, 14 and 15 characters. In particular, the GW20 dataset contains only a single word with 14 characters, cf. Figure 37.

5.3.6 *Character models*

Character models allow for defining the query word model based on textual input, cf. Section 4.5.3. For this purpose, the query word HMM is dynamically created according to the characters in the query. The patch size is obtained based on the width and height estimates of the character models. In order to do so, a dataset with annotated training samples is required. In this regard it is important to note that the measured performance for all query-by-string experiments cannot be compared to the performance of the query-by-example experiments even though the benchmarks might be defined on the same document images. The following parameter evaluation is performed on the GW20 query-by-string benchmark that uses the *15-5* cross-validation, see Section 5.2.1. The method configuration is based on the same meta parameters that are used in the optimized query-by-example configuration, see Table 4. Meta parameters that are required for query-by-string word spotting are evaluated in Table 16.

Table 16: Query-by-string meta parameters

Context	Topology	Min. states	Height percentile	GW20 mAP [%]
Independent	Bakis	6	20	*69.5*
Dependent	Bakis	6	20	**83.3**
Independent	Linear	6	20	*70.1*
Dependent	Linear	6	20	82.0
Dependent	Bakis	3	20	*79.1*
Dependent	Bakis	9	20	82.9
Dependent	Bakis	6	5	82.4
Dependent	Bakis	6	50	*81.1*

Character models are obtained for each character that occurs in the training dataset. After the uniform initialization of the character HMMs, the training procedure consists of two stages. Table 16 shows the performances that are measured for the model obtained after the second stage. Except for the number of states, the meta parameters in Table 16 only affect the second training stage.

In the first stage, context independent models are estimated with the Baum-Welch algorithm until convergence, here for seven iterations. In this stage, the models always use a linear topology. The number of states per model is a meta parameter, cf. Table 16. In order to perform the initialization of the second estimation stage, the training data is automatically annotated on character level. The model obtained in the first stage is used for this purpose. Based on this annotation, the minimum number of BoF vectors that have been aligned with each model is recorded. The number of states for the models of the second stage is equal to this recorded number if the models use a linear topology. Greater flexibility is achieved with a Bakis topology. Due to the possibility to skip single states, the number of states per model is up to 50% higher than the minimum number of BoF vectors that have been aligned with each model. It is important to note that the number of states for the models of the first stage is a *lower bound* for the minimum number of BoF vectors that can be aligned with a model. Based on the initial alignment, the models are initialized according to Equation 47 in Section 4.5.2. As in the first stage, the Baum-Welch algorithm is iterated for seven times. Table 16 shows that 6 states per model in the first stage, i.e., the minimum number of states per model in the second stage, achieve a locally optimal performance. Using a Bakis topology in the second stage results in a higher mAP than using a linear topology. The absolute improvement of 1.3% mAP is not significant ($\hat{p} > 0.05$).

The choice between context-dependent and context-independent models in the second stage has the biggest performance impact ($\hat{p} \leqslant 0.05$). This is due to number of context-dependent models which is considerably larger than the number of context-independent models, in average 36 context-independent models in comparison to an average of 1,706 context-dependent models for the four training datasets in the 15-5 cross-validation of the GW20 query-by-string benchmark. This means that there are substantially more model parameters that have to be estimated. The estimation of such a large model inventory is only feasible with semi-continuous HMMs. In the application considered, the mixture model is not optimized along with the state-dependent parameters.

The width estimate that is required for each character model is given as the average width obtained after aligning the character models with the training data. The height estimate is based on the height distribution of the bounding boxes that are annotated with the corresponding characters. In order to be robust with respect to very small heights, the height estimate is obtained at a lower percentile of the distribution. Table 16 shows that a local optimum is obtained at 20%.

5.3.7 *Context models*

Context models are required for localizing the query in a patch more accurately. For this purpose, a *background* HMM represents arbitrary document image content and *whitespace* HMMs represent the left-side and right-side context of the text in the document images, see Section 4.5.4. In the compound query HMM, the context models surround the *query word* HMM, see Figure 25 in Section 4.5. For word spotting, it is important that the patch similarity scores are solely based on the query *word* HMM. The other models are not specific to the query and would, therefore, dominate the geometric mean of output probabilities, cf. Section 4.6.2.

The context models are used with the query word HMM irrespective of the query modality. The following evaluation is performed in the query-by-example scenario where no additional annotated training material is available besides the query word image since this is the targeted scenario of the proposed method. The experiments are based on the optimized query-by-example meta parameters, cf. Table 4.

Table 17 shows the influence of the context models for representing a BoF vector sequence from a patch. If *no* context models are used, the patches are represented only with the query word HMM. Thus, within a patch the query word cannot be localized. The patch-similarity score represents the *entire* patch and the decoding framework can, therefore, be considered as *static*.

One of the most important results in this evaluation is that the word spotting performance achieved with the *static* patch decoding frame-

Table 17: Context model evaluation

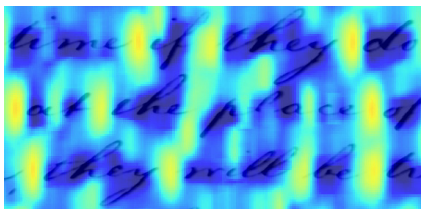
Context models	Iterations whitespace models	GW20 mAP [%]	BT50 mAP [%]
None	–	*71.4*	*52.3*
Background	–	74.3	56.3
Background, whitespace	0	**75.1**	**56.9**
Background, whitespace	3	75.2	57.0
Background, whitespace	5	75.2	57.0

work is *significantly worse* ($\hat{p} \leqslant 0.05$) than the performance of the reference configuration that uses the *dynamic* patch decoding framework (printed in a bold font in Table 17). The dynamic decoding framework achieves significantly better results because the similarity scores only represents the most likely occurrence of the query within the patch. Thus, the dynamic decoding framework can be seen as an adaptation of HMM-based word spotting on line level. Instead of representing the context of the query with a character-level recognition model, i.e., the filler model, the context models in the proposed method take advantage of the information that is available despite the lack of an annotated training dataset. The background HMM is defined by the output mixture model which is estimated in an unsupervised manner. The whitespace HMMs are based on whitespace hypotheses that are obtained in a *bottom-up* manner from text hypotheses.

Given the benefits of *dynamic* patch decoding, the influence of the different context models can be analyzed. For this purpose, Table 17 shows the results for an experiment where no whitespace models are used but only the background HMM. It is interesting to note that the performances are *not* significantly worse ($\hat{p} > 0.05$) than the performances of the reference configuration on both benchmarks. Thus, the possibility for *dynamic* patch decoding is more important than the particular modeling of the context. The use of whitespace HMMs is important if the whitespace models represent the immediate context of a relevant occurrence of the query word better than the generic background model. The *partial* observation sequence, that is aligned with the states of the query word HMM, depends on the optimal path for the *entire* observation sequence which is computed with the Viterbi algorithm. This makes the similarity score more specific since the similarity score is based on the partial output probabilities for the BoF vectors that have been aligned with the query word model.

While the background HMM is directly defined by the mixture component prior probabilities, the whitespace HMMs are estimated in analogy to the query word HMM for query-by-example or query-by-string, depending on the word spotting scenario. In this regard an interesting observation was made with respect to the number of Baum-

Left-side whitespace model



Right-side whitespace model

Figure 44: Comparison of the patch-similarity scores for the whitespace models that represent the left-side context and the right-side context of text in the document images. The similarity scores are obtained by sliding a patch over the dense grid of visual words. For each patch position, the similarity is based on the optimal output probability for the BoF vector sequence from the patch. The patch width and the patch height are given by the average width and the average height of the corresponding left-side or right-side whitespace regions. The colors visualize similarity scores in the logarithmic domain and correspond to the same values as in Figure 41. Blue refers to low similarity and yellow towards orange refers to high similarity with respect to the whitespace models.

Welch iterations in the query-by-example scenario, cf. Table 14. While it is reasonable to omit Baum-Welch training for estimating the query word HMM with just a single example, this approach cannot be motivated for the estimation of whitespace models in the same way. For this reason, a parameter evaluation for the number of training iterations is shown in Table 17. Since the performance differences are only marginal ($\hat{p} > 0.05$), the estimation of whitespace models can be performed in analogy to the query word models. This way, an additional meta parameter can be avoided.

A qualitative impression of the whitespace models is given in Figure 44. The figure shows patch-similarity scores for *whitespace spotting*. The whitespace HMMs are used as query models in a static patch-based decoding framework for this purpose. The patch sizes are given by the average size of the left-side whitespace hypotheses and the average size of the right-side whitespace hypotheses. Figure 44 shows that the document regions that achieve high similarity scores are mostly similar in comparison between the left-side and right-side models.

5.3.8 *Patch-based retrieval*

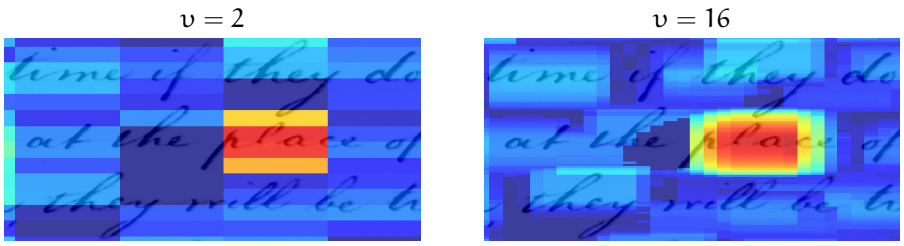
Word spotting with BoF-HMMs is based on a dense patch retrieval framework. The patch positions are aligned with the dense grid of visual words, cf. Section 4.6.1. For each patch, the similarity with respect to the query word model can be obtained with the Viterbi algorithm. However, the evaluation with the Viterbi algorithm for *all* possible patch positions is computationally infeasible. Patches that

Table 18: Patch-based decoding

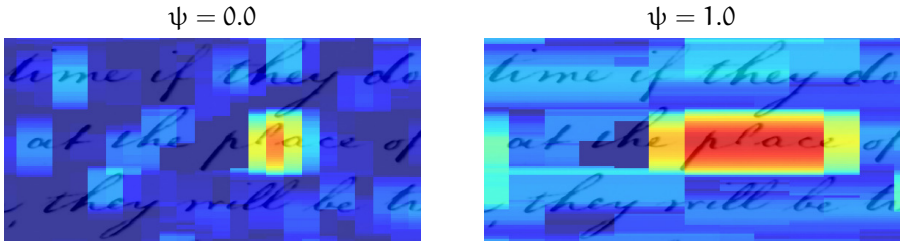
Patch sampling rate υ	Patch width expansion ψ	GW20 mAP [%]	BT50 mAP [%]
2	0.5	*65.0*	*51.2*
4	0.5	74.3	56.5
8	0.5	**75.1**	**56.9**
16	0.5	74.5	56.4
8	0.0	*67.7*	*49.0*
8	1.0	*73.6*	53.7

are potentially relevant with respect to the query can be obtained efficiently with mixture component voting, cf. Section 4.6.3. For this purpose, the mixture component voting scheme is based on the *same* patch-framework which is used for Viterbi decoding. The patch resolution depends on the patch size and on the *sampling rate* meta parameter υ, see Equation 49 in Section 4.6.1. In order to increase the method's robustness with respect to word size variabilities, the query word context, cf. Section 4.5.4, is taken into account in order to model the appearance of the query word *within* a patch. The patch width, which is obtained from the query word model, is increased according to the *patch width expansion* factor ψ for this purpose, cf. Figure 31 in Section 4.6.2. Table 18 shows the evaluation of the patch sampling rate and the patch width expansion meta parameters. A qualitative impression of the resulting patch similarity scores is given in Figure 45. The patch retrieval framework is independent of the query modality. The following results are obtained for the query-by-example scenario.

The patch sampling rate controls the patch resolution in the patch-framework. More patches are sampled for smaller query words than for larger query words due to the dependence on the patch size. For accurate detections it is important that for each instance of the query word in a document image, there exists a patch that is centered over the corresponding document image region. In this regard the horizontal positioning of the patch is less important as the vertical positioning. As long as the patch encloses the instance of the query word, the accurate start and end positions within the patch are determined with the Viterbi algorithm on frame level. If the patches are inaccurately positioned in vertical direction, the upper or lower context of the query word instance is included in the BoF sequence representation. It has already been shown that this affects the performance significantly, cf. Table 15. The evaluation of the sampling rate in Table 18 confirms this behavior. On both benchmarks considered, only the performance for the sampling rate $\upsilon = 2$ is *significantly worse* ($\hat{p} \leqslant 0.05$) than the performance of the reference configuration which uses the locally optimal sampling rate $\upsilon = 8$. Figure 45a illustrates the patch



(a) Patch sampling rate



(b) Patch width expansion

Figure 45: Patch-based decoding. The patch similarity scores illustrate the effect of different parameterizations. The patch sampling rate (a) controls the patch resolution. The patch width expansion (b) is important for coping with word size variabilities. The larger the expansion factor, the less specific are the similarity scores with respect to the patch positions. The colors represent patch similarities in the logarithmic domain and correspond to the same values across all images (including the similarity scores in Figure 41 and 44). Blue corresponds to low similarity and red corresponds to high similarity with respect to the query word model.

similarity scores for two different sampling rates. It can be seen that the patch resolution obtained for $\upsilon = 2$ is very coarse.

The patch width expansion is required for *dynamic* patch decoding, also cf. Section 5.3.7. The larger the expansion factor the larger is the word size variability that can potentially be handled. However, at the same time, the similarity scores also become less specific to the position of the corresponding patch in the document image. Further, the capability to generalize across large word size variabilities is inherently limited if the query word model is estimated from just a single example. Table 18 shows that a locally optimal trade-off is found for $\psi = 0.5$ for the GW20 benchmark and the BT50 benchmark. The performance differences are mostly significant ($\hat{p} \leqslant 0.05$, indicated with an italic font in Table 18). An impression of the effect on the patch similarity scores for two different patch width expansion factors is given in Figure 45b. The influence on the specificity can be observed for high similarity scores as well as for low similarity scores.

Finally, an architectural design choice is analyzed in Table 19. Computing the patch similarity with the *partial output probabilities* for the BoF vectors that are generated with the query word HMM most likely, is motivated by the limited visual variability that the method can

Table 19: Score normalization

Similarity scores	GW20 mAP [%]	BT50 mAP [%]
Partial output probability	**75.1**	**56.9**
Background model normalization	*73.5*	53.9

cope with in the target scenario. The limitation is due to the lack of an annotated training dataset for query-by-example word spotting, cf. Section 4.6.2.

Alternative approaches for obtaining similarity scores for word spotting with HMMs have been proposed in [RP09b] and [FKF+12]. Both approaches are based on approximating the posterior probability for the query word (log-odds scoring). The probability for the query model is normalized for this purpose, cf. Equation 10 in Section 3.3.2. The difference between the two approaches is that a character-level recognition model, i.e., a filler model, is used for score normalization in [FKF+12] whereas a background model is used for score normalization in [RP09b]. The background model is used due to the lack of annotated training data. Since only a segmentation-*based* scenario is considered in [RP09b], a *background model normalization* for the segmentation-free, patch-based decoding framework, cf. Section 4.6.2, is obtained by combining both normalization approaches, see Table 19. For this purpose, the filler model normalization is adapted such that the recognition model is replaced with the background model, cf. Equation 12 in Section 3.3.2.

The background model normalization makes the patch similarity scores independent of the *typical* visual appearance of the document images. However, this also causes problems if irrelevant document image regions are represented with similarly low scores by the query word model *and* the background model. The score normalization emphasizes these regions which leads to false positives. Table 19 shows that using the *partial output probability*, as proposed in Section 4.6.2, is significantly better ($\hat{p} \leqslant 0.05$) than using the background model normalization on the GW20 benchmark. The improvement of 3% mAP over the background model normalization on the BT50 benchmark is *not* significant ($\hat{p} > 0.05$).

5.3.9 *Retrieval efficiency*

Efficiency is achieved by decoding in two stages, see Section 4.6.4. Mixture component voting, cf. Section 4.6.3, retrieves potentially relevant regions at high *recall* and with high *computational efficiency*. Viterbi decoding allows for re-ranking and refining these regions with high *average precision* at the *cost* of high *computational complexity*. Since both

Table 20: Viterbi-decoding efficiency (second stage only)

Patch framework	GW20 LR	BT50 LR
Regular	71,502	34,880
Line hypotheses	68,335	29,178

Table 21: Viterbi-decoding performance (second stage only)

Patch framework	GW20 mAP [%]	BT50 mAP [%]
Regular	75.4	57.7
Line hypotheses	**75.4**	**57.6**

decoding stages are using the same model and the same patch-based framework, the parameter configurations that have been evaluated in the previous sections mostly affect *both* decoding stages. Exceptions are the meta parameters that are specific to methodologies which are only required for decoding with the Viterbi algorithm, like the application of context models and score normalization. Architectural design choices that are required for the two-stage integration will be evaluated with respect to accuracy *and* computational efficiency in this section. For this purpose, the influence of *line hypotheses* and *soft non-maximum suppression* (NMS) will be discussed. Finally, the decoding stages will be evaluated individually and jointly in order to analyze the *two-stage integration*. The experiments are based on the optimized query-by-example configuration, cf. Table 4.

All of these design choices have in common that they affect the number of patches P that have to be processed in the second decoding stage. If relevant patches are excluded in the first stage, there is no chance for compensation in the second stage. Thus, a trade-off with respect to accuracy and computational complexity must be found.

Line hypotheses

Mixture component voting is based on indexing frame representations from line hypotheses. Given the height of the query, line hypotheses reduce the search space in comparison to the number of dense patches that are *regularly* sampled from the *entire* document, see Table 20. The generation of line hypotheses is designed such that many *overlapping* line hypotheses are generated for *each* text hypothesis with a height which is *similar* to the query height, cf. Section 4.2.2 and 4.6.1. This way, the average number of patches LR can be reduced by 4.4% on the GW20 benchmark and by 16.3% on the BT50 benchmark. Table 20 shows the average number of patches for Viterbi decoding

using a *regular* grid of dense patches over the entire document and the patch framework that samples patches from line hypotheses. It is important to note that using line hypotheses does not affect retrieval performance as shown in Table 21. The differences between relative reduction of the numbers of patches on GW20 (4.4%) and on BT50 (16.3%) can be explained with the writing which is less dense on BT50. This means that the distances between adjacent text lines are typically larger in the BT50 documents than in the GW20 documents.

Since mixture component voting is based on line hypotheses, the effect of using line hypotheses is not directly evaluated in the two-stage decoding framework. Nevertheless, the results from Table 20 and 21 are still relevant for two-stage decoding. They show that the document regions which are excluded due to the line hypotheses do typically *not* contain any occurrences of the query word. Furthermore, excluding regions from the first stage will typically reduce the number of re-ranked patches in the second stage. This is due to the selection of *locally optimal* patches in the first stage which also produces detections in document regions that would not have been represented by line hypotheses.

Soft NMS

The selection of locally optimal patches in the first stage influences retrieval performance and computational efficiency. Mixture component voting is very sensitive to *any* document image regions that are visually similar to the query which leads to *false positives*. Therefore, the size of the neighborhood for selecting locally optimal patches is important. A large neighborhood leads to fewer detections in the first stage but increases the risk of suppressing relevant patches. Due to the sensitivity of the mixture component voting approach, a region which is only *partially* similar to the query can still obtain a high similarity score, e.g., due to individual characters.

In Section 4.6.4 this challenge has been addressed with *soft* NMS. The key idea is to smooth the mixture component voting scores with a Gaussian *before* applying a maximum filter that is *smaller* than the Gaussian filter for NMS, cf. Equation 66 in Section 4.6.4. Strong local optima in close proximity to each other are more likely to be detected with a small maximum filter mask. The Gaussian is applied in order to decrease the number of patches that are detected in the first stage and have to be processed in the second stage. Table 22 and 23 show the effect of soft NMS after the first decoding stage in order to achieve the two-stage integration. The meta parameter ζ controls the filter size. In this regard, $\zeta = 2$ results in a filter that corresponds to the patch size. Applying NMS with a maximum filter with $\zeta = 2$ does not allow for overlapping patches. Patches are allowed to overlap up to 50% with a maximum filter that uses $\zeta = 1$.

Table 22: Soft NMS efficiency

Maximum filter size ζ	Gaussian filter size ζ	GW20 P	BT50 P
1	2	1,667	821
1	–	4,804	2,106
2	–	1,614	735

Table 23: Soft NMS performance

Maximum filter size ζ	Gaussian filter size ζ	GW20 mAP [%]	BT50 mAP [%]
1	2	**75.1**	**56.9**
1	–	74.9	55.9
2	–	73.7	54.6

Table 22 shows the average numbers of patches P that have to be re-ranked with the Viterbi algorithm for different NMS configurations on the GW20 and the BT50 benchmarks. It can been seen that the number of patches that have to be re-ranked with soft NMS (first row) is only marginally larger than using NMS with a large maximum filter ($\zeta = 2$). In comparison, the number of patches that have to be re-ranked with a small maximum filter ($\zeta = 1$) is considerably larger. At the same time, soft NMS achieves the highest retrieval performance as shown in Table 23. On GW20 the performance is even *significantly better* ($\hat{p} \leqslant 0.05$) than using (hard) NMS with a maximum filter with $\zeta = 2$. It is important to note that the results reported in Table 22 and 23 are obtained with the optimized query-by-example configuration that uses a 3×3 re-ranking mask. Thus, the number of patches in Table 22, i.e., patches that have to be re-ranked, is around nine times larger than the number of patches that are detected in the first stage. For example, 186 patches are retrieved in the first stage on the GW20 benchmark in average.

Two-stage integration

The integration of the two decoding stages is based on re-ranking potentially relevant patches from the first stage in the second stage. Therefore, the computational complexity for the two-stage integration can be obtained by combining the computational complexity of mixture component voting in the first stage with the computational complexity of Viterbi decoding in the second stage as shown below, cf. Section 4.6.4.

$$O(S\hat{M}_{HMM}\hat{M}_{IFS} + PF\hat{M}\hat{S}^2)$$

Table 24: Average complexity measures

Measure	GW20	BT50
HMM states per query word model (S)	44	50
Mixture components per HMM state ($\hat{M}_{HMM}$)	14	18
Mixture components per IFS entry ($\hat{M}_{IFS}$)	2,210	1,536
Re-ranked patches per document (P)	1,667	821
Frames per patch (F)	97	113
Mixture components per frame ($\hat{M}$)	10	13
Possible transitions per HMM state ($\hat{S}$)	2	2
Dense patches per document (LR)	68,335	29,178
Line hypotheses per line height (L)	949	713
Frames per line hypothesis (T_l)	667	611

The first part of the sum describes the complexity of mixture component voting while the second part refers to the application of the Viterbi algorithm to P patches. If the two decoding stages are applied on entire documents individually, the computational complexity for mixture component voting is $O(S\hat{M}_{HMM}\hat{M}_{IFS})$ and the computational complexity for Viterbi decoding is $O(LRF\hat{M}\hat{S}^2)$. In order to evaluate the average computational complexity for word spotting on the GW20 benchmark and the BT50 benchmark, the average complexity measures for the optimized query-by-example configuration on both benchmarks are summarized in Table 24. The values are obtained by averaging over all queries in the corresponding benchmarks.

Table 24 shows that the results for the two benchmarks are consistent with each other. For two-stage decoding, the average number of entries in the inverted file structure ($\hat{M}_{IFS}$) and the average number of patches for re-ranking (P) are the dominating values. It should be noted, that the number of patches for re-ranking is more important since the complexity for Viterbi decoding is influenced by four factors whereas the complexity for mixture component voting is only influenced by three factors. However, the values are orders of magnitudes smaller than the average number of dense patches (LR) that is dominating the computational complexity for Viterbi decoding on the *entire* document image.

In order to assess the decoding approaches in general, the computational complexity alone is insufficient. Since the different methods produce different retrieval results, the retrieval performance has to be considered as well. For this purpose, Figure 46 plots the average computational complexity against the mAP for different decoding strategies on the GW20 benchmark. These strategies include the two decoding stages individually (denoted as IFS and VT) and their integrations (denoted as IFS-VT) where different sizes of the re-ranking mask

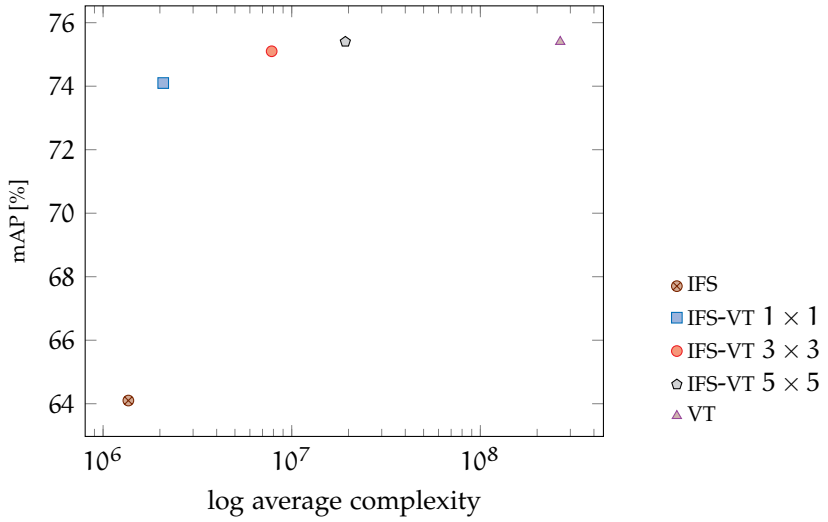


Figure 46: Average computational complexity versus retrieval performance on the GW20 benchmark. The markers indicate complexity and mAP for different BoF-HMM retrieval configurations. The first decoding stage is denoted as IFS and the second decoding stage is denoted as VT. The size of the re-ranking mask is specified if the two stages are applied jointly. It is important to note that the minimum and maximum values are adapted to the values of the data points on both axes. The average complexity values are shown in the logarithmic domain.

are used. It can be seen that mixture component voting achieves the lowest computational complexity along with the lowest mAP. The benefits of using Viterbi decoding in the second stage are considerable (IFS-VT 1×1). The increase in the computational complexity comes with a large improvement of the retrieval performance. Further, the mAP increases with a 3×3 re-ranking mask, which corresponds to the optimized query-by-example configuration. Afterwards, the improvements are only marginal. It is important to note that the average computational complexity increases considerably with each of these decoding methods. One of the most important results is that the reference configuration (IFS-VT 3×3) comes very close to the mAP of the brute-force search with the Viterbi algorithm (VT) while the computational complexity is orders of magnitudes smaller.

The advantage of evaluating retrieval efficiency in terms of computational complexity is that the measurement is independent of the implementation, programming language and hardware. The disadvantage is that the absolute complexity values are hard to interpret alone but have to be interpreted in comparison to each other. In order to allow for an easier assessment of the applicability in practice, the retrieval efficiency is also measured in average milliseconds (ms) per query and document as shown for the GW20 and the BT50 benchmarks in Table 25. For this purpose, it is assumed that the line representations and the corresponding inverted indices are stored in memory.

Table 25: Two-stage decoding efficiency (query per page)

Method	Re-ranking mask	GW20 time [ms]	BT50 time [ms]
Mixture voting (IFS)	–	24	19
Viterbi decoding (VT)	–	17,886	14,075
Two-stage (IFS-VT)	1×1	477	293
Two-stage (IFS-VT)	3×3	939	717
Two-stage (IFS-VT)	5×5	1,908	1,542

Table 26: Two-stage decoding performance

Method	Re-ranking mask	GW20 mAP [%]	BT50 mAP [%]
Mixture voting (IFS)	–	*64.1*	*34.1*
Viterbi decoding (VT)	–	75.4	57.6
Two-stage (IFS-VT)	1×1	74.1	55.2
Two-stage (IFS-VT)	3×3	**75.1**	**56.9**
Two-stage (IFS-VT)	5×5	75.4	57.3

Mixture component voting is implemented in *C++*, the Viterbi algorithm is implemented in the HMM toolkit *ESMERALDA* in *C* and the application of both components for segmentation-free word spotting is implemented in *Python*. The processor is an *Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz*.

The average retrieval times in Table 25 are approximately consistent with the average computational complexities reported in Figure 46. The large differences between mixture component voting and two-stage decoding with a 1×1 re-ranking mask can be explained with the different execution times achieved in *C++* and *Python*. While the retrieval times for two-stage decoding can be considered as demanding, retrieval times for processing *all* patches with Viterbi decoding can be considered as infeasible.

In contrast to the retrieval times, the *query estimation* time does not scale with the number of documents. On the GW20 benchmark, the average time for obtaining the query model is 202 milliseconds. Thus, overall, the time can be neglected.

It is important that the retrieval times in Table 25 are recorded for processing a single document image by using a single thread in a single process. However, the time for searching an entire document collection can be improved if more than one document image is processed in parallel which is directly possible with modern multi-core CPUs. For example, the specified workstation CPU has 12 physical cores. Thus, 10 pages can be processed in parallel without affecting

the individual retrieval times. Regarding the *query-per-page* retrieval times in Table 25, it should be noted that the CPU is intended for parallel processing rather than for running few processes with high speed.

In addition to the performance values in Figure 46, the mAP values for the different decoding methods on the GW20 and the BT50 benchmarks are presented in Table 26. An expected result is that mixture component voting performs *significantly worse* ($\hat{p} \leqslant 0.05$) than the reference configuration (printed in a bold font) on both benchmarks. However, the improvement achieved with a 3×3 re-ranking mask over the performance achieved with a 1×1 re-ranking mask is *not* significant ($\hat{p} > 0.05$). Therefore, the two-stage integration with a 1×1 re-ranking mask achieves the best trade-off between computational efficiency and retrieval performance. The reference configuration emphasizes retrieval performance.

Since retrieval in the two-stage process is based on the results in the first stage and the two-stage integration (IFS-VT 3×3) achieves similar results as processing the entire documents with Viterbi decoding (VT), mostly all relevant document regions that can be detected with the brute-force search (VT) can also be detected with mixture component voting (IFS). Thus, the first stage performs retrieval with high speed and high recall. The mean recall for the complete retrieval lists on GW20 is 88.7% with mixture component voting alone and 89.7% with the two-stage integration (IFS-VT 3×3). The difference shows that even patches which are positioned inaccurately in the first stage, can still be refined successfully in the second stage.

Finally, the memory requirements for storing line representations and inverted indices have to be analyzed. As presented at the end of Section 4.6.3, the memory complexity is $5 \cdot O(LT_l\hat{M})$. Given the average complexity measures in Table 24, the average memory requirement per document image and line height can be computed for the optimized query-by-example configuration. Assuming that each number is represented with 32 bit, storing the line representation and the inverted index requires an average of 121 megabytes on GW20 and an average of 108 megabytes on BT50 per line height and document. Since the average number of different line heights per document is 55 on GW20 and 45 on BT50, several terabyte of disk space with very high read-speed are required in order to store all precomputed document representations for a large document collection of several hundred pages. Given that many line heights will never be used for word spotting at all, a simple approach that will not affect retrieval performance is to compute line representations on demand.

Table 27 shows the average processing times for computing line representations with the visual-word mixture model and the inverted index for the line hypotheses of a single line height per document. The visual-word mixture model requires the computation of a his-

Table 27: Representation efficiency (line height per page)

Method	GW20 time [ms]	BT50 time [ms]
Visual-word mixture model	1,123	986
IFS indexing	259	229

togram for each frame in all line hypotheses. The inverted index is built by iterating over all non-zero mixture components in all frames while recording the spatial frame coordinates as well as the mixture component probabilities in the IFS entries for the corresponding mixture components. The processing times are obtained under the same prerequisites as the processing times in Table 25. The algorithms are implemented in *C++* and are run in a single thread. If line representations and inverted indices are computed on demand, the average time for searching a document image is obtained by adding the processing time for the corresponding decoding method in Table 25 and the processing times in Table 27. It should be noted that the on-demand computation can be considered as computationally infeasible for any other mixture model than the visual-word mixture model. In comparison, the evaluation of the EDCM mixture model with 512 mixture components takes in average 35 seconds per line height and document on GW20. This processing time also includes the time which is required for computing BoF representations for each frame.

5.3.10 *Region snapping*

Region snapping is applied in cases where highly accurate detections are required, cf. Section 4.6.5. The document regions obtained after Viterbi decoding are refined for this purpose. In order to do so, these spotted region bounds are snapped to the bounds of text hypotheses. Text hypotheses are extracted from the document images in a bottom-up manner and are, therefore, independent of the query, see Section 4.2.1. Since text hypotheses can represent arbitrary parts of text and also background clutter, the spotted region is *not* snapped to a single text hypothesis but each region boundary (left, right, top, bottom) is snapped to a corresponding region boundary of any suitable text hypothesis. The selection of suitable hypotheses is performed according to different criteria which will be evaluated in the following. The experiments are based on the optimized query-by-example configuration, cf. Table 4. Quantitative results are reported for the BT50 benchmark in Table 28. In this regard the high detection accuracy is demonstrated by adapting the relevance criterion in the evaluation protocol, cf. Section 5.1.1. In addition to reporting results for 50% IoU ($\iota = 0.5$), results are also reported for 70% IoU ($\iota = 0.7$). Only a quali-

Table 28: Region snapping

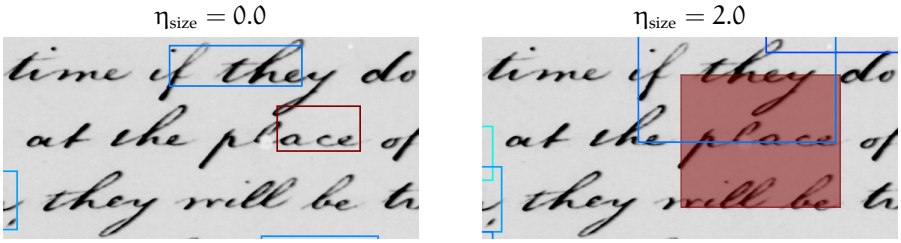
Region snapping area η_{size}	Region snapping weight η_{weight}	BT50 mAP [%] $\iota = 0.5$	BT50 mAP [%] $\iota = 0.7$
–	–	56.4	*46.2*
0.0	0.1	*39.1*	*21.6*
1.0	0.1	**56.9**	**52.7**
2.0	0.1	57.2	53.1
1.0	0.0	56.8	53.0
1.0	0.5	56.4	51.3
1.0	1.0	56.3	51.2

tative evaluation is performed on GW20, see Figure 47. This is due to inaccurate bounding box annotations of the benchmark.

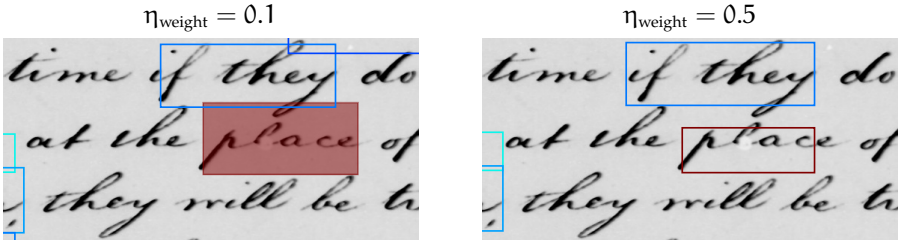
Table 28 shows the results for the *region snapping area* meta parameter and the *region snapping weight* meta parameter. Depending on the height of the spotted region, the snapping area parameter is a factor that defines the document image context for selecting suitable hypotheses. The bounds that the region will be snapped to are determined according to a distance criterion and a size criterion. The snapping weight parameter controls the influence of both criteria in the joint indicator. Table 28 shows that region snapping does generally *not* make a significant difference ($\hat{p} > 0.05$) for 50% IoU on the BT50 benchmark. However, an important result is that the absolute difference of 6.5% mAP between the performance *with* region snapping (reference configuration, printed in a bold font) and *without* region snapping *is significant* ($\hat{p} \leqslant 0.05$) for 70% IoU. With respect to the effect of the meta parameters, the only parameterization that achieves a *significantly worse* result than the reference configuration is obtained for the *region snapping area* with $\eta_{size} = 0.0$. This is due to the limited flexibility if only text hypotheses are selected that are located within the region that was obtained with Viterbi decoding.

However, the parameter evaluation for BT50 is not necessarily representative. Since the text hypotheses are not selected according to similarity to the query, the performance strongly depends on the characteristics of the text in the document images. The results for the parameterizations reported in Table 28 can be explained with the typically large spacing between the text lines in the BT50 dataset.

The qualitative evaluation on GW20 demonstrates the challenges for a writing style which is more dense. Figure 47a shows that the size of the snapping area can have unwanted effects if the area is either too small or too large. With respect to the region snapping weight, Figure 47b shows that the aspect ratio of the snapped regions is more uniform if the influence of the size criterion becomes stronger. This



(a) Region snapping area



(b) Region snapping weight

Figure 47: Region snapping. The figure shows a qualitative parameter evaluation on the GW20 benchmark. Only a single parameter is changed at a time. The reference configuration is the same as in Table 28, i.e., $\eta_{size} = 1.0$ and $\eta_{weight} = 0.1$. The region snapping area (a) defines the size of the context for selecting text hypotheses. The region snapping weight (b) controls the emphasis of the text hypothesis size criterion. The colored rectangles indicate the snapped regions. Regions that are filled in red are relevant according to the 50% IoU measure which is used in the GW20 evaluation.

can be an advantage for being robust with respect to text hypotheses that represent background clutter but can be a disadvantage for detecting long ascenders and descenders accurately, cf. Figure 47b. The parameterization for the reference configuration, i.e., $\eta_{size} = 1.0$ and $\eta_{weight} = 0.1$, emphasizes the distance criterion since the snapped region will typically be more similar to the region that has been retrieved with the Viterbi algorithm. This region is more likely to be relevant to the query than the regions in the local neighborhood. Since these regions have been suppressed with NMS they are less similar with respect to the query model.

5.4 RESULTS

The results for segmentation-free word spotting with BoF-HMMs are presented on five benchmark datasets, see Section 5.2. For this purpose, the meta parameters are chosen according to the evaluations in Section 5.3, cf. Table 4. The only parameter that requires dataset specific adjustments is the SIFT descriptor size, see Section 5.3.3. A method for obtaining a size estimate is based on the typical height of the text core area. This automatic estimate is used in order to apply BoF-HMMs to an *unknown* dataset (Section 5.4.1).

Based on the optimized query-by-example configuration, the focus is to compare the performance achieved with the BoF-HMM to the performance achieved by a *baseline method* and to the performance achieved by related methods from the literature. The baseline method allows for comparing BoF-HMMs to a different model without changing the underlying features and the patch-based decoding framework (Section 5.4.2). The comparison to related methods from the literature is possible due to the use of established word spotting benchmarks. The *George Washington* (GW20) dataset allows for comparing word spotting performance in the query-by-example scenario and in the query-by-string scenario (Section 5.4.3). Furthermore, the keyword spotting competitions attracted a lot of attention in the word spotting community and allow for comparing performance with many word spotting methods. The *Jeremy Bentham* manuscripts have been considered in the 2014 competition (BT50) and in the 2015 competition (BT70). Word spotting performance will be compared in the query-by-example scenario (Section 5.4.4). Query-by-example and query-by-string benchmarks from the 2016 competition are considered for the comparison on the *Konzilsprotokolle* (KP20) dataset and on the *Botany* (BO20) dataset (Section 5.4.5).

The differences between the BoF-HMM performance and the performance of the baseline method are analyzed with permutation tests ($\hat{p} \leqslant 0.05$), see Section 5.1.2. Permutation tests cannot be applied for analyzing performance differences with related methods. The individual query result are unavailable for these methods. Apart from the quantitative evaluation, an impression of the BoF-HMM performance will be given in a qualitative evaluation. The top-ten retrieval results for a query word image are shown for each dataset.

5.4.1 *Descriptor size estimation*

Obtaining a locally optimal SIFT descriptor size requires an annotated validation set that is representative for the corresponding document collection. The GW20 benchmark and the BT50 benchmark have been used for this purpose in Section 5.3. Since the BT70 dataset belongs to the same document collection as the BT50 dataset, the evaluation on BT50 can be considered as representative for BT70. The scenario in which *no* annotated validation set is available will be simulated for the KP20 and the BO20 benchmarks. This can be seen as the typical scenario when a collection of document images is explored with automatic methods for the first time. In order to make BoF-HMMs directly applicable in this *targeted* scenario, an automatic estimation of the SIFT descriptor size is required. Since the descriptor size is related to the typical text height, an estimate of the typical height of the text core area can be used. For this purpose, it is assumed that the text is mostly written in lines. In a text line, the core area is the zone where

Table 29: Text core height estimation

Percentile $q_{profile}$	GW20 height	BT50 height	BT70 height	KP20 height	BO20 height
20	22	34	33	52	53
25	24	35	35	59	55
30	26	35	35	66	53

Table 30: SIFT descriptor size estimates

Estimate	GW20	BT50	BT70	KP20	BO20
Text core height	24	34.7	34.3	59	53.7
Descriptor size	24	32	32	56	56

most character cores are located, i.e., above the character descenders and below the character ascenders, cf. [FB14, Fig. 12.3]. The descriptor size refers to the edge length of the square descriptor area in image pixels.

The height estimate is based on horizontal projection-profiles, cf. [Kis14]. A projection-profile is computed for each gray-scale document image. Provided that the pen-stroke is represented with low image intensity values, text lines will be represented as valleys and document background regions will be represented as elevations in the projection-profile. A simple approach to text line detection is to threshold the projection-profile at a *lower* percentile $q_{profile}$ of the distribution of projection-profile values. The height estimate for each detection is given by the run-length of projection-profile values that fall below the threshold obtained at percentile $q_{profile}$. It is important to note that the estimated heights *strongly* depend on the chosen percentile. For this reason, the global height estimate is obtained at an *upper* percentile $q_{height} = 100 - q_{profile}$ of the estimated text height distribution of the height estimates across all documents. Choosing percentile q_{height} in dependence of $q_{profile}$ leads to a larger global height estimate if the height distribution is dominated by smaller heights and a smaller global height estimate if the height distribution is dominated by larger heights.

Table 29 presents the global height estimates for all benchmarks considered. The evaluation of three different percentiles shows that the estimation leads mostly to stable results. The largest deviations can be observed for the KP20 benchmark which is due to the frequent and typically long ascenders and descenders in German Kurrent. For each benchmark, the descriptor size estimate is obtained by rounding the average of the three height estimates to any of the descriptor sizes $\{16, 24, 32, \dots\}$, as shown in Table 30. It can be noted that the descriptor size is underestimated on GW20 in comparison to the locally

optimal descriptor size 48 that has been determined in Table 6. This is due to the low visual variability in the GW20 document images that allows for performance improvements with larger and, therefore, more specific descriptors as explained in Section 5.3.3. Since the GW20 dataset has been used for optimizing meta parameters in Section 5.3, also the following results on GW20 are obtained with the locally *optimal* descriptor size. As expected, the document images in the BT50 and BT70 benchmarks have characteristics which are very similar to each other. The descriptor size estimates are consistent with the locally optimal descriptor size that was obtained for the BT50 benchmark. Results on the KP20 and BO20 datasets are obtained with the estimated descriptor size 56, see Table 30. An evaluation of the descriptor sizes for BT70, KP20 and BO20 can be found in Table 48 in Section A.2.

5.4.2 *Baseline method*

Representing a word image with a BoF-HMM can be seen as a dynamic probabilistic extension of a spatial pyramid, see Section 5.3.5. The baseline method is based on a spatial pyramid and uses a temporal cell structure that resembles the temporal modeling structure of the BoF-HMM. For this purpose, the baseline method uses the same descriptors, the same visual vocabulary, the same patch sizes and the same patch sampling steps which are used for segmentation-free word spotting with the optimized BoF-HMM query-by-example configuration. The query word image and all patches are represented with the temporal spatial-pyramid adaptation. Similarity of the patch representations with respect to the query word representation is computed with cosine similarity, cf. Section 3.3.1. In analogy to word spotting with BoF-HMMs, cf. Section 4.6.4, the patch score matrix is smoothed with an anisotropic Gaussian that corresponds to the size of the query word image, cf. [RAT+15a]. Locally most similar patches are retrieved with NMS such that the retrieved patches do not overlap with each other. Patches are retrieved directly *without* an additional patch refinement step.

While the patch-based framework of the baseline method corresponds to the patch-based framework that is used for word spotting with BoF-HMMs, an important question is how to adapt the spatial pyramid to the characteristics of text. Two spatial pyramid extensions which have a large impact on word spotting performance [ART+15] are evaluated in Table 31. The *cell configuration* controls the temporal resolution of the representation and the *power normalization* controls the influence of large vector components. Since the baseline method is intended as a reference for word spotting *performance*, computational efficiency is not a major consideration. Therefore, the evaluation is performed on the BT50 benchmark which contains only a relatively small number of queries, see Section 5.2.2.

Table 31: Baseline method optimization

Cell configuration	Power normalization α	BT50 mAP [%]
$(1 \times 1, 1 \times 2)$	0.35	*39.3*
(1×4)	0.35	**45.6**
(1×4)	0.1	45.6
(1×4)	1.0	*41.0*

The segmentation-free approach for the baseline method is mostly inspired by the patch-based word spotting method that was presented in [RAT+15a]. Document image regions are represented with a temporal pyramid with a *global* cell on the first level and two cells that split the region into *left* and *right* on the second level. However, following the results in [ART+13; ART+15] and also [SF17], a larger number of cells leads to a better temporal resolution and better word spotting performance. Table 31 shows an evaluation where the pyramidal cell configuration used in [RAT+15a] is compared to a temporal cell configuration with just a single level but four consecutive cells. The design is inspired by the BoF-HMM that also does not use a pyramidal structure but represents the temporal structure with states. Table 31 shows that the temporal cell structure *significantly* outperforms the pyramidal cell structure on the BT50 query-by-example benchmark. According to the results in [ART+15], it can be expected that increasing the number of cells beyond four, will also increase the word spotting performance. However, increasing the number of cells has a substantial impact on the dimensionality of the holistic representation. With the visual vocabulary from the optimized BoF-HMM configuration, each patch is represented with a $4 \cdot 4096 = 16,384$ dimensional vector. Provided that 29,178 patches per page have to be processed for word spotting on BT50 in average, see Table 24 in Section 5.3.9, this dimensionality can already be considered as extremely demanding.

Another spatial pyramid extension which has been applied to word spotting is power normalization [ART+13; ART+15]. Provided that all vector components are non-negative, the key idea is to normalize each vector component with a power normalization exponent α with $0 < \alpha < 1$. Power normalization is disabled for $\alpha = 1$. Due to the characteristics of the root functions which are represented by the power of α, large absolute frequencies in the spatial pyramid are discounted more than small absolute frequencies. This has a positive effect since high individual frequencies tend to dominate the similarity measure. For example, in BoW representations this is typically addressed with stop word filtering or inverse-document-frequency term weighting, cf. [BR11, Sec. 3.2]. For word spotting with spatial pyramid representations, power normalization can lead to performance improvements

Figure 48: Qualitative result on the GW20 benchmark. The retrieval list is showing the top-ten results. The list is organized in two rows such that the top-five results are in the first row and similarity decreases from left to right. Detections that are considered as relevant are shown with a green frame, an irrelevant detection is shown with a red frame.

without increasing the computational complexity [ART+15]. Table 31 confirms this observation for the normalization exponent that was used in [ART+15]. Power normalization with $\alpha = 0.35$ improves word spotting performance on BT50 *significantly* ($\hat{p} \leqslant 0.05$).

In the following, the baseline method will be evaluated for all query-by-example benchmarks considered. The comparison of the baseline method to segmentation-free word spotting with BoF-HMMs allows to assess the benefit of patch-based decoding with the Viterbi algorithm. In this regard, the most important distinguishing characteristic is the possibility to infer the most likely occurrence of the query word within a patch.

5.4.3 *George Washington letters*

The George Washington letters are widely used for evaluating word spotting performance, cf. [GSG+17]. The GW20 benchmark, see Section 5.2.1, defines a protocol for evaluating segmentation-free word spotting on document level on the 20 pages. The benchmark has been used for meta-parameter optimization in Section 5.3.

Figure 48 shows a qualitative result for the query word image which has been used in order to illustrate word spotting methods throughout the previous chapters. The result gives an impression of the relatively homogeneous visual appearance of text in the GW20 dataset. This makes the GW20 document images highly suitable for query-by-example word spotting without annotated training material. The only irrelevant detection in the top-ten results is an occurrence of the word *plan* which is quite similar to the query word image.

Table 32 shows the quantitative comparison to related methods from the literature which follow the same evaluation protocol for segmentation-free query-by-example word spotting. Besides the word spotting performance, average retrieval times for processing a single query on a single document image are provided. The results show that the optimized BoF-HMM configuration, i.e., the two-stage integration with a 3×3 re-ranking mask, outperforms all related methods

Table 32: Query-by-example results on George Washington

Method	query/page time [ms]	GW20 mAP [%]
Baseline	> 60,000	*70.2*
BoF-HMM (IFS)	1,406	*64.1*
BoF-HMM (IFS-VT 1×1)	1,859	74.1
BoF-HMM (IFS-VT 3×3)	2,321	**75.1**
BoF-HMM (VT)	19,268	75.4
Spatial pyramid indexing [RAT+15a]	3	61.4
Exemplar SVM [AGF+14b]	86	59.1
Scale-space pyramid [RKE16]	115	56.0
Random projections [KWD14]	80	50.1

in terms of word spotting performance by a large margin. Due to the small word size variability in the GW20 dataset, all related methods but one are built on patch-based frameworks with a fixed patch geometry. The *exemplar SVM* [AGF+14b] and the *scale-space pyramid* [RKE16] use an individual patch size for each query while the *spatial pyramid indexing* method [RAT+15a] uses one out of four different patch sizes for each query. Only the *random projections* method [KWD14] uses word hypotheses. For this purpose, the document images have to be binarized which is one possible explanation for the low performance of the method. The comparison between the baseline method and the best BoF-HMM configuration shows that patch decoding with the Viterbi algorithm offers advantages over a *static* patch retrieval framework. The difference is *significant* ($\hat{p} \leqslant 0.05$).

The average retrieval times show that the high word spotting performance of the BoF-HMM comes at the cost of computational efficiency. The retrieval times for the BoF-HMM and the baseline method have been recorded under the same hardware and implementation prerequisites as in Section 5.3.9. In order to show results for a scenario with a common memory setup, e.g., 16 gigabyte RAM, the retrieval times refer to a configuration where line representations and inverted indices are *not* cached but computed *on demand*. Retrieval times refer to processing within a single thread in a single process. The parallelization capabilities of the CPU are *not* taken advantage of. The results in Table 32 can be seen a *worst case* estimate for this reason. Based on the results in Table 25 in Section 5.3.9, the *best case* estimate for the optimized BoF-HMM configuration on the specified CPU[9] is around 100 milliseconds (ms) per query and page. The estimate is obtained assuming that line representations and inverted indices can instantly be loaded in memory and 10 document images are

9 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz

Table 33: Query-by-string results on George Washington

Method	Regions	Annotations words in avg.	GW20 mAP [%]
BoF-HMM	Mixture voting	1,215	72.4
BoF-HMM	Mixture voting	3,645	83.3
Region prop. net. [WLB17]	Word hypotheses	3,645	**91.0**
PHOCNET [RSR+17]	Word hypotheses	3,645	84.6
Region prop. net. [WLB17]	Word hypotheses	1,215	73.8

processed in parallel. If only the parallelization capabilities are taken into consideration but all line representations and inverted indices are computed on demand, average retrieval times of around 140 ms (IFS), 190 ms (IFS-VT 1×1) and 230 ms (IFS-VT 3×3) per query and page are suitable for searching larger document collections. An important result is that the two-stage integration makes the application of the BoF-HMM feasible, even if a CPU with less than 12 cores is used for parallel processing. It should be noted that the retrieval times for the related approaches could be improved with parallelization as well.

In [AGF+14b] computational efficiency is achieved by compressing features with *product quantization* (PQ), see Section 3.3.3. The same approach is extended with a multi-scale search-space refinement strategy in [RKE16]. In addition to feature compression, PQ allows for even faster retrieval times by computing *approximate* patch-similarity scores [RAT+15a]. This is achieved by precomputing similarities between the sub-quantizer codewords [JDS11], also see Section 3.4.1, and storing them in a look-up table. The random projections method [KWD14] is computationally efficient due to the number of word hypotheses per page which is small in comparison to the number of possible patches per page. Word hypothesis representations are compressed through randomized projections and randomized max pooling. Finally, it has to be noted that the retrieval times reported for the baseline method are not obtained per query but for all queries that share the same retrieval patch size, see Table 32. Therefore, the processing time of one minute per query and page is a lower estimate.

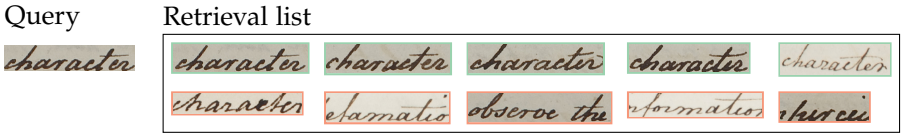
The results for segmentation-free query-by-string word spotting on document level are compared with two state-of-the-art methods in Table 33. These methods have been evaluated with the same evaluation protocols as the BoF-HMM, see Section 5.2.1. The *15-5* cross validation defines four cross-validation folds such that each training set contains 15 document images with 3,645 annotated word images in average as shown in Table 33. The *5-15* cross-validation is used in order to evaluate the performance with limited training data. The training sets contain five document images with 1,215 annotated word images in average.

Both of the two methods considered in the comparison [WLB17; RSR+17] use attribute embeddings, such as *pyramidal histogram of characters* (PHOC), in order to represent the query word and word hypotheses. The attribute vector for the word hypotheses is predicted with a CNN, see Section 3.2.3. The most important difference between both methods is the generation of the word hypotheses, cf. Section 3.1.3.

The region proposal network [WLB17] generates word hypotheses with 15 different aspect ratios and sizes, also cf. [RHG+15]. Each hypothesis is represented with bounding box *coordinates*, a *wordness* score and the *attribute embedding* vector which serve as a basis for end-to-end training. However, the recall achieved with these region proposals can still be improved by adding so-called *dilated text proposals* [WLB17]. Dilated text proposals are obtained by generating text hypotheses based on connected components and classifying them according to *word* and *non-word* with a CNN. The dilation of the text proposals is an adaptation of the method to the annotations in the GW20 benchmark which are arbitrarily padded with document image background [WLB17].

The method presented in [RSR+17] uses the PHOCNET and is inspired by the *region-based CNN* [GDD+16] framework where the region proposal generation is independent of the region representation. The word hypothesis generation is based on the *same* method that has been presented for generating text hypotheses in Section 4.2.1. The major difference is the use of *learned* word detection scores which are used in addition to the SIFT contrast scores. The result in Table 33 is obtained by combining contrast scores with *attribute activation* scores. A dense grid of scores in a document image is given by the attribute activations with maximum value for each attribute vector in the output feature map of a fully convolutional PHOCNET [RSR+17].

Table 33 shows that both methods outperform the BoF-HMM in the segmentation-free query-by-string scenario on GW20. However, the BoF-HMM achieves similar performance as the PHOCNET with the 15-5 cross-validation and similar performance as the region proposal network with the 5-15 cross-validation that limits the amount of annotated training data. A possible explanation for the performance differences of all three methods is the different degree of training visual representations and semantic representations in an end-to-end fashion. The region proposal network achieves the highest integration and is, therefore, also sensitive to the amount of annotated training samples. Training the region proposal network with only 1,215 samples has only been possible by adapting a network which has been pretrained on a large corpus of synthetically generated data [WLB17]. In contrast, the BoF-HMM is mainly intended for the query-by-example scenario where the query word model is estimated from just a single sample.



(a) Bentham 2014 (BT50)

(b) Bentham 2015 (BT70)

Figure 49: Qualitative results on the Bentham manuscripts. The top-ten retrieval lists are shown for an exemplary query of the BT50 benchmark (a) and of the BT70 benchmark (b). The retrieval lists are organized in two rows where the top-five results are shown in the first row and the similarity with respect to the query decreases from left to right. Relevance of the detections according to the benchmark annotations is indicated with green and red frames.

5.4.4 *Jeremy Bentham manuscripts*

The Bentham manuscripts have been used in the keyword spotting competitions at the conferences of the document analysis community in the year 2014 [PZG+14] and in the year 2015 [PTV15b], see Section 5.2.2. The evaluation protocols which have been used in these competitions put an emphasis on detection accuracy. In addition to a comparison according to these original protocols, a comparison to related methods will be given according to the standard 50% IoU relevance criterion, as defined in Section 5.1.

Figure 49 shows qualitative results on the two benchmarks. The results give an impression of the dataset characteristics. These can be considered as a challenge. The BoF-HMM has to generalize across writing styles based on a single example of the query. This capability is illustrated for the relevant detection at retrieval list position five for the example on BT50, see Figure 49a. The instance of the query word has a very different visual appearance than the query word image. The detection at retrieval list position six is considered as irrelevant because the query is only appearing within a longer word. The irrelevant detections for both queries share visual features with the corresponding query word images. Furthermore, the detection of word boundaries is very accurate due to *region snapping*, particularly in Figure 49b. The large word size variability can be noted.

Table 34 to Table 37 show the comparison of the BoF-HMM with the baseline method and related methods from the literature. The results for the related methods in Table 34 and 36 are published in [ZPG17]. The results for the related methods in Table 35 are published

Table 34: Query-by-example results on Bentham 2014

Method	Regions	BT50 mAP [%]
Baseline	Dense patches	*45.6*
BoF-HMM	Mixture voting	**56.9**
Local feature matching [ZPG17]	Descriptor matches	51.7
Random projections [KWD14]	Word hypotheses	42.3
Inkball models [How13]	Dense patches	40.9
Cohesive elastic matching [PDF+14]	Word starting positions	39.7
Cohesive elastic matching [LOL+09]	Word starting positions	22.1

in [VTP15], cf. [PZG+14], and the results for the related methods in Table 37 can be found in [PTV15b]. The BoF-HMM configuration has been optimized for the BT50 benchmark in Section 5.3. The emphasis in the competitions is generally on word spotting performance rather than on computational efficiency. The optimized BoF-HMM configuration has a similar emphasis with the 3×3 re-ranking mask which is used in the two-stage integration. From all methods in the comparison on BT50 and BT70, only the method presented in [VTP15] uses annotated training material besides the query word image, cf. Table 35.

Table 34 shows the results for segmentation-free word spotting on the BT50 dataset with a 50% IoU relevance criterion. Besides the BoF-HMM, only the *local feature matching* method [ZPG17] achieves a mAP higher than 50%. This method is designed for handling larger word size variabilities which fits with the characteristics of the Bentham manuscripts. The method works in two stages. Potential occurrences of the query word are detected in a first stage based on descriptor matches between the query word image and the document images. In the second stage, the matching process is refined with respect to differently sized contexts around the potential query word detections from the first stage. Within such a context the spatial locations of the descriptors are size normalized with respect to the corresponding detection points. Given the size-normalized descriptor locations in the query word image, a similarity score for each detection point and context is obtained by considering only those descriptor matches that are spatially consistent in the size-normalized coordinate systems, cf. Section 3.3.1. Due to the spacious writing style in the Bentham manuscripts, it is unlikely that the search contexts interfere with adjacent text lines. This characteristic is also beneficial for computing word hypotheses in the *random projections* method [KWD14]. The *inkball model* [How13] is inspired by deformable part-based models for object detection [FGM+10]. By computing the costs for generating pen-strokes with the model, document image regions that are visually similar to the query can be detected in a patch-based framework.

Table 35: Query-by-example results on Bentham 2014 (original)

Method	Annotations	BT50 mAP [%] $\iota_A = 0.7$
Baseline	1 word	*41.7*
BoF-HMM	1 word	**54.9**
HMM word graphs and LM [VTP15]	8,019 lines	71.5
Random projections [KWD14]	1 word	41.9
Inkball models [How13]	1 word	37.2
Cohesive elastic matching [PDF+14]	1 word	34.7
Cohesive elastic matching [LOL+09]	1 word	20.9

Inkball models are also created for the most similar patches which are then verified against the query word image. The *cohesive elastic matching* [LOL+09] method uses an approach to local feature matching which is similar to the approach in [ZPG17]. Given the relative positions of local features in the query word image with respect to the query word starting point, only those features are considered for the matching process that are spatially consistent with word starting positions that have been detected in the document image, cf. Section 3.3.1. The large performance difference for two applications of *cohesive elastic matching* in [LOL+09] and [PDF+14], see Table 34, is an example for the potentially strong influence of manual meta-parameter tuning. In this regard, the robustness of BoF-HMMs has been demonstrated in Section 5.3. Only the adaptation of a single meta parameter, i.e., the SIFT descriptor size, leads to *significantly* different word spotting results on two datasets with very different characteristics, see Table 6 in Section 5.3.3. Besides the *random projections* method, all of the approaches in Table 34 can be considered as computationally expensive.

Table 35 presents the results that have been obtained with the relevance measure that has originally been used in the 2014 competition. In contrast to intersection over union, the intersection between two regions is *not* normalized with the union of the two regions but only with the area of the *relevant* region from the benchmark annotations. This is indicated as ι_A in Table 35. It should be noted that this gives an advantage to methods that tend to detect larger regions in contrast to methods that detect regions which fit the detected word instances tightly. In accordance with [PZG+14], the results have been obtained with a 70% relevance threshold.

The results in Table 35 are consistent with the results in Table 34. Furthermore, a comparison with an HMM-based query-by-example method [VTP15] is presented. Given a lexicon, the method computes a word graph for all text line images. The line images are automatically segmented from the document images. Based on an n-best recog-

Table 36: Query-by-example results on Bentham 2015

Method	Regions	BT70 mAP [%]
Baseline	Dense patches	*30.4*
BoF-HMM	Mixture voting	**39.4**
Local feature matching [ZPG17]	Descriptor matches	32.6
Spatial pyramid matching [SF15]	Word segmentation	29.3
Spatial pyramid matching [ART+13]	Dense patches	11.6

nition of the query word image, the joint probability of possible transcriptions of the query word image and the occurrence of the recognized words in the text line can be obtained, cf. Section 3.3.2. The high word spotting performance comes at the cost of a large training corpus which is annotated at line level. A large text corpus for estimating a *language model* (LM) is required in addition. The dataset for HMM training originates from the Bentham manuscripts. The dataset for training the LM contains texts from the Bentham manuscripts along with other text corpora [VTP15]. It is important to note that these requirements are hardly met in a word spotting scenario where an unknown document collection should be explored. Thus, the word graph method is intended for a different scenario than the scenario targeted by the other approaches in Table 34 and 35. Among these methods, the BoF-HMM clearly achieves the highest performance.

Table 36 shows the results for segmentation-free query-by-example word spotting on the BT70 dataset with a 50% IoU relevance criterion. Besides the *local feature matching* [ZPG17] method, all other related methods, including the baseline method, are based on spatial pyramids. While all of the spatial pyramid adaptations are very high-dimensional, the major difference between the methods lies in the computation of region hypotheses in the document images. The most effective approach is used for *spatial pyramid matching* [SF15] where words are automatically segmented from document images based on a connected components analysis. The spacious writing in the Bentham manuscripts makes the heuristic segmentation feasible while the number of segments is limited. This is advantageous for handling the $49,152$-dimensional representations. The *spatial pyramid matching* [ART+13], cf. [PTV15b], approach goes even further by concatenating spatial pyramid representations for four different descriptor sizes. In the patch-based framework, only a limited number of different patch sizes can be handled due to the $172,032$-dimensional representations. In contrast, the baseline method uses an individual patch size for each query. Each patch is represented with a $16,384$-dimensional vector. Table 36 shows that the BoF-HMM outperforms all of these methods by a large margin.

Table 37: Query-by-example results on Bentham 2015 (original)

Method	Regions	BT70 mAP_{ip} [%] $\iota = 0.7$
Baseline	Dense patches	*23.5*
BoF-HMM	Mixture voting	**37.4**
Spatial pyramid matching [SF15]	Word segmentation	27.6
Patch-based matching [GP09]	Dense patches	10.2
Spatial pyramid matching [ART+13]	Dense patches	8.5

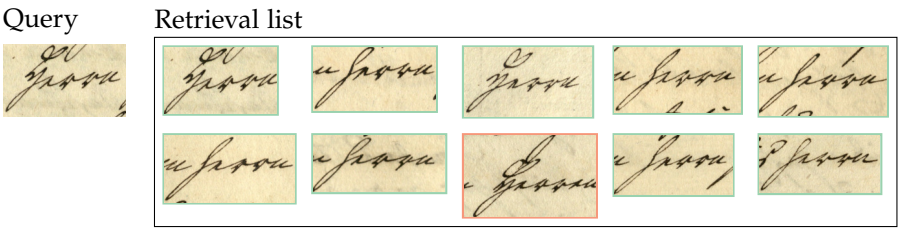
The results in Table 37 are obtained with a 70% IoU relevance threshold in accordance with evaluation protocol in the 2015 competition. In comparison to the results in Table 36 (50% IoU), the performance drop of the BoF-HMM is small in comparison to the performance drop of the baseline method. This emphasizes the importance of refining patches with Viterbi decoding and region snapping. The difference of mAP and mAP_{ip} is 0.4% for the BoF-HMM.

The *patch-based matching* [GP09] method uses an individual patch size per query and performs template matching based on image intensity values. The results for the *spatial pyramid matching* methods [SF15; ART+13] are consistent with Table 36.

The results on the BT50 and BT70 datasets demonstrate the word spotting capabilities of the BoF-HMM in a very challenging word spotting scenario. The BoF-HMM outperforms all methods besides the HMM word graphs. However, instead of a single annotated sample for model estimation, large training corpora for HMM training and LM training are required. The word graphs are based on a lexicon which is representative for the words appearing in the Bentham manuscripts. While the BoF-HMM meta-parameters have been subject to optimization on BT50, no further optimization has been performed for BT70. In comparison to the baseline method, the benefits of patch refinement with Viterbi decoding and region snapping are particularly distinctive. All differences are *significant* ($\hat{p} \leqslant 0.05$). With a 70% relevance threshold, the relative improvements over the baseline method of 31.7% (13.2% absolute) on BT50 and 59.1% (13.9% absolute) on BT70 can be seen as major advancements. BoF-HMMs are orders of magnitudes more efficient than the baseline method.

5.4.5 *Konzilsprotokolle and Botany*

The Konzilsprotokolle dataset and the Botany dataset have been used in the 2016 keyword spotting competition [PZG+16]. The focus of the competition was to analyze how word spotting methods make use of different amounts of annotated training material. For this purpose,



(a) Konzilsprotokolle (KP20)

(b) Botany (BO20)

Figure 50: Qualitative query-by-example results on the Konzilsprotokolle and Botany datasets. The top-ten retrieval results are shown for a query on the KP20 benchmark (a) and for a query on the BO20 benchmark (b). The retrieval lists are organized in two rows where the top-five detections are shown in the first row and similarity with respect to query decreases from left to right. Corresponding to the benchmark annotations, the relevance of the detections is indicated with green and red frames.

differently sized training sets are available for the query-by-example and query-by-string benchmarks on both datasets, see Section 5.2.3. Word spotting with BoF-HMMs is based on the optimized configuration that has been obtained for BT50 in Section 5.3, see Table 4. Only the SIFT descriptor size is specifically adapted according to the estimated descriptor size in Section 5.4.1, see Table 30.

Figure 50 shows qualitative results for query-by-example word spotting on both datasets. The retrieval list for KP20 is highly accurate. Only a single instance of the query word is considered as irrelevant. This is due to an insufficient IoU for a detection that is principally correct. A descender from the line above does not allow for finding an accurately positioned bounding box with region snapping. In addition, it can be noted that a similar descender is also present in the query word image. The qualitative result for query-by-example word spotting on BO20 shows that the BoF-HMM does not generalize across different writing styles sufficiently but is more sensitive to the writing style in the query word image. An impression of the writing style variability for the query word is given in Figure 39b in Section 5.2.3. Furthermore, it can be seen that three detections among the top-five results show the query but are considered as irrelevant. The detection at retrieval list position two shows the query as part of another word. The regions at list positions three and four are considered as irrelevant because a decoration at the top left of the first character in each detection is not fully detected.

Table 38: Query-by-example results on Konzilsprotokolle

Method	Regions	Annotations words	KP20 mAP_{ip} [%]
Baseline	Dense patches	1	72.9
BoF-HMM	Mixture voting	1	**79.3**
PHOCNET [RSR+17]	Word hypotheses	16,920	91.1
Rand. projections [KWD14]	Word hypotheses	1	61.8
PHOCNET [SF16]	Dense patches	1,850	52.2
Attribute SVMs [GV15b]	Dense patches	16,920	0.0

Table 39: Query-by-example results on Botany

Method	Regions	Annotations words	BO20 mAP_{ip} [%]
Baseline	Dense patches	1	44.6
BoF-HMM	Mixture voting	1	**47.3**
PHOCNET [RSR+17]	Word hypotheses	21,982	74.5
Rand. projections [KWD14]	Word hypotheses	1	37.5
PHOCNET [SF16]	Dense patches	1,685	15.9
Attribute SVMs [GV15b]	Dense patches	21,982	0.4

The quantitative evaluation for segmentation-free query-by-example and query-by-string word spotting uses the mAP_{ip} measure with a 50% IoU relevance criterion on both datasets. The results of the related methods have been published in [PZG+16] and [RSR+17]. Table 38 and 39 show the results in the query-by-example scenario. Along with the performance, the number of annotated samples and the method for obtaining regions are provided. Since the query word image is considered as an annotated sample, the number of word annotations is *one* for methods that do not use the training sets. In consistence with the results on GW20 and BT50, the *random projections* [KWD14] method is clearly outperformed. As for the query-by-example benchmarks on all other datasets, the BoF-HMM outperforms the baseline method. In this regard, the difference on BO20 is *not* significant ($\hat{p} > 0.05$). This can be explained with the large visual variability and the small number of queries on BO20. The low performance of the *attribute SVMs* [GV15b] is most likely due to the application of the method rather than due to a principle problem.

An interesting observation can be made with respect to the performance of two different applications of the PHOCNET. The PHOCNET application [SF16] uses a patch-based framework with six different patch sizes and the smallest of the training sets. Despite the use of

Table 40: Query-by-string results on Konzilsprotokolle

Method	Regions	Annotations words	KP20 mAP_{ip} [%]
BoF-HMM	Mixture voting	1,849	73.4
BoF-HMM	Mixture voting	16,919	74.5
PHOCNET [RSR+17]	Word hypotheses	16,919	**89.9**
PHOCNET [SF16]	Dense patches	1,849	48.4

Table 41: Query-by-string results on Botany

Method	Regions	Annotations words	BO20 mAP_{ip} [%]
BoF-HMM	Mixture voting	1,684	7.1
BoF-HMM	Mixture voting	21,981	22.7
PHOCNET [RSR+17]	Word hypotheses	21,981	**78.8**
PHOCNET [SF16]	Dense patches	1,684	11.8

an annotated training set, the PHOCNET application [SF16] is outperformed by the BoF-HMM, the baseline method and the *random projections* [KWD14] method on both benchmarks. However, in contrast to the PHOCNET, these methods just use a single annotated example for word spotting. A possible explanation is that the patch-based framework with only six patch sizes will typically not generate very accurate detections. Another explanation is the small training set that might not be sufficient for training the deep neural network. In comparison, the PHOCNET with word hypotheses and the largest training set [RSR+17] achieves the highest performance that has been reported for segmentation-free word spotting on these benchmarks.

Finally, the results for segmentation-free query-by-string word spotting on the KP20 and BO20 datasets are reported in Table 40 and 41. The BoF-HMM configurations are directly based upon the corresponding query-by-example configurations. The query-by-string meta-parameters have been optimized on the GW20 dataset, see Section 5.3.6.

Table 40 and 41 show that the query-by-string performance of the BoF-HMM is very robust if the visual variability in the document images is limited. However, in comparison to the PHOCNET with word hypotheses [RSR+17] the BoF-HMM is clearly outperformed. The large increase of the training set on KP20 is not reflected in the word spotting performance. Similarly, the results on BO20 show that the high performance of the BoF-HMM for query-by-example word spotting with just a single example comes at the cost of limited learning capabilities if the visual variability in the document images is large.

6

CONCLUSION

Segmentation-free word spotting with BoF-HMMs makes document images searchable with minimum manual effort. The user is supported directly after acquiring a collection of document images. The methods that have been presented in the previous chapters allow for preparing the collection for word spotting fully automatically. Most importantly, no annotated sample data has to be provided. This works in the query-by-example scenario where the user selects a single example of the query word. Document image regions are retrieved according to visual similarity to the query word image and are presented to the user in a ranked retrieval list. The information that is available about the problem domain is integrated at various levels within this process. Highly accurate results can be achieved despite the lack of an annotated training dataset. The computation of the retrieval results is fast by searching with the BoF-HMM in two decoding stages.

- The *only* meta parameter that is sensitive to the visual characteristics of the document collection is the SIFT descriptor size. An *automatic* size estimate is based on an estimate of the typical text core height.
- BoF document image representations are automatically adapted to the visual characteristics of the document collection. The BoF meta parameters are robust since the representations are learned from sample data in an unsupervised manner.
- Text, line and whitespace hypotheses are extracted in a bottom-up manner. They *indicate* the presence of the corresponding document contents. Text hypotheses can be considered as alternatives to each other and do *not* represent a segmentation of the document image. This also applies to line hypotheses.
- Patch-based decoding is performed in line hypotheses that are relevant to the query due to similar height. The patch width is relative to the query width. Due to the sequential modeling in the HMM, the text orientation must be approximately horizontal.
- Patch similarity scores are computed with the query HMM. The query HMM is a compound of context HMMs and the query *word* HMM. While the context models are estimated in an unsupervised manner, the query word model is estimated from the query word image. Computational efficiency is achieved by decoding the compound query HMM only in document image regions that share visual features with the query word model.

Consequently, the proposed method exploits the characteristic of document images on a very general level. The most important assumption is that the text has a sequential structure and an approximately horizontal orientation. In the same way as text hypotheses *indicate* the presence of text, line hypotheses *indicate* document regions that contain text which is similarly high as the query. Essentially, line hypotheses prune the dense grid of patches. Based on text hypotheses, the line hypotheses are obtained for potential text heights. In addition to the horizontal orientation of text, further assumptions about the text layout are only required for the descriptor size estimation. The text core height estimation assumes that the document images contain *mostly text* and that the text is mostly written in *horizontal lines*. In case these assumptions are violated, the descriptor size has to be optimized experimentally or with expert knowledge. The typical text core height is a good estimate for the edge length of the square descriptor area if the visual variability in the document images is high. If the visual variability is low, better performance can be achieved with larger descriptors.

If only a single example is available for query model estimation, an inherent characteristic is that only those instances of the query word can be detected that are written in a similar style as the query word image. The ability to cope with these visual variabilities depends on the generalization capabilities of a method. This is an important factor for the word spotting performance. The qualitative results in Figure 48, 49 and 50 give an impression of the generalization capabilities of the BoF-HMM. Assuming that no annotated training dataset is available, BoF-HMMs outperform all other query-by-example methods that have been evaluated on any of the five benchmarks considered. Table 42 shows a summary of the query-by-example results that have been presented in Section 5.4. The results of the related methods can be found in [RKE16] for GW20, in [ZPG17] for BT50 and BT70 as well as [RSR+17; PZG+16] for KP20 and BO20. In comparison to the *best* results that have been reported in the literature, the BoF-HMM achieves relative mAP improvements of 22.3% (13.7% absolute) [RAT+15a] on GW20, 10.1% (5.2% absolute) [ZPG17] on BT50, 20.9% (6.8% absolute) [ZPG17] on BT70, 28.3% (17.5% absolute) [KWD14] on KP20 and 26.1% (9.8% absolute) [KWD14] on BO20.

A comprehensive list of segmentation-free word spotting methods from the literature can be found in Table 1 in Section 3.4. The BoF-HMM has been compared to seven out of 13 methods in Table 1 that do not use an annotated training dataset. The improvements can be considered as major advancements with respect to the current state-of-the-art in segmentation-free query-by-example word spotting with just a single annotated example, also cf. Table 49 in Appendix C.

With respect to the query-by-example results on KP20 and BO20, the BoF-HMM gets only outperformed by the PHOCNET that uses word

Table 42: Query-by-example results summary

Method	GW20 mAP [%]	BT50 mAP [%]	BT70 mAP [%]	KP20 mAP_{ip} [%]	BO20 mAP_{ip} [%]
Baseline	*70.2*	*45.6*	*30.4*	*72.9*	44.6
BoF-HMM	**75.1**	**56.9**	**39.4**	**79.3**	**47.3**
Spatial pyramid indexing [RAT+15a]	61.4	–	–	–	–
Exemplar SVM [AGF+14b]	59.1	–	–	–	–
Scale-space pyramid [RKE16]	56.0	–	–	–	–
Local feature matching [ZPG17]	–	51.7	32.6	–	–
Random projections [KWD14]	50.1	42.3	–	61.8	37.5
Inkball models [How13]	–	40.9	–	–	–
Cohesive elastic matching [PDF+14]	–	39.7	–	–	–
Cohesive elastic matching [LOL+09]	–	22.1	–	–	–
Spatial pyramid matching [SF15]	–	–	29.3	–	–
Spatial pyramid matching [ART+13]	–	–	11.6	–	–
PHOCNET [RSR+17] (*)	–	–	–	91.1	74.5
PHOCNET [SF16] (*)	–	–	–	52.2	15.9
Attribute SVMs [GV15b] (*)	–	–	–	0.0	0.4

(*) Requires an annotated training dataset.

hypotheses [RSR+17] and a training dataset with 16,919 word-level annotations on KP20 and 21,981 word-level annotations on BO20, in addition to the query word image. The relative mAP improvements over the BoF-HMM of 14.9% (11.8% absolute) on KP20 and 57.5% (27.2% absolute) on BO20 can be explained with the use of just a single annotated example for estimating the BoF-HMM. Given the largely reduced manual effort if no annotated training dataset is required, the performance improvement on KP20 can be considered as limited. However, the large performance improvement on BO20 shows that a training dataset is required for coping with the large visual variabilities in the BO20 benchmark dataset. The methods that require annotated training data are listed at the bottom of Table 42.

The PHOCNET with word hypotheses [RSR+17] outperforms the BoF-HMM for query-by-string word spotting on KP20 similarly as in

the query-by-example scenario. The performance difference on BO20 is even larger than in the query-by-example scenario due to the increased challenge of the BO20 query-by-string benchmark. However, if training data is limited, query-by-string word spotting with BoF-HMMs has shown a performance that is competitive with the region proposal network [WLB17] on the GW20 *5-15* cross-validation, see Section 5.2.1. This shows that word spotting with BoF-HMMs in the query-by-string scenario is suited for extending the query-by-example scenario where no annotated training dataset is available. While performing query-by-example word spotting, the user can give relevance feedback with respect to the document regions in the retrieval list. While the collection of annotated samples grows, the character models can iteratively be re-estimated. In this way, the system can even ask for examples that are relevant for character classes which are not represented well in the training dataset.

The BoF-HMM can be compared to two different applications of the PHOCNET in the segmentation-free scenario on KP20 and BO20. This is interesting because of the typically high PHOCNET performance [SF18]. The BoF-HMM clearly outperforms the PHOCNET that uses a patch-based framework with six different patch sizes [SF16; PZG+16] on the query-by-example *and* query-by-string benchmarks. Besides the small training dataset that can be a reason for reduced performance, the reduced detection accuracy in the patch-based framework is likely to affect performance severely. This can also be observed if the performance of the *baseline method*, cf. Section 5.4.2, is compared to the performance of spatial pyramid matching [RAT+15a], cf. results for GW20 and BT70 in Table 42. By using an individual patch size per query, the baseline method clearly achieves the better performance even though the models are similar. The comparison between the baseline method and the BoF-HMM confirms the importance of decoding the most likely occurrence of the query *within* a patch. This *most distinguishing property* between the two approaches leads to *consistently better performance* on all query-by-example benchmarks, see Table 42. The performance differences are significant ($\hat{p} \leqslant 0.05$) except for the BO20 benchmark. This can be explained with the large visual variability on BO20 which generally limits the applicability of methods that do not use an annotated training dataset.

The following important architectural design choices for segmentation-free word spotting with BoF-HMMs have been confirmed in the parameter optimization on the GW20 and BT50 benchmarks:

- The use of *line hypotheses* reduces the search space without affecting retrieval performance.
- The *visual-word mixture model* outperforms three other mixture models that are relevant for representing BoF vectors.

- Decoding the *most likely occurrence of the query within a patch* is better than using a static patch-decoding framework.
- Ranking document regions according to the *output probability* is better than approximating the query posterior probability with the background model.
- Mixture component voting makes the application of the computationally expensive Viterbi decoding feasible in practice. Word spotting performance is affected only marginally.
- Region snapping allows for highly accurate detections by refining the detected region bounds with the text hypotheses bounds.

The choice of the BoF output model was an important question in this thesis. The visual-word mixture model is motivated by interpreting the relative visual word frequencies in a BoF vector as discrete probability distribution. In the generative approach, this corresponds to the generation of a single visual word. Although this model assumption is clearly violated, the suitability of the method was demonstrated in an extensive evaluation and comparison with three other output models. All of these models are better suited with respect to their model assumptions. However, the visual-word mixture model outperforms these models significantly ($\hat{p} \leqslant 0.05$) on two query-by-example word spotting benchmarks where no annotated training data is available. The behavior can be explained with the better generalization capabilities of the visual words model in this scenario. High similarity can already be measured if *individual* visual words from the query model are observed in the BoF vectors in the document images. In contrast, the other approaches model configurations of multiple visual words.

With respect to retrieval efficiency, the direct application of Viterbi decoding in a patch-based framework can be considered as infeasible in practice. This is due to the large number of patches and the computational complexity of the Viterbi algorithm. The application of mixture component voting in the two-stage decoding process is essential for this reason. Assuming a common memory and disk configuration, average processing times of 230 milliseconds per query and page can be achieved if 10 pages are processed in parallel on the Intel(R) Xeon(R) multicore CPU used in the evaluation, cf. Section 5.4.3. In practice, the hardware requirements grow with the required processing speed. However, in case that larger collections with several hundred handwritten documents have to be searched, it can be doubted if the visual variability in the document images is generally limited as required in the targeted query-by-example scenario. In order to improve the user experience, the processing order of the document images can be sorted according to writing style similarity with respect to the query. This way, the user can be presented with

the most promising results first. Furthermore, the applicability of the method to the document collection gets more transparent to the user. Writing style similarity could, for example, be measured based on holistic BoF representations of the document images and a holistic BoF representation of the query word.

In this thesis, a method for segmentation-free word spotting has been proposed that largely outperforms the state-of-the-art in the query-by-example scenario where no annotated training data is available. An extension to query-by-string word spotting is possible. The evaluations have shown that it is important to address region detection and region retrieval jointly. Computing word hypotheses partly requires to recognize different words. Considering the complexity of this task with hundred thousands of different word classes, it can be expected that this will hardly be a solution that works in general. Patch-based frameworks do not have such requirements since retrieval is *directly* based on similarity to the query. However, static patch geometries are not sufficient for handling word size variabilities. The proposed method addresses these challenges by a *new* combination of the different region detection approaches with Viterbi decoding. The computational effort is limited with a *new* voting-based decoding algorithm for semi-continuous HMMs. These methodological contributions go far beyond the state-of-the-art which is demonstrated by the outstanding word spotting performance in the targeted scenario.

The future relevance of the contributions to segmentation-free word spotting are motivated from a *methodological* point-of-view and a *practical* point-of-view.

With respect to the methodological point-of-view, the contributions highlight the importance of sequence models for handling segmentation and recognition jointly. Possible extensions of the method could address the limitations for learning from a larger number of annotated samples. The main research questions are how to train the BoF output model and the HMM in an end-to-end manner and how to decode the most likely occurrence of the query in a patch not only in horizontal but also in vertical direction. Decoding could be based on the output of a fully convolutional neural network in order to take advantage of the high CNN word spotting performance.

With respect to the practical point-of-view, the proposed method constitutes a complete word spotting system. Since no manual preparation of the document images is required, the system is highly interesting for exploring newly acquired document collections. The assumptions about the visual appearance of text in the document images are kept to a minimum. The system does not make any final decisions but leaves the interpretation of the results to the user. Scholars in the humanities can be supported with a state-of-the-art system for searching document images automatically.

A

ADDITIONAL EXPERIMENTS

In addition to the evaluation that has been presented in Chapter 5, experiments that analyze the behavior of the *bag-of-features* (BoF) output models will be presented in the following. These experiments are intended as supplemental results. They confirm the argumentation in Section 5.3.4. This is achieved by presenting an evaluation of the *multinomial mixture model* (Section A.1) that has been used as a reference for a statistical BoF model. Furthermore, the effect of different *descriptor sizes* on different BoF output models is analyzed (Section A.2). Unless otherwise noted, performance is measured with mAP and a 50% *intersection over union* (IoU) relevance criterion, cf. Section 5.1.1. Within each of the following tables, the *significance* of the performance differences is analyzed with permutation tests ($\hat{p} \leqslant 0.05$), cf. Section 5.1.2. The reference result is indicated with a bold font. Results that differ from the reference result significantly are indicated with an italic font.

A.1 MULTINOMIAL MIXTURE MODEL

The multinomial mixture model has been presented in Section 2.4. The experiments that will be presented in this section show how different parameterizations of the multinomial mixture model affect the word spotting performance. This justifies the performance that is specified for the multinomial model in Table 13 in Section 5.3.4. The experiments are performed in analogy to the experiments that have been shown for the EDCM mixture model in the same section.

Table 43 presents an evaluation of the parameters that are used for the mixture model estimation with the EM algorithm. An evaluation of the number of mixture components and the number of visual words is presented in Table 44. For this purpose, the experiments are performed on the GW20 and the BT50 query-by-example benchmarks, see Section 5.2. The training sets are given by all BoF vectors that are extracted from all line hypotheses. The line hypotheses are obtained from the document images of the corresponding benchmark.

The key idea for representing the probability of a BoF vector with the multinomial distribution is to model the joint occurrence of all visual words in the BoF vector as a product of visual word probabilities $p(\mathcal{V} = v)$ where $\mathcal{V}$ is a random variable over the event space $\Omega_v = \{0, \ldots, V-1\}$, cf. Equation 2 in Section 2.4. If multinomial distributions are used as mixture components in a mixture model, the visual word probabilities are specific to the mixture component $k \in \Omega_M$ with $\Omega_M = \{0, \ldots, M-1\}$. Thus, the probability for visual word v in

Table 43: Multinomial estimation parameters

Smoothing α	Det. annealing $(\tau_0, \ldots, \tau_n)$	GW20 mAP [%]	BT50 mAP [%]
0.5	$(2, 1)$	67.3	50.0
10^{-5}	$(2, 1)$	**68.1**	**49.5**
10^{-10}	$(2, 1)$	68.2	49.4
10^{-5}	$(5, 2, 1)$	67.3	49.5
10^{-5}	1	68.6	48.7

Table 44: Multinomial mixture model

Mixture components M	Vocabulary size V	GW20 mAP [%]	BT50 mAP [%]
256	4096	67.2	50.1
512	4096	**68.1**	**49.5**
1024	4096	67.9	48.0
512	2048	67.2	49.1
512	8192	67.2	49.6

component k can be denoted as $p(\mathcal{V} = v \,|\, \mathcal{M} = k)$ where $\mathcal{M}$ is a random variable over the event space $\Omega_{\mathcal{M}}$. These probabilities are the *model parameters* of the multinomial mixture model that are estimated from sample data in an unsupervised manner. Due to the product of probabilities in the multinomial distribution, it must be ensured that *none* of the probability estimates is *zero*. The estimates will be zero for all visual words that cannot be observed in the sample data, because the estimation is based on relative visual word frequencies. In a mixture model with a larger number of mixture components and BoF vectors that contain only few non-zero entries, most of the visual words will not be observed for most of the mixture components, see Section 2.4. Provided that all visual words have non-zero frequency in the training dataset, then $p(\mathcal{V} = v) > 0,\ \forall v \in \Omega_v$. Zero probabilities in the more specific component-dependent distributions $p(\mathcal{V} = v \,|\, \mathcal{M} = k)$ can be avoided by *interpolating* with the more general background distribution as shown in Equation 82.

$$p(\mathcal{V} = v \,|\, \mathcal{M} = k) \leftarrow (1 - \alpha)\, p(\mathcal{V} = v \,|\, \mathcal{M} = k) + \alpha\, p(\mathcal{V} = v) \qquad (82)$$

The influence of the background distribution is controlled by interpolation factor α, cf. [MKM07]. In order to increase the robustness of the model estimation process with respect to the random initialization, *deterministic annealing* can be applied in analogy to the estimation of the EDCM model [UN98; Elk06], cf. Equation 20 in Section 4.4.2. The key idea of deterministic annealing is to smooth the mixture component

Table 45: vMF descriptor size evaluation

SIFT descriptor size	GW20 mAP [%]	BT50 mAP [%]
16	*57.6*	*38.0*
24	*66.2*	48.6
32	*69.6*	**50.6**
40	71.1	47.4
48	**71.6**	*42.3*
56	*70.1*	*35.5*

Table 46: EDCM descriptor size evaluation

SIFT descriptor size	GW20 mAP [%]	BT50 mAP [%]
16	*63.4*	*43.2*
24	*66.7*	48.0
32	*69.0*	**50.8**
40	70.2	49.5
48	**70.1**	*45.0*
56	*68.0*	*39.7*

posterior probability distributions that are obtained in the expectation step of the EM algorithm. Table 43 shows that none of the parameterizations have a significant effect on GW20 or BT50 ($\hat{p} > 0.05$). Further, none of the different numbers of mixture components or visual words have a significant effect on any of the two benchmarks as shown in Table 44. These results are consistent with the results for the EDCM mixture model, see Table 10 and 11 in Section 5.3.4, and can be explained with the large training dataset of BoF vectors.

A.2 DESCRIPTOR SIZE EVALUATION

The size of the SIFT descriptors in the BoF representations is the most important parameter for word spotting with BoF-HMMs. Since the descriptor size optimization strongly depends on the individual characteristics of a document collection, it is unclear how different descriptor sizes affect the performance with *different* BoF output models.

For this purpose, Table 45, 46 and 47 present the word spotting performance for different descriptor sizes for the vMF mixture model, the EDCM mixture model and the multinomial mixture model. In analogy to the descriptor size evaluation of the visual-word mixture model, see Table 6 in Section 5.3.3, the experiments are performed on the query-by-example benchmarks of the GW20 and BT50 datasets.

Table 47: Multinomial model descriptor size evaluation

SIFT descriptor size	GW20 mAP [%]	BT50 mAP [%]
16	*63.0*	*43.3*
24	67.1	47.8
32	67.1	**49.5**
40	68.4	47.0
48	**68.1**	*43.8*
56	*65.6*	*38.9*

Table 48: BoF descriptor size evaluation

SIFT descriptor size	BT70 mAP [%]	KP20 mAP_{ip} [%]	BO20 mAP_{ip} [%]
16	*34.5*	*52.3*	*37.6*
24	39.3	*66.8*	44.6
32	**39.4**	*72.7*	47.7
40	*35.6*	77.5	48.4
48	*31.0*	78.5	**48.6**
56	*26.7*	**79.3**	47.3
64	*22.4*	77.9	44.4

The tables show that the results are mostly consistent across all BoF output models and for both benchmarks. In this regard, results are considered as consistent if the performance differences that are measured for different descriptor sizes are *significantly* different ($\hat{p} \leqslant 0.05$, italic font in the tables) with respect to the reference configuration (bold font in the tables) for different BoF output models. In comparison with Table 6, this justifies the use of the visual-word mixture model in the optimized BoF-HMM configuration. The word spotting performances obtained for the parameter variations of the vMF, EDCM and multinomial mixture models are *not* better than the word spotting performance measured for the visual-word mixture model, see Table 13 in Section 5.3.4.

Finally, Table 48 shows the descriptor size evaluation for the query-by-example benchmarks that have been considered in addition to GW20 and BT50 in Section 5.4. It is important to note that these results have *not* been used for selecting the descriptor size for the comparison with the results from the literature. Instead, the results are provided in order to allow to assess the quality of the *descriptor size estimation* in Section 5.4.1. In comparison to Table 30, Table 48 shows that the *locally optimal* descriptor size diverges from the *estimated* descriptor size only on BO20. The difference is *not* significant ($\hat{p} > 0.05$).

B

DIRICHLET COMPOUND MULTINOMIAL DISTRIBUTION

The *Dirichlet compound multinomial* (DCM) distribution has been studied for classifying [MKE05] and clustering [Elk06] digital texts that are represented as *bag-of-words* (BoW). An important characteristic in this scenario is that the vocabulary size of the BoW representations is large and the texts are short. This leads to very sparse BoW representations. For this reason, the characteristics of these BoW representations are comparable to the characteristics of the *bag-of-features* (BoF) representations as presented in Section 4.4.

For representing the data with a mixture model, the DCM distribution has advantages over the multinomial distribution. If the multinomial mixture components are estimated from sparse BoF vectors, each component models only few visual words, see Section 2.4. In order to avoid visual-word probability estimates that are zero, the visual-word probability distributions of all mixture components have to be smoothed, for example through interpolation with a more general distribution, see Section A.1.

In contrast to the multinomial distribution, the sparsity of the representation is already incorporated in the model assumption of the DCM distribution. This is achieved in a hierarchical approach where a Dirichlet distribution models the generation of multinomial parameters first. Afterwards, the generation of the BoF representation is modeled by the multinomial distribution. Therefore, the Dirichlet distribution can be understood as a topic model. It represents the likelihood of multinomial parameters which represent the probability for visual words in the visual vocabulary. The continuous distribution of multinomial parameter vectors, which is modeled by the Dirichlet distribution, allows for generating parameter vectors with arbitrary small visual word probabilities. This model assumption of the Dirichlet distribution is the main advantage in comparison to estimating visual word probabilities from discrete data directly.

In the following, the derivation of the DCM distribution from the Dirichlet distribution and from the multinomial distribution will be presented (Section B.1). In order to make the DCM distribution better applicable in practice, an adaptation that exploits the sparsity of the BoW vectors has been proposed in [Elk06]. The approximation involved in the so-called *exponential Dirichlet compound multinomial* (EDCM) model is derived (Section B.2). The purpose is to formally describe the model assumptions that are implied if the EDCM mixture model is used as an output model in the given scenario.

B.1 DERIVATION

The derivation of the DCM distribution is based on [MKE05] and in particular on [FKG10, Sec. 1.4]. The following presentation is more elaborate than the individual presentations in the literature.

In order to derive the DCM probability mass function, the Dirichlet probability density function $p(\mathbf{p}|\boldsymbol{\alpha})$ and the multinomial probability mass function $p(\mathbf{x}|\mathbf{p})$ are defined in Equation 83 and 84. The concentration parameters of the Dirichlet distribution are denoted as vector $\boldsymbol{\alpha} \in \mathbb{R}^V_{>0}$ and the visual word probabilities for the multinomial distribution are given by parameter vector $\mathbf{p} = (p(\mathcal{V} = 0), \dots, p(\mathcal{V} = V-1))^\top$ for a random variable $\mathcal{V}$ over the event space $\Omega_v = \{0, \dots, V-1\}$. The BoF vector with absolute term frequencies for a visual vocabulary of size V is denoted by $\mathbf{x} \in \mathbb{N}^V_{\geqslant 0}$.

$$p(\mathbf{p}|\boldsymbol{\alpha}) = \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\prod_{v=0}^{V-1} \Gamma(\alpha_v)} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{\alpha_v - 1} \tag{83}$$

$$p(\mathbf{x}|\mathbf{p}) = \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{x_v} \tag{84}$$

The DCM probability mass function $p(\mathbf{x} \mid \boldsymbol{\alpha})$ is then based on Equation 85. The compound is obtained by integrating over all multinomial parameter vectors $\mathbf{p}$. Substituting the definitions of $p(\mathbf{p}|\boldsymbol{\alpha})$ and $p(\mathbf{x}|\mathbf{p})$ leads to Equation 86. Given the definition of $p(\mathbf{p}|\boldsymbol{\alpha})$ and $p(\mathbf{x}|\mathbf{p})$, it can be noted that the normalization factors are independent of the integration variable $\mathbf{p}$. Furthermore, the products over V visual words can be combined. Equation 87 is obtained by rearranging the factors in Equation 86 accordingly.

$$p(\mathbf{x} \mid \boldsymbol{\alpha}) = \int_{\mathbf{p}} p(\mathbf{x} \mid \mathbf{p})\, p(\mathbf{p} \mid \boldsymbol{\alpha}) \mathrm{d}\mathbf{p} \tag{85}$$

$$= \int_{\mathbf{p}} \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{x_v} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\prod_{v=0}^{V-1} \Gamma(\alpha_v)} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{\alpha_v - 1} \mathrm{d}\mathbf{p} \tag{86}$$

$$= \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\prod_{v=0}^{V-1} \Gamma(\alpha_v)} \int_{\mathbf{p}} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{x_v + \alpha_v - 1} \mathrm{d}\mathbf{p} \tag{87}$$

It can be noted that the product over visual words in the integral in Equation 87 corresponds to the product over visual words in the Dirichlet probability density function in Equation 83 with concentration parameters $\mathbf{x} + \boldsymbol{\alpha}$. In order to obtain an integral over the Dirichlet probability density function $p(\mathbf{p} \mid \mathbf{x} + \boldsymbol{\alpha})$, the missing normalization

factor is incorporated in Equation 88. This is achieved by multiplying with the corresponding normalization factor in the integral and multiplying with the *inverse* of the normalization factor in front of the integral. Since the integral $\int_{\mathbf{p}} p(\mathbf{p} \mid \mathbf{x} + \boldsymbol{\alpha}) d\mathbf{p} = 1$ by the definition of a probability density function, the left-hand side of Equation 88 is equal to the inverse of the normalization factor of the Dirichlet probability density function $p(\mathbf{p} \mid \mathbf{x} + \boldsymbol{\alpha})$, see Equation 89.

$$\int_{\mathbf{p}} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{x_v + \alpha_v - 1} d\mathbf{p} = \frac{\prod_{v=0}^{V-1} \Gamma(\alpha_v + x_v)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)} \left(\int_{\mathbf{p}} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)}{\prod_{v=0}^{V-1} \Gamma(\alpha_v + x_v)} \prod_{v=0}^{V-1} p(\mathcal{V} = v)^{x_v + \alpha_v - 1} d\mathbf{p} \right) \tag{88}$$

$$= \frac{\prod_{v=0}^{V-1} \Gamma(\alpha_v + x_v)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)} \tag{89}$$

Equation 90 is obtained after substituting Equation 89 in Equation 87. By rearranging the denominators from Equation 90 to 91, the definition of the DCM probability mass function is obtained in Equation 92.

$$p(\mathbf{x} \mid \boldsymbol{\alpha}) = \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\prod_{v=0}^{V-1} \Gamma(\alpha_v)} \frac{\prod_{v=0}^{V-1} \Gamma(\alpha_v + x_v)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)} \tag{90}$$

$$= \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)} \frac{\prod_{v=0}^{V-1} \Gamma(\alpha_v + x_v)}{\prod_{v=0}^{V-1} \Gamma(\alpha_v)} \tag{91}$$

$$= \frac{\|\mathbf{x}\|_1!}{\prod_{v=0}^{V-1} x_v!} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)} \prod_{v=0}^{V-1} \frac{\Gamma(x_v + \alpha_v)}{\Gamma(\alpha_v)} \tag{92}$$

B.2 APPROXIMATION

Due the computational cost of estimating and evaluating DCM mixture models, a DCM adaptation for very sparse BoW vectors has been proposed in [Elk06]. The following presentation goes beyond the explanations in [Elk06] and provides additional details with respect to the approximation involved for obtaining the EDCM model.

The first step towards a more efficient definition of the DCM probability mass function is to limit the evaluation to visual words that are non-zero as shown in Equation 93. The products over visual words are limited to the non-zero entries in BoF vector $\mathbf{x}$ for this purpose. The simplification is possible since $0! = 1$ and $\frac{\Gamma(\alpha_v)}{\Gamma(\alpha_v)} = 1$, respectively.

$$p(\mathbf{x} \mid \boldsymbol{\alpha}) = \frac{\|\mathbf{x}\|_1!}{\prod_{v:x_v \geqslant 1} x_v!} \frac{\Gamma(\|\boldsymbol{\alpha}\|_1)}{\Gamma(\|\boldsymbol{\alpha}\|_1 + \|\mathbf{x}\|_1)} \prod_{v:x_v \geqslant 1} \frac{\Gamma(x_v + \alpha_v)}{\Gamma(\alpha_v)} \tag{93}$$

Further simplifications are possible through a linear approximation of the so-called Pochhammer symbol, cf. [OLB+10, Sec. 5.2]. The ap-

proximation has to be considered for $x_v \geqslant 1$ due to the application in Equation 93 and is shown in Equation 94.

$$\frac{\Gamma(x_v + \alpha_v)}{\Gamma(\alpha_v)} \approx \Gamma(x_v)\alpha_v \quad \text{for} \quad x_v \geqslant 1,\ x_v \in \mathbb{N}_{\geqslant 0} \tag{94}$$

In order to show that $\Gamma(x_v)\alpha_v$ is a *linear* approximation as a function of α_v of the Pochhammer symbol for $x_v \geqslant 1$, it will be shown that $\Gamma(x_v)\alpha_v$ is tangential to $\frac{\Gamma(x_v+\alpha_v)}{\Gamma(\alpha_v)}$ in $\alpha_v = 0$. Therefore, the derivative of the Pochhammer symbol with respect to α_v has to be $\Gamma(x_v)$ for $\alpha_v = 0$. For this purpose, it is helpful to note that the Pochhammer symbol can be expressed as a product for $x_v \geqslant 1$ as shown in Equation 95.

$$\frac{\Gamma(x_v + \alpha_v)}{\Gamma(\alpha_v)} = \prod_{k=0}^{x_v-1} (\alpha_v + k) \quad \text{for} \quad x_v \geqslant 1,\ x_v \in \mathbb{N}_{\geqslant 0} \tag{95}$$

The derivative of $\frac{\Gamma(x_v+\alpha_v)}{\Gamma(\alpha_v)}$ in $\alpha_v = 0$ is defined in Equation 96. With the help of Equation 95, it can be seen that the Pochhammer symbol for $\alpha_v = 0$ equals to zero which results in Equation 97. This is due to the product that includes the factor $k = 0$. By extracting the factor $(\alpha_v + k)$ for $k = 0$ in Equation 98, the product over absolute visual word frequencies is defined from $k = 1$ to $x_v - 1$. This allows to cancel out α_v in the fraction of Equation 98 which leads to Equation 99. By considering the limit $\alpha_v \to 0$ in Equation 100 and applying the definition of the factorial $(x_v - 1)!$ in Equation 101, the final result $\Gamma(x_v)$ is obtained with the help of the definition of the Gamma function for integers, cf. [OLB+10, Equ. 5.4.1], in Equation 102.

$$\lim_{\alpha_v \to 0} \frac{\frac{\Gamma(x_v+\alpha_v)}{\Gamma(\alpha_v)} - \frac{\Gamma(x_v+0)}{\Gamma(0)}}{\alpha_v - 0} = \lim_{\alpha_v \to 0} \frac{\prod_{k=0}^{x_v-1}(\alpha_v + k) - \prod_{k=0}^{x_v-1} k}{\alpha_v - 0} \tag{96}$$

$$= \lim_{\alpha_v \to 0} \frac{\prod_{k=0}^{x_v-1}(\alpha_v + k)}{\alpha_v} \tag{97}$$

$$= \lim_{\alpha_v \to 0} \frac{\alpha_v \prod_{k=1}^{x_v-1}(\alpha_v + k)}{\alpha_v} \tag{98}$$

$$= \lim_{\alpha_v \to 0} \prod_{k=1}^{x_v-1} (\alpha_v + k) \tag{99}$$

$$= \prod_{k=1}^{x_v-1} k \tag{100}$$

$$= (x_v - 1)! \tag{101}$$

$$= \Gamma(x_v) \tag{102}$$

Therefore, $\Gamma(x_v)\alpha_v$ is a linear approximation of the Pochhammer symbol in $\alpha_v = 0$ with $x_v \geqslant 1,\ x_v \in \mathbb{N}_{\geqslant 0}$. The approximation is accurate

for small α_v and small x_v. For example, the BoF vectors in the GW20 document images, cf. Section 5.2.1, contain 10 visual words with non-zero frequency in average. Within these non-zero visual words, the average visual word frequency is 1.6. The mean estimated concentration parameter value over 512 mixture components and 4096 visual words is 0.01, cf. Section 5.3.4. According to Equation 94, the error induced by the linear approximation of the Pochhammer symbol is $6.3 \cdot 10^{-5}$ in average on GW20.

In order to obtain the EDCM probability mass function with the linear approximation of the Pochhammer symbol, $\Gamma(x_v)\alpha_v$ is substituted for the Pochhammer symbol in Equation 93 according to Equation 94. The result is shown in Equation 103. It has to be noted that the parameter vector is referred to as $\boldsymbol{\beta}$ in order to distinguish the adaptation from the original DCM probability mass function. In Equation 104, the Gamma function $\Gamma(x_v)$ for integer x_v is expressed as a factorial. This factorial cancels out with the factorial in the denominator of the first normalization factor in Equation 104. The final result for the EDCM probability mass function is shown in Equation 105.

$$p(\mathbf{x} \mid \boldsymbol{\beta}) = \frac{\|\mathbf{x}\|_1!}{\prod_{v:x_v \geqslant 1} x_v!} \frac{\Gamma(\|\boldsymbol{\beta}\|_1)}{\Gamma(\|\boldsymbol{\beta}\|_1 + \|\mathbf{x}\|_1)} \prod_{v:x_v \geqslant 1} \Gamma(x_v)\beta_v \tag{103}$$

$$= \frac{\|\mathbf{x}\|_1!}{\prod_{v:x_v \geqslant 1} x_v!} \frac{\Gamma(\|\boldsymbol{\beta}\|_1)}{\Gamma(\|\boldsymbol{\beta}\|_1 + \|\mathbf{x}\|_1)} \prod_{v:x_v \geqslant 1} (x_v - 1)!\, \beta_v \tag{104}$$

$$= \frac{\|\mathbf{x}\|_1!}{\prod_{v:x_v \geqslant 1} x_v} \frac{\Gamma(\|\boldsymbol{\beta}\|_1)}{\Gamma(\|\boldsymbol{\beta}\|_1 + \|\mathbf{x}\|_1)} \prod_{v:x_v \geqslant 1} \beta_v \tag{105}$$

C

WORD SPOTTING METHODS FROM 1993 TO 2018

Table 49 presents a comprehensive overview of methods for word spotting in *document images*. The selection and organization of the methods is mainly performed according to the methodological novelty at the time of their publication. The methods are briefly characterized with respect to different properties. This allows for identifying word spotting literature which is relevant for different retrieval scenarios. Methods that have already been applied under the given conditions and constrains can be identified. Furthermore, the number of methods that are using certain techniques indicates whether a technique is robust. For example, in the sense that a method has been independently applied by different researchers and works on different datasets. Finally, the development of the field of word spotting can be retraced. From the first approaches that have been presented in 1993 to the current state-of-the-art in 2018, influential trends can be found. In the early methods these are mostly inspired from word spotting in speech signals. Later, the inspiration is rather found in computer vision. However, since the application domain is document images, the influence of various document analysis topics, from handwriting recognition to writer identification, is ever present.

Table 49 contains the columns *Year*, *Method*, *Query*, *Style*, *Annotation*, *Segmentation*, *Features* and *Characteristics*. The *Year* shows when a method has been presented first. This is important as the most relevant publication that is referenced in the *Method* column might have been published at a later time. If this is the case, the original publication can be found within the *Characteristics* column, also along with other related methods. There, the main contributions are briefly described and can be distinguished. In contrast, the title, listed in the *Method* column, focusses on the major contribution and aims at characterizing an approach as a whole. A very important aspect of a word spotting method are the image features that are being considered. Mainly, these fall into one of the categories pen-stroke features and appearance features and are listed in the column *Features*, cf. Section 3.2. As the column focusses on the *image* features, the *semantic* representations, cf. Section 3.2.3, are indicated by the use of a string/-word embedding under *Method* or *Characteristics*. Methods that use semantic representations are built upon spatial pyramid, Fisher vector or CNN image features.

The remaining columns describe the scenario in which the word spotting methods have been applied. The *query* modalities are categorized in *example*, *word*, *string*, *speech* and *online handwriting*, cf. Sec-

tion 1.2 and Section 3.3. The category *word* refers to the scenario where word models are estimated from a given training dataset. Therefore, query words are restricted to a lexicon, cf. Section 3.3.2. Within the column *Style* the text style in the documents is specified. The scenarios *printed*, *single* writer and *multi* writer are distinguished. They indicate the expected visual variability for which the methods have been designed. The higher the visual variability, the more annotated training material is required. These *annotations* can be given on different levels from *synthetic* over *query*, *line* to *word* and *character*. The order reflects the manual effort. No manual annotations are required if the training data is generated synthetically. However, the synthesized data still needs to have similar characteristics as the text in the document images. Query refers to the query-by-example scenario where only a single exemplary occurrence of the query word is provided by the user. Character-level annotations usually have to be obtained with an existing recognizer or the visual variability in the document images must be very limited. For example, only a single template per character is needed in printed document scenarios. Finally, the column *Segmentation* indicates if the word spotting method requires a given segmentation of the document images. For this purpose, segmentation levels are given as *none*, *line*, *word* and *character*. Methods for word and line segmentation are discussed in Section 3.1.1. If no segmentation is required the methods work with patch or word region hypotheses, cf. Section 3.1.2 and Section 3.1.3.

The most important trends and developments that can be identified in Table 49, refer to the word spotting scenario as well as the methods. Until the year 2013, methods have been specifically addressing one query modality. Query-by-example always referred to the scenario where only the single annotated instance of the query word is given. Consequently, it was applied on documents with very limited visual variability. If training material was considered, it was used to estimate word or character models. With the recent popularity of attribute representation and CNNs this distinction became less important. These methods are supporting multiple and also non-standard modalities. For the first time, query-by-example benefits from annotated training material without recognizing the example first. Consequently, the visual variability in the documents considered increases over time.

Regarding the methods, an interesting development can be found in the features. Over time, pen-stroke representations have been replaced with appearance features. Using supervised learning, these are the basis for obtaining semantic representations. The most important distinction in the retrieval methods can be made between holistic and sequence-based approaches. While more holistic approaches can be found in total, sequence-based methods have been continuously considered as well. They are the only possibility for training models with line-level annotations.

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
1993	XOR, distance maps [KH93]	Example	Printed	Query	Word	Binary intensities	Indexing, improving optical character recognition
1993	Character HMMs [CWB93]	String	Printed	Word	Word	Geometric	Sub-character filler model approach, cf. [RP90]
1994	Pseudo 2D-HMMs [KA94]	Word	Printed	Word	Word	Binary intensities	Models sequence of binary 2D pixel values
1994	Word sequence indexing [Hul94]	Example	Printed	Query	Word	Number of CCs	Voting scheme for feature matches
1996	XOR, distance maps [MHR96]	Example	Single	Query	Word	Binary intensities	Applies [KH93] to historic handwritten documents
1997	Keyword signature matching [KGG97]	Word	Multi	Word	None	Geometric	Matching with holistic and graph representations
2000	Line-oriented DTW [KAA+00]	Word	Multi	Word	Line	Geometric	Word candidates, non-maximum-suppression
2003	Keypoint matching [RFR03]	Example	Single	Query	Word	Image patches	Spatial distribution of corresponding keypoints
2003	DTW [RM03]	Example	Single	Query	Word	Geometric	Cmp. of word image matching methods, [RM07] for indexing
2003	Probabilistic retrieval [RLM03]	Words	Single	Word	Word	Geometric (discrete)	Statistical model for feature-word co-occurrences

. . . continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2003	Gradient-based binary features [ZSH03]	Example	Multi	Query	Word	Gradients, geometric	Features from handwriting identification
2003	Synthetic queries and CCs [MMS03]	String	Printed	Synthetic	Char.	Binary intensities	CC-prototype indexing and matching
2004	Generalized HMMs [CZF06]	String	Single	Char.	Line	Discriminative subspace	Variable frame width in statistical process, cf. [ETF+04]
2005	Boosted decision trees [HRM05]	Words	Single	Word	Word	Gray-level intensities	Statistical model [RLM03] based on AdaBoost scores
2005	Eigenspace DTW [TNK05]	Example	Single	Query	Line	Intensity subspace	Patch-based DTW matching
2005	Cohesive elastic matching [LLE07]	Example	Single	Query	None	Gradients	Guides and RoIs for reducing search space [LBE05; PDF+14]
2007	Shape signature indexing [LS07]	Example	Printed	Query	Word	Shape context	Document indexing based on shape context codewords
2007	Character probabilities [CBG09], cf. [CG07]	String	Multi	Char.	Line	Gabor wavelets	SVM-based character hypothesis classification
2007	BoF [AD07]	Example	Single	Query	Word	BoF	Based on SIFT, similarity with Kullback-Leibler divergence
2008	Local gradient histograms [RP08a]	Word	Multi	Word	Word	Local gradient histograms	Matching with HMMs and DTW, SIFT-like features, cf. [TT09]

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2008	Monk system [ZSH08]	Example Word	Single	Query Word	Line	Visual cortex model	Bootstrapping recognition system with HMMs and SVMs
2009	Fisher kernels [PR09]	Word	Multi	Word	Word	Local gradient histograms	Features: generative, retrieval: discriminative
2009	SC-HMMs [RP09b]	Word	Multi	Word	Line	Local gradient histograms	Word hypotheses, HMM score normalization [RP08b]
2009	Multi-level classifiers and clustering [MC09]	Example	Single	Query	None	Geometric	Pen stroke classifiers and CC prototype matching
2009	Patch-based template matching [GP09]	Example	Printed	Query	None	Binary intensities	Block-wise pixel densities, template augmentation
2009	Slit-style HoG [TT09]	Example	Single	Query	Line	HoG	HoG adaption for line-based DTW, i.e., sub-sequence matching
2009	Cohesive elastic matching [LOL+09]	String	Printed	Char.	None	Gradients	Query model generated from character templates, cf. [LLE07]
2009	Model-based sequence similarity [RP12a]	Example	Multi	Query	Word	Local gradient histograms	DTW of SC-HMM state-dependent mixture probabilities [RPL+09]
2009	Synthetic data and SC-HMMs [RP12b]	String	Multi	Synthetic	Word	Local gradient histograms	SC-HMM estimated from real and synthetic data [RP09a]
2010	Character HMMs [FKF+12]	String	Multi	Line	Line	Geometric	Models query in text line, filler model [FKF+10]

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2010	Recurrent neural networks [FFM+12]	String	Multi	Line	Line	Geometric	Decodes character posterior probabilities in text line [FFB10]
2011	Deformable blurred shape model [FFF+11]	Example	Single	Query	Word	Blurred shape model	Symbol recognition model adaptation to word spotting
2011	Pseudo-structural descriptor [FLF11]	Example	Single	Query	Word	Loci	Indexable geometric descriptor from character recognition
2011	Spatial pyramid indexing [RAT+15a]	Example	Single	Query	None	Spatial pyramid	Subspace-embedding [RAT+11], indexing with PQ, cf. [RDE+14]
2012	IFS indexing [SJ12]	Example	Printed	Query	Word	BoF	Visual word indexing, spatial consistency re-ranking
2012	Indexing and SVM refinement [AFF+12]	Word	Single	Word	Word	Loci, HoG	Loci-indexing, patch-based RoI refinement with SVM
2012	Exemplar SVM [AGF+14b]	Example	Single	Query	None	HoG	PQ patch indexing [AGF+12], re-ranking, query expansion
2012	User feedback integration [RL14]	Example	Single	Query	Word	Spatial pyramid	Query fusion and relevance feedback [RL12]
2012	Synthetic grapheme queries [LFG12]	String	Single	Word	Word	Geometric	Grapheme prototype matching
2013	Heat kernel signatures [ZT13]	Example	Single	Query	None	Heat diffusion	Optimal sequence of matched descriptors (query, document)

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2013	Inkball models [How13]	Example	Single	Query	Word	Geometric	Similarity between deformable part-based word models
2013	HMM character lattice [TV13]	String	Multi	Line	Line	Geometric	HMM score decoding with character lattice
2013	Character HMMs and n-grams [FFB+13]	String	Multi	Line	Line	Geometric	Language model re-scoring based on [FKF+12]
2013	Deformable HoG [AFV13]	Example	Single	Query	Word	Deformable HoG	HoG cells are adapted to script in word image
2013	Common subspace [ART+13; RAT+15b]	String Speech	Single	Word	Word	Spatial pyramid	Subspace models cross-domain feature correlations
2013	Contextual [FML+13; FMF+14]	Example	Multi	Query	Word	Loci, Shape context	Query modeled within its context in the document
2013	Attribute SVMs [AGF+14a]	Example String	Multi	Word	Word	Fisher vector	PHOC subspace of visual and textual features [AGF+13; SGL+15]
2013	BoF-HMMs [RRF13]	Example	Single	Query	None	BoF	Patch-based, BoF-HMM estimated from single example
2014	HMM word graphs [TVR+16]	Word	Multi	Line	Line	Geometric	Query posterior with word graphs [PTV14], symbol spotting [CTV18]
2014	Random projections [KWD14]	Example	Single	Query	None	HoG, local binary pattern	Word regions embedded in randomized vector space

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2014	Spatial pyramid hashing [RDE+14]	Example	Single	Query	None	Spatial pyramid	Patches indexed with kernelized probabilistic hashing
2014	Graph embedding [WEG+14a]	Example	Single	Query	Line	Shape context	Coarse-to-fine, DTW based on graph edit distance [WEG+14b]
2014	IFS indexing and BoF-HMMs [RRL+14]	Example	Single	Query	None	BoF	Visual word indexing, region voting, BoF-HMM re-ranking
2015	Graph embedding and indexing [RLF15]	Example	Single	Query	None	Blurred shape model	Graph indexing, graph edit distance on subgraphs
2015	HMM *n*-gram-character lattice [TPV15]	String	Multi	Line	Line	Geometric	Decodes sub-path in character *n*-gram lattice [TV13; FFB+13]
2015	HMM word graphs [VTP15]	Example	Single	Line	Line	Moment-based normalization	Query-by-example *n*-best decoding on word graphs [PTV14]
2015	Attribute SVMs [GV15b; GV15a]	Example String	Multi	Word	None	Fisher vector	Segmentation-free extensions of [AGF+14a], PHOC indexing
2015	Distance learning [SRF15; SF15; RSL+17]	Example	Single	Query	Word	Spatial pyramid	SIFT [SRF15] and word descriptor [SF15; RSL+17] embedding
2015	Deep HMM [TCH+15]	String	Multi	Line	Line	Deep belief network	Hybrid HMM with a deep neural network output model
2015	CNN adaptation [SK15]	Example	Multi	Word	Word	CNN	Pretrained-CNN adaptation for feature extraction

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2015	Feature matching [HF16], cf. [KKG15]	Example	Single	Query	None	Fourier descriptor	Keypoint detection, matching and inlier-outlier detection
2015	BoF-HMMs [RF15]	String	Single	Word	None	BoF	Patch-based, patch geometry estimated from sample data
2016	Scale-space pyramid [RKE16]	Example	Single	Query	None	HoG	Pyramidal refinement, histogram matching, PQ indexing
2016	Two-way Attribute SVMs [WRF16]	O.-Hand-writing	Single	Word	Word	Spatial pyramid	Cross-domain PHOC embedding (multi user)
2016	Flexible sequence matching [MRR+16]	Example	Single	Query	Line	Geometric	DTW extension allowing skips in target sequence
2016	Recurrent neural networks [SGL+16]	String	Single	Line	Line	Frequency filters	Multi-word spotting based on character probability decoding
2016	HMM n-gram-character lattice [TPV16]	String	Multi	Line	Line	Geometric	Character sequence posteriors from character lattice
2016	CNN hypercolumn features [SRG16]	Example	Single	Query	Word	Off-the-shelf CNN	Word-level descriptor with temporal CNN feature aggregation
2016	PHOCNET [SF16]	Example String	Multi	Word	Word	CNN	CNN-based PHOC embedding, synthetic data [GSF18]
2016	Inkball models [BHM16]	Example	Single	Query	None	Geometric	Matching in born-digital cuneiform tablets, cf. [BM18b]

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2016	CNNs and Attribute SVMs [KDJ16]	Example String	Multi	Word	Word	CNN	CNN-features projected to PHOC-space
2016	Siamese CNNs [ZPJ+16; BAE18]	Example	Single	Word	Word	CNN	Distance learning with two-channel CNNs
2016	triplet-CNN word embedding [WB16]	Example String	Multi	Word	Word	CNN	triplet-CNN-based word embedding, cf. Siamese CNN [BAE18]
2017	Recurrent neural network [BHK+17]	String	Multi	Line	Line	CNN	Character sequence posteriors from character lattice [TPV16]
2017	Feature matching [ZPG17]	Example	Single	Query	None	Gradient histograms	Keypoint detection, matching, spatial consistency, cf. [ZPG14]
2017	Region proposal network [WLB17]	Example String	Multi	Word	None	CNN	End-to-end learning for region hypotheses, word embeddings
2017	R-PHOC [GV17]	Example	Single	Word	None	CNN	Region hypotheses based on CC-grouping, PHOC embedding
2017	Levenshtein deep embedding [GRK17]	String	Single	Word	Word	CNN	Learned string embedding for word images
2017	Cross-domain PHOCNET [SRF17]	O.-Hand-writing	Multi	Word	Word	CNN	Cross-domain PHOC embedding with separate and a single CNN
2017	TPP-PHOCNET [SF18; SF17]	Example String	Multi	Word	Word	CNN	Temporal pooling, cf. [RSS+18], embeddings, loss functions

... continued

... continued

Year	Method	Query	Style	Annotation	Segmentation	Features	Characteristics
2017	Word hypotheses [RSR+17]	Example String	Multi	Word	None	CNN	ER hypotheses on CNN word detector scores, PHOC embedding
2018	Probabilistic indexing [LPT+18]	String Words	Multi	Line	Line	CNN	Indexing pseudo words in character lattice [BHK+17]
2018	Zone sequence matching [RLS+18]	Example	Single	Query	Word	Orientation histograms	Sequence matching of temporal zone descriptors (similar to DTW)
2018	Filters for graph-based spotting [SFR18b; SFR18a]	Example	Single	Query	Word	Binary intensities	Fast rejection of graphs that are dissimilar to the query graph
2018	Synthetic codebook [AR18]	Example	Single	Query	Word	Spatial pyramid	Synthetically generated universal BoF codebook
2018	Matching in 3D cuneiform tablets [BM18a]	Example	Single	Query	None	Holistic image descriptors	Patch-based 2D-embedding of 3D triangular meshes
2018	HWNet v2 PHOC embedding [KDJ18]	Example String	Multi	Word	Word	CNN	ResNet with real and synthetic data, cf. [DKM+18]
2018	Compact deep descriptors [RSL+18]	Example	Multi	Word	Word	CNN	n-gram attributes and CNN zoning features, cf. [SRG16]
2018	Direct attribute prediction [RRM+18]	Example String	Multi	Word	Word	CNN	Probabilistic model for the similarity of PHOC vectors

Table 49: Word spotting methods 1993–2018. The table shows different approaches and characterizes them with respect to selected properties.

BIBLIOGRAPHY

[AD07] E. Ataer and P. Duygulu. "Matching Ottoman words: an image retrieval approach to historical document indexing." In: *Proc. of the Int. Conf. on Image and Video Retrieval*. CIVR '07. New York, NY, USA: ACM, 2007, pp. 341–347.

[AFF+12] J. Almazán, D. Fernández, A. Fornés, J. Lladós, and E. Valveny. "A coarse-to-fine approach for handwritten word spotting in large scale historical documents collection." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2012, pp. 455–460.

[AFV13] J. Almazán, A. Fornés, and E. Valveny. "Deformable HOG-based shape descriptor." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 1022–1026.

[AGF+12] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. "Efficient exemplar word spotting." In: *British Machine Vision Conf.* 2012.

[AGF+13] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. "Handwritten word spotting with corrected attributes." In: *Proc. of the Int. Conf. on Computer Vision*. Dec. 2013, pp. 1017–1024.

[AGF+14a] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. "Word spotting and recognition with embedded attributes." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 36.12 (2014), pp. 2552–2566.

[AGF+14b] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. "Segmentation-free word spotting with exemplar SVMs." In: *Pattern Recognition* 47.12 (2014), pp. 3967–3978.

[AHP06] T. Ahonen, A. Hadid, and M. Pietikainen. "Face description with local binary patterns: application to face recognition." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.12 (Dec. 2006), pp. 2037–2041.

[AR18] D. Aldavert and M. Rusiñol. "Synthetically Generated Semantic Codebook for Bag-of-Visual-Words Based Word Spotting." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 223–228.

[ART+13] D. Aldavert, M. Rusiñol, R. Toledo, and J. Lladós. "Integrating visual and textual cues for query-by-string word spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. 2013, pp. 511–515.

[ART+15] D. Aldavert, M. Rusiñol, R. Toledo, and J. Lladós. "A study of bag-of-visual-words representations for handwritten keyword spotting." In: *Int. Journal on Document Analysis and Recognition* 18.3 (Sept. 2015), pp. 223–234.

[Aug96] G. Augst. "Germany: script and politics." In: *The World's Writing Systems*. Ed. by P. T. Daniels and W. Bright. New York: Oxford University Press, 1996, pp. 765–768.

[AV07] D. Arthur and S. Vassilvitskii. "K-means++: The Advantages of Careful Seeding." In: *Proc. of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[BAE18] B. K. Barakat, R. Alasam, and J. El-Sana. "Word spotting using convolutional siamese network." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 229–234.

[Bal05] N. Balakrishnan. "Universal Digital Library: Future research directions." In: *Journal of Zhejiang University Science* 6A.11 (2005), pp. 1204–1205.

[Bal81] D. Ballard. "Generalizing the Hough transform to detect arbitrary shapes." In: *Pattern Recognition* 13.2 (1981), pp. 111–122.

[BDG+05] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. "Clustering on the unit hypersphere using von Mises-Fisher Distributions." In: *Journal of Machine Learning Research* 6 (Dec. 2005), pp. 1345–1382.

[Bha43] A. Bhattacharyya. "On a measure of divergence between two statistical populations defined by their probability distributions." In: *Bulletin of Cal. Math. Soc.* 35.1 (1943), pp. 99–109.

[BHK+17] T. Bluche, S. Hamel, C. Kermorvant, J. Puigcerver, D. Stutzmann, A. H. Toselli, and E. Vidal. "Preparatory KWS experiments for large-scale indexing of a vast medieval manuscript collection in the HIMANIS project." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 01. Nov. 2017, pp. 311–316.

[BHK97] C. Barrett, R. Hughey, and K. Karplus. "Scoring hidden Markov models." In: *Comput Appl Biosci* 13.2 (1997), pp. 191–199.

[BHM16] B. Bogacz, N. Howe, and H. Mara. "Segmentation free spotting of cuneiform using part structured models." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Oct. 2016, pp. 301–306.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[BM18a] B. Bogacz and H. Mara. "Feature descriptors for spotting 3D characters on triangular meshes." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 363–368.

[BM18b] B. Bogacz and H. Mara. "From extraction to spotting for cuneiform script analysis." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 199–204.

[BMC+15] G. Bideault, L. Mioulet, C. Chatelain, and T. Paquet. "Benchmarking discriminative approaches for word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 201–205.

[BR11] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition*. Pearson Education Ltd., Harlow, England, 2011.

[BWG10] S. Bengio, J. Weston, and D. Grangier. "Label embedding trees for large multi-class tasks." In: *Advances in Neural Information Processing Systems 23*. Ed. by J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. Curran Associates, Inc., 2010, pp. 163–171.

[CBG09] H. Cao, A. Bhardwaj, and V. Govindaraju. "A probabilistic method for keyword retrieval in handwritten document images." In: *Pattern Recognition* 42.12 (2009). New Frontiers in Handwriting Recognition, pp. 3374–3382.

[CDF+04] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. "Visual categorization with bags of keypoints." In: *Workshop on Statistical Learning in Computer Vision*. 2004, pp. 1–22.

[CG07] H. Cao and V. Govindaraju. "Template-free word spotting in low-quality manuscripts." In: *Int. Conf. on Advances in Pattern Recognition*. 2007, pp. 45–53.

[CGS+18] T. Causer, K. Grint, A.-M. Sichani, and M. Terras. "Making such bargain: Transcribe Bentham and the quality and cost-effectiveness of crowdsourced transcription." In: *Digital Scholarship in the Humanities* 33.3 (2018), pp. 467–487.

[CLV+11] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. "The devil is in the details: an evaluation of recent feature encoding methods." In: *British Machine Vision Conf.* Ed. by J. Hoey, S. McKenna, and E. Trucco. BMVA Press, 2011, pp. 76.1–76.12.

[Coh95] P. R. Cohen. *Empirical methods for artificial intelligence*. Cambridge, MA, USA: MIT Press, 1995.

[CRM00] D. Comaniciu, V. Ramesh, and P. Meer. "Real-time tracking of non-rigid objects using mean shift." In: *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Vol. 2. 2000, 142–149 vol.2.

[CSV+14] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. "Return of the devil in the details: delving deep into convolutional nets." In: *British Machine Vision Conf.* 2014. arXiv: 1405.3531 [cs].

[CT14] T. Causer and M. Terras. "Many hands make light work, many hands together make merry work: Transcribe Bentham and crowdsourcing manuscript collections." In: *M. Ridge (ed.), Crowdsourcing Our Cultural Heritage (Ashgate)*. 2014.

[CTV18] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal. "Probabilistic music-symbol spotting in handwritten scores." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 558–563.

[CWB93] F. R. Chen, L. D. Wilcox, and D. S. Bloomberg. "Word spotting in scanned images using hidden Markov models." In: *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing*. Vol. 5. Apr. 1993, 1–4 vol.5.

[CZF06] J. Chan, C. Ziftci, and D. Forsyth. "Searching off-line Arabic documents." In: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*. Vol. 2. 2006, pp. 1455–1462.

[Dan80] P.-E. Danielsson. "Euclidean distance mapping." In: *Computer Graphics and Image Processing* 14.3 (1980), pp. 227–248.

[DBW08] M. Donoser, H. Bischof, and S. Wagner. "Using web search engines to improve text recognition." In: *Proc. of the Int. Conf. on Pattern Recognition*. Dec. 2008, pp. 1–4.

[DDF+90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. "Indexing by latent semantic analysis." In: *Journal of the American Society for Information Science* 41.6 (1990), pp. 391–407.

[DHS00] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2Nd Edition)*. Wiley-Interscience, 2000.

[DKM+18] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar. "Towards spotting and recognition of handwritten documents in Indic scripts." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 32–37.

[DT05] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." In: *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Vol. 1. June 2005, pp. 886–893.

[EEV+15] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "The Pascal Visual Object Classes Challenge: A Retrospective." In: *International Journal of Computer Vision* 111.1 (Jan. 2015), pp. 98–136.

[Elk06] C. Elkan. "Clustering Documents with an Exponential-family Approximation of the Dirichlet Compound Multinomial Distribution." In: *Proc. of the Int. Conf. on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 289–296.

[ET98] B. Efron and R. J. Tibshirani. *An introduction to the bootstrap*. Monographs on Statistics and Applied Probability 57. Boca Raton, Florida, USA: Chapman & Hall/CRC, 1998.

[ETF+04] J. Edwards, Y. W. The, D. Forsyth, R. Bock, M. Maire, and G. Vesom. "Making latin manuscripts searchable using gHMMs." In: *Proc. of the Int. Conf. on Neural Information Processing Systems*. NIPS'04. Vancouver, British Columbia, Canada: MIT Press, 2004, pp. 385–392.

[FB14] V. Frinken and H. Bunke. "Continuous handwritten script recognition." In: *Handbook of Document Image Processing and Recognition*. Ed. by D. Doermann and K. Tombre. London: Springer-Verlag, 2014, pp. 391–425.

[FEH+11] C. Forbes, M. Evans, N. Hastings, and B. Peacock. *Statistical distributions*. 4th ed. Wiley series in probability and statistics. Hoboken, New Jersey: John Wiley & Sons, Inc., 2011.

[FFB+13] A. Fischer, V. Frinken, H. Bunke, and C. Y. Suen. "Improving HMM-based keyword spotting with character language models." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 506–510.

[FFB10] V. Frinken, A. Fischer, and H. Bunke. "A novel word spotting algorithm using bidirectional long short-term memory neural networks." In: *Workshop on Artificial Neural Networks in Pattern Recognition*. Cairo, Egypt: Springer Berlin Heidelberg, 2010, pp. 185–196.

[FFF+11] A. Fornés, V. Frinken, A. Fischer, J. Almazán, G. Jackson, and H. Bunke. "A keyword spotting approach using blurred shape model-based descriptors." In: *Proc. of the Int. Workshop on Historical Document Imaging and Processing*. HIP '11. Beijing, China, USA: ACM, 2011, pp. 83–90.

[FFM+12] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke. "A novel word spotting method based on recurrent neural networks." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 34.2 (2012).

[FGM+10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. "Object detection with discriminatively trained part-based models." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 32.9 (Sept. 2010), pp. 1627–1645.

[Fin14] G. A. Fink. *Markov Models for Pattern Recognition, From Theory to Applications*. 2nd ed. Advances in Computer Vision and Pattern Recognition. London: Springer, 2014.

[FKF+10] A. Fischer, A. Keller, V. Frinken, and H. Bunke. "HMM-based word spotting in handwritten documents using subword models." In: *Proc. of the Int. Conf. on Pattern Recognition*. 2010, pp. 3416–3419.

[FKF+12] A. Fischer, A. Keller, V. Frinken, and H. Bunke. "Lexicon-free handwritten word spotting using character HMMs." In: *Pattern Recognition Letters* 33.7 (2012), pp. 934–942.

[FKG10] B. A. Frigyik, A. Kapila, and M. R. Gupta. *Introduction to the Dirichlet distribution and related processes*. Tech. rep. UWEETR-2010-0006. Department of Electrical Engineering, University of Washington, 2010.

[FLF11] D. Fernández, J. Lladós, and A. Fornés. "Handwritten word spotting in old manuscript images using a pseudo-structural descriptor organized in a hash structure." In: *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2011)*. Ed. by J. Vitrià, J. M. Sanches,

and M. Hernández. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 628–635.

[FMF+14] D. Fernández-Mota, R. Manmatha, A. Fornés, and J. Lladós. "Sequential word spotting in historical handwritten documents." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2014, pp. 101–105.

[FML+13] D. Fernández, S. Marinai, J. Lladós, and A. Fornés. "Contextual word spotting in historical manuscripts using Markov logic networks." In: *Proc. of the Int. Workshop on Historical Document Imaging and Processing*. HIP '13. Washington, District of Columbia, USA: ACM, 2013, pp. 36–43.

[FP05] L. Fei-Fei and P. Perona. "A Bayesian hierarchical model for learning natural scene categories." In: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*. Vol. 2. June 2005, 524–531 vol. 2.

[FP08] G. A. Fink and T. Plötz. "Developing pattern recognition systems based on Markov models: The ESMERALDA framework." In: *Pattern Recognition and Image Analysis* 18.2 (2008), pp. 207–215.

[FRG14] G. A. Fink, L. Rothacker, and R. Grzeszick. "Grouping historical postcards using query-by-example word spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2014, pp. 470–475.

[FU15] V. Frinken and S. Uchida. "Deep BLSTM neural networks for unconstrained continuous handwritten text recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 911–915.

[Gat14] B. G. Gatos. "Imaging techniques in document analysis processes." In: *Handbook of Document Image Processing and Recognition*. Ed. by D. Doermann and K. Tombre. London: Springer-Verlag, 2014, pp. 73–131.

[GDD+16] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Region-based convolutional networks for accurate object detection and segmentation." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 38.1 (Jan. 2016), pp. 142–158.

[GG92] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Communications and Information Theory. Boston: Kluwer Academic Publishers, 1992.

[Glu67] H. Glucksman. "Classification of mixed-font alphabetics by characteristic loci." In: *Proc. of the IEEE Computer Conf.* Sept. 1967, pp. 138–141.

[GP09] B. Gatos and I. Pratikakis. "Segmentation-free word spotting in historical printed documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. 2009.

[GRF13] R. Grzeszick, L. Rothacker, and G. A. Fink. "Bag-of-features representations using spatial visual vocabularies for object classification." In: *Int. Conf. on Image Processing*. Sept. 2013, pp. 2867–2871.

[GRK17] L. Gómez, M. Rusiñol, and D. Karatzaz. "LSDE: Levenshtein space deep embedding for query-by-string word spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 01. Nov. 2017, pp. 499–504.

[GSF18] N. Gurjar, S. Sudholt, and G. A. Fink. "Learning Deep Representations for Word Spotting under Weak Supervision." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 7–12.

[GSG+17] A. P. Giotis, G. Sfikas, B. Gatos, and C. Nikou. "A survey of document image word spotting techniques." In: *Pattern Recognition* 68 (2017), pp. 310–332.

[GSW03] J. M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer. "Fast anisotropic Gauss filtering." In: *IEEE Trans. on Image Processing* 12.8 (Aug. 2003), pp. 938–943.

[GV15a] S. Ghosh and E. Valveny. "Query by string word spotting based on character bi-gram indexing." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 881–885.

[GV15b] S. K. Ghosh and E. Valveny. "A sliding window framework for word spotting based on word attributes." In: *Pattern Recognition and Image Analysis*. Ed. by R. Paredes, J. S. Cardoso, and X. M. Pardo. Vol. 9117. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 652–661.

[GV17] S. K. Ghosh and E. Valveny. "R-PHOC: Segmentation-free word spotting using CNN." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 01. Nov. 2017, pp. 801–806.

[GV18] S. Ghosh and E. Valveny. "Text box proposals for handwritten word spotting from documents." In: *Int. Journal on Document Analysis and Recognition* (Apr. 2018).

[GW02] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 2002.

[HF16] A. Hast and A. Fornés. "A segmentation-free handwritten word spotting approach by relaxed feature matching." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2016, pp. 150–155.

[How13] N. R. Howe. "Part-structured inkball models for one-shot handwritten word spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 582–586.

[HRM05] N. R. Howe, T. M. Rath, and R. Manmatha. "Boosted decision trees for word recognition in handwritten document retrieval." In: *Proc. of the Int. Conf. on Research and Development in Information Retrieval*. SIGIR '05. Salvador, Brazil: ACM, 2005, pp. 377–383.

[HS88] C. Harris and M. Stephens. "A combined corner and edge detector." In: *Proc. of Fourth Alvey Vision Conf.* 1988, pp. 147–151.

[HSW09] T. J. Hazen, W. Shen, and C. White. "Query-by-example spoken term detection using phonetic posteriorgram templates." In: *Workshop on Automatic Speech Recognition Understanding*. Nov. 2009, pp. 421–426.

[Hul94] J. J. Hull. "Document image matching and retrieval with multiple distortion-invariant descriptors." In: *Proc. of the Int. Workshop on Document Analysis Systems*. 1994, pp. 383–400.

[JDS11] H. Jegou, M. Douze, and C. Schmid. "Product quantization for nearest neighbor search." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33.1 (Jan. 2011), pp. 117–128.

[JKB97] N. Johnson, S. Kotz, and N. Balakrishnan. *Discrete multivariate distributions*. A Wiley-Interscience publication. New York, NY: Wiley, 1997. XXII, 299.

[JVZ14] M. Jaderberg, A. Vedaldi, and A. Zisserman. "Deep features for text spotting." In: *Proc. of. the European Conf. on Computer Vision*. Cham: Springer International Publishing, 2014, pp. 512–528.

[KA94] S.-S. Kuo and O. E. Agazzi. "Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 16.8 (Aug. 1994), pp. 842–848.

[KAA+00] A. Kolcz, J. Alspector, M. Augusteijn, R. Carlson, and G. Viorel Popescu. "A line-oriented approach to word spotting in handwritten documents." In: *Pattern Analysis & Applications* 3.2 (2000), pp. 153–168.

[KDJ16] P. Krishnan, K. Dutta, and C. Jawahar. "Deep feature embedding for accurate recognition and retrieval of handwritten text." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2016, pp. 289–294.

[KDJ18] P. Krishnan, K. Dutta, and C. Jawahar. "Word Spotting and Recognition Using Deep Embedding." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 1–6.

[KDN13] M. Kozielski, P. Doetsch, and H. Ney. "Improvements in RWTH's system for off-line handwriting recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Washington, DC, USA, Aug. 2013, pp. 935–939.

[KGG97] P. Keaton, H. Greenspan, and R. Goodman. "Keyword spotting for cursive document retrieval." In: *Workshop on Document Image Analysis*. June 1997, pp. 74–81.

[KH93] S. Khoubyari and J. J. Hull. "Keyword location in noisy document image." In: *Symposium on Document Analysis and Information Retrieval*. 1993, pp. 217–231.

[Kis14] K. Kise. "Page segmentation techniques in document analysis." In: *Handbook of Document Image Processing and Recognition*. Ed. by D. Doermann and K. Tombre. London: Springer-Verlag, 2014, pp. 135–175.

[KKG15] T. Konidaris, A. L. Kesidis, and B. Gatos. "A segmentation-free word spotting method for historical printed documents." In: *Pattern Analysis and Applications* (2015), pp. 1–14.

[KLP01] S. Kane, A. Lehman, and E. Partridge. *Indexing George Washington's handwritten manuscripts*. Tech. rep. MM-34. Center for Intelligent Information Retrieval, University of Massachusetts Amherst, 2001.

[KWD14] A. Kovalchuk, L. Wolf, and N. Dershowitz. "A simple and fast word spotting method." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2014, pp. 3–8.

[LBE05] Y. Leydier, F. L. Bourgeois, and H. Emptoz. "Omnilingual segmentation-free word spotting for ancient manuscripts indexation." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2005, 533–537 Vol. 1.

[LFG12] Y. Liang, M. Fairhurst, and R. Guest. "A synthesised word approach to word retrieval in handwritten documents." In: *Pattern Recognition* 45.12 (2012), pp. 4225–4236.

[LLE07] Y. Leydier, F. Lebourgeois, and H. Emptoz. "Text search for medieval manuscript images." In: *Pattern Recognition* 40.12 (2007), pp. 3552–3567.

[LLM+13] Q. Liao, J. Z. Leibo, Y. Mroueh, and T. Poggio. "Can a biologically-plausible hierarchy effectively replace face detection, alignment, and recognition pipelines?" In: *ArXiv e-prints* (Nov. 2013). arXiv: 1311.4082 [cs.CV].

[Llo82] S. P. Lloyd. "Least squares quantization in PCM." In: *IEEE Trans. on Information Theory* 28.2 (1982).

[LM81] D. G. Long and A. T. Milne. *The manuscripts of Jeremy Bentham : a chronological index to the collection in the Library of University College London*. Library and Bentham Committee, University College London, 1981.

[LOL+09] Y. Leydier, A. Ouji, F. LeBourgeois, and H. Emptoz. "Towards an omnilingual word retrieval system for ancient manuscripts." In: *Pattern Recognition* 42.9 (2009).

[Low04] D. G. Lowe. "Distinctive image features from scale-invariant keypoints." In: *Int. Journal of Computer Vision* 60 (2004).

[LPT+18] E. Lang, J. Puigcerver, A. Toselli, and E. Vidal. "Probabilistic indexing and search for information extraction on handwritten German parish records." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 44–49.

[LRF+12] J. Lladós, M. Rusiñol, A. Fornés, D. F. Mota, and A. Dutta. "On the influence of word representations for handwritten word spotting in historical documents." In: *Int. Journal on Pattern Recognition and Artificial Intelligence* 26.5 (2012).

[LRM04] V. Lavrenko, T. M. Rath, and R. Manmatha. "Holistic word recognition for handwritten historical documents." In: *Workshop on Document Image Analysis for Libraries*. 2004, pp. 278–287.

[LS07] J. Lladós and G. Sanchez. "Indexing historical documents by word shape signatures." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 1. Sept. 2007, pp. 362–366.

[LSP06] S. Lazebnik, C. Schmid, and J. Ponce. "Beyond bags of features: spatial pyramid matching for recognizing natural scene categories." In: *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Vol. 2. 2006, pp. 2169–2178.

[Mac67] J. MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations." In: *Proc. of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Ed. by L. M. Le Cam and J. Neyman. Vol. 1. 1967, pp. 281–296.

[MC09] R. F. Moghaddam and M. Cheriet. "Application of multi-level classifiers and clustering for automatic word spotting in historical document images." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. 2009, pp. 511–515.

[MCU+04] J. Matas, O. Chum, M. Urban, and T. Pajdla. "Robust wide-baseline stereo from maximally stable extremal regions." In: *Image and Vision Computing* 22.10 (2004). British Machine Vision Computing 2002, pp. 761–767.

[MGE11] T. Malisiewicz, A. Gupta, and A. A. Efros. "Ensemble of exemplar-SVMs for object detection and beyond." In: *Proc. of the Int. Conf. on Computer Vision*. Nov. 2011, pp. 89–96.

[MHR96] R. Manmatha, C. Han, and E. Riseman. "Word spotting: a new approach to indexing handwriting." In: *Proc. of the IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. 1996.

[MKE05] R. E. Madsen, D. Kauchak, and C. Elkan. "Modeling Word Burstiness Using the Dirichlet Distribution." In: *Proc. of the Int. Conf. on Machine Learning*. ICML '05. Bonn, Germany: ACM, 2005, pp. 545–552.

[MKM07] T. Masada, S. Kiyasu, and S. Miyahara. "Clustering images with multinomial mixture models." In: *Int. Symposium on Advanced Intelligent Systems*. 2007, pp. 343–348.

[MMS03] S. Marinai, E. Marino, and G. Soda. "Indexing and retrieval of words in old documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 1. Aug. 2003, pp. 223–227.

[MN98] A. McCallum and K. Nigam. "A Comparison of Event Models for Naive Bayes Text Classification." In: *Learning for Text Categorization: Papers from the 1998 AAAI Workshop*. 1998, pp. 41–48.

[MR05] R. Manmatha and J. L. Rothfeder. "A scale space approach for automatically segmenting words from historical handwritten documents." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 27.8 (Aug. 2005), pp. 1212–1225.

[MRR+16] T. Mondal, N. Ragot, J.-Y. Ramel, and U. Pal. "Flexible sequence matching technique: an effective learning-free approach for word spotting." In: *Pattern Recognition* 60 (2016), pp. 596–612.

[MZ05] J. Matas and K. Zimmermann. "A new class of learnable detectors for categorisation." In: *Image Analysis*. Ed. by H. Kalviainen, J. Parkkinen, and A. Kaarna. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 541–550.

[NG06] A. Neubeck and L. V. Gool. "Efficient Non-Maximum Suppression." In: *Proc. of the Int. Conf. on Pattern Recognition*. Vol. 3. 2006, pp. 850–855.

[NM16] L. Neumann and J. Matas. "Real-time lexicon-free scene text localization and recognition." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 38.9 (Sept. 2016), pp. 1872–1885.

[NMT+00] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. "Text Classification from Labeled and Unlabeled Documents using EM." In: *Machine Learning* 39.2 (May 2000), pp. 103–134.

[NS06] D. Nister and H. Stewenius. "Scalable recognition with a vocabulary tree." In: *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. Vol. 2. 2006, pp. 2161–2168.

[OD11] S. O'Hara and B. A. Draper. "Introduction to the bag of features paradigm for image classification and retrieval." In: *ArXiv e-prints* (Jan. 2011). arXiv: 1101.3354 [cs.CV].

[OLB+10] F. W. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark. *NIST Handbook of Mathematical Functions*. 1st. New York, NY, USA: Cambridge University Press, 2010.

[PDF+14] W. Pantke, M. Dennhardt, D. Fecker, V. Märgner, and T. Fingscheidt. "An historical handwritten Arabic dataset for segmentation-free word spotting - HADARA80P." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2014, pp. 15–20.

[PF05] T. Plötz and G. A. Fink. "Robust remote homology detection by feature based profile hidden Markov models." In: *Statistical Applications in Genetics and Molecular Biology* 4.1 (2005).

[PF09] T. Plötz and G. A. Fink. "Markov models for offline handwriting recognition: a survey." In: *Int. Journal on Document Analysis and Recognition* 12.4 (2009), pp. 269–298.

[PR09] F. Perronnin and J. A. Rodríguez-Serrano. "Fisher kernels for handwritten word-Spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. July 2009, pp. 106–110.

[PSM10] F. Perronnin, J. Sánchez, and T. Mensink. "Improving the Fisher kernel for large-scale image classification." In: *Proc. of. the European Conf. on Computer Vision*. Ed. by K. Daniilidis, P. Maragos, and N. Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 143–156.

[PTV14] J. Puigcerver, A. H. Toselli, and E. Vidal. "Word-graph and character-lattice combination for KWS in handwritten documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2014, pp. 181–186.

[PTV15a] J. Puigcerver, A. H. Toselli, and E. Vidal. "Probabilistic interpretation and improvements to the HMM-filler for handwritten keyword spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 731–735.

[PTV15b] J. Puigcerver, A. Toselli, and E. Vidal. "ICDAR2015 competition on keyword spotting for handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. 2015, pp. 1176–1180.

[PZG+14] I. Pratikakis, K. Zagoris, B. Gatos, G. Louloudis, and N. Stamatopoulos. "ICFHR 2014 competition on handwritten keyword spotting (H-KWS 2014)." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2014, pp. 814–819.

[PZG+16] I. Pratikakis, K. Zagoris, B. Gatos, J. Puigcerver, A. Toselli, and E. Vidal. "ICFHR2016 handwritten keyword spotting competition (H-KWS 2016)." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2016, pp. 613–618.

[RAT+11] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. "Browsing heterogeneous document collections by a segmentation-free word spotting method." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Sept. 2011, pp. 63–67.

[RAT+15a] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. "Efficient segmentation-free keyword spotting in historical document collections." In: *Pattern Recognition* 48.2 (2015), pp. 545–555.

[RAT+15b] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. "Towards query-by-speech handwritten keyword spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 501–505.

[RB09] K. Riesen and H. Bunke. "Approximate graph edit distance computation by means of bipartite graph matching." In: *Image and Vision Computing* 27.7 (2009), pp. 950–959.

[RDE+14] I. Rabaev, I. Dinstein, J. El-Sana, and K. Kedem. "Segmentation-free keyword retrieval in historical document images." In: *Proc. of the Int. Conf. on Image Analysis and Recognition*. Ed. by A. Campilho and M. Kamel. Cham: Springer International Publishing, 2014, pp. 369–378.

[RF15] L. Rothacker and G. A. Fink. "Segmentation-free query-by-string word spotting with bag-of-features HMMs." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 661–665.

[RF16] L. Rothacker and G. A. Fink. "Robust output modeling in bag-of-features HMMs for handwriting recognition." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Oct. 2016, pp. 199–204.

[RFB+13] L. Rothacker, G. A. Fink, P. Banerjee, U. Bhattacharya, and B. B. Chaudhuri. "Bag-of-features HMMs for segmentation-free Bangla word spotting." In: *Proc. of the Int. Workshop on Multilingual OCR*. MOCR '13. New York, NY, USA: ACM, Aug. 2013, 5:1–5:5.

[RFM+15] L. Rothacker, D. Fisseler, G. G. W. Müller, F. Weichert, and G. A. Fink. "Retrieving cuneiform structures in a segmentation-free word spotting framework." In: *Proc. of the Int. Workshop on Historical Document Imaging and Processing*. HIP '15. New York, NY, USA: ACM, Aug. 2015, pp. 129–136.

[RFR03] J. L. Rothfeder, S. Feng, and T. M. Rath. "Using corner feature correspondences to rank word images by similarity." In: *Computer Vision and Pattern Recognition Workshop*. Vol. 3. June 2003, pp. 30–35.

[RHG+15] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." In: *Advances in Neural Information Processing Systems 28*. Ed. by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett. Curran Associates, Inc., 2015, pp. 91–99.

[RKE16] I. Rabaev, K. Kedem, and J. El-Sana. "Keyword retrieval using scale-space pyramid." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2016, pp. 144–149.

[RL12] M. Rusiñol and J. Lladós. "The role of the users in handwritten word spotting applications: query fusion and relevance feedback." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2012, pp. 55–60.

[RL14] M. Rusiñol and J. Lladós. "Boosting the handwritten word spotting experience by including the user in the loop." In: *Pattern Recognition* 47.3 (2014), pp. 1063–1072.

[RLF15] P. Riba, J. Lladós, and A. Fornés. "Handwritten word spotting by inexact matching of grapheme graphs." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 781–785.

[RLM03] T. M. Rath, V. Lavrenko, and R. Manmatha. *A statistical approach to retrieving historical manuscript images without recognition*. Tech. rep. ADA478157. SPACE and NAVAL WARFARE SYSTEMS CENTER SAN DIEGO CA, 2003.

[RLS+18] G. Retsinas, G. Louloudis, N. Stamatopoulos, and B. Gatos. "Efficient learning-free keyword spotting." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* (2018). To appear.

[RM03] T. M. Rath and R. Manmatha. "Word image matching using dynamic time warping." In: *Proc. of the Int. Conf. on Computer Vision and Pattern Recognition*. Vol. 2. June 2003, pp. 521–527.

[RM07] T. Rath and R. Manmatha. "Word spotting for historical documents." In: *Int. Journal on Document Analysis and Recognition* 9.2–4 (2007).

[Rot11] L. Rothacker. "Learning bag-of-features representations for handwriting recognition." Diploma thesis. Dortmund, Germany: TU Dortmund University, Nov. 2011.

[RP08a] J. Rodríguez-Serrano and F. Perronnin. "Local gradient histogram features for word spotting in unconstrained handwritten documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2008, pp. 7–12.

[RP08b] J. A. Rodríguez and F. Perronnin. "Score normalization for HMM-based word spotting using universal background model." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2008, pp. 82–87.

[RP09a] J. A. Rodríguez-Serrano and F. Perronnin. "Handwritten word image retrieval with synthesized typed queries." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. July 2009, pp. 351–355.

[RP09b] J. Rodríguez-Serrano and F. Perronnin. "Handwritten word-spotting using hidden Markov models and universal vocabularies." In: *Pattern Recognition* 42.9 (2009).

[RP12a] J. Rodríguez-Serrano and F. Perronnin. "A model-based sequence similarity with application to handwritten word spotting." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2108–2120.

[RP12b] J. A. Rodríguez-Serrano and F. Perronnin. "Synthesizing queries for handwritten word image retrieval." In: *Pattern Recognition* 45.9 (2012), pp. 3270–3276.

[RP90] R. C. Rose and D. B. Paul. "A hidden Markov model based keyword recognition system." In: *Proc. of the Int. Conf. on Acoustics, Speech, and Signal Processing*. Apr. 1990, pp. 129–132.

[RPL+09] J. A. Rodríguez-Serrano, F. Perronnin, J. Lladós, and G. Sánchez. "A similarity measure between vector sequences with application to handwritten word image retrieval." In: *Proc. IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*. 2009, pp. 1722–1729.

[RRF13] L. Rothacker, M. Rusiñol, and G. A. Fink. "Bag-of-features HMMs for segmentation-free word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 1305–1309.

[RRL+14] L. Rothacker, M. Rusiñol, J. Lladós, and G. A. Fink. "A two-stage approach to segmentation-free query-by-example word spotting." In: *manuscript cultures* 7 (2014), pp. 47–57.

[RRM+18] E. Rusakov, L. Rothacker, H. Mo, and G. A. Fink. "A probabilistic retrieval model for word spotting based on direct attribute prediction." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 38–43.

[RSL+17] G. Retsinas, N. Stamatopoulos, G. Louloudis, G. Sfikas, and B. Gatos. "Nonlinear manifold embedding on keyword spotting using t-SNE." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Vol. 01. Nov. 2017, pp. 487–492.

[RSL+18] G. Retsinas, G. Sfikas, G. Louloudis, N. Stamatopoulos, and B. Gatos. "Compact deep descriptors for keyword spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Aug. 2018, pp. 315–320.

[RSR+17] L. Rothacker, S. Sudholt, E. Rusakov, M. Kasperidus, and G. A. Fink. "Word hypotheses for segmentation-free word spotting in historic document images." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nov. 2017, pp. 1174–1179.

[RSS+18] G. Retsinas, G. Sfikas, N. Stamatopoulos, G. Louloudis, and B. Gatos. "Exploring critical aspects of CNN-based keyword spotting. A PHOCNet study." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 13–18.

[RST+03] J. D. M. Rennie, L. Shih, J. Teevan, and D. R. Karger. "Tackling the poor assumptions of naive Bayes text classifiers." In: *Proc. of the Int. Conf. on Machine Learning*. ICML'03. Washington, DC, USA: AAAI Press, 2003, pp. 616–623.

[RVF12] L. Rothacker, S. Vajda, and G. A. Fink. "Bag-of-features representations for offline handwriting recognition applied to Arabic script." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2012, pp. 1490–154.

[SAC07] M. D. Smucker, J. Allan, and B. Carterette. "A comparison of statistical significance tests for information retrieval evaluation." In: *Proc. of the ACM Conf. on Information and Knowledge Management*. CIKM '07. Lisbon, Portugal: ACM, 2007, pp. 623–632.

[SC78] H. Sakoe and S. Chiba. "Dynamic programming algorithm optimization for spoken word recognition." In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26.1 (Feb. 1978), pp. 43–49.

[SF15] S. Sudholt and G. A. Fink. "A modified isomap approach to manifold learning in word spotting." In: *Proc. of the German Conf. on Pattern Recognition*. 2015, pp. 529–539.

[SF16] S. Sudholt and G. A. Fink. "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Oct. 2016, pp. 277–282.

[SF17] S. Sudholt and G. A. Fink. "Evaluating word string embeddings and loss functions for CNN-based word spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nov. 2017, pp. 493–498.

[SF18] S. Sudholt and G. A. Fink. "Attribute CNNs for word spotting in handwritten documents." In: *Int. Journal on Document Analysis and Recognition* 21.3 (Sept. 2018), pp. 199–218.

[SFR18a] M. Stauffer, A. Fischer, and K. Riesen. "Graph-based keyword spotting in historical documents using context-aware Hausdorff edit distance." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2018, pp. 49–54.

[SFR18b] M. Stauffer, A. Fischer, and K. Riesen. "Filters for graph-based keyword spotting in historical handwritten documents." In: *Pattern Recognition Letters* (2018). To appear.

[SGL+15] G. Sfikas, A. P. Giotis, G. Louloudis, and B. Gatos. "Using attributes for word spotting and recognition in polytonic greek documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 686–690.

[SGL+16] T. Strauß, T. Grüning, G. Leifert, and R. Labahn. "CITlab ARGUS for keyword search in historical handwritten documents: Description of CITlab's system for the ImageCLEF 2016 handwritten scanned document retrieval task." In: *Working Notes of CLEF (Conference and Labs of the Evaluation forum)*. 2016, pp. 399–412.

[SJ12] R. Shekhar and C. Jawahar. "Word image retrieval using bag of visual words." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Mar. 2012, pp. 297–301.

[SK15] A. Sharma and P. S. K. "Adapting off-the-shelf CNNs for word spotting & recognition." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 986–990.

[Smi07] R. Smith. "An overview of the tesseract OCR engine." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. 2007, pp. 629–633.

[SPC12] J. Sánchez, F. Perronnin, and T. de Campos. "Modeling the spatial layout of images beyond spatial pyramids." In: *Pattern Recognition Letters* 33.16 (2012), pp. 2216–2223.

[SRF15] S. Sudholt, L. Rothacker, and G. A. Fink. "Learning local image descriptors for word spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 651–655.

[SRF17] S. Sudholt, L. Rothacker, and G. A. Fink. "Query-by-online word spotting revisited: using CNNs for cross-domain retrieval." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Nov. 2017, pp. 481–486.

[SRG16] G. Sfikas, G. Retsinas, and B. Gatos. "Zoning aggregated hypercolumns for keyword spotting." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2016, pp. 283–288.

[SRM+11] W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult. "Meta-Recognition: The Theory and Practice of Recognition Score Analysis." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33.8 (Aug. 2011), pp. 1689–1695.

[SZ03] J. Sivic and A. Zisserman. "Video Google: a text retrieval approach to object matching in videos." In: *Proc. of the Int. Conf. on Computer Vision*. Vol. 2. Oct. 2003, pp. 1470–1477.

[Sze11] R. Szeliski. *Computer Vision - Algorithms and Applications*. Texts in Computer Science. Springer, 2011.

[TCH+15] S. Thomas, C. Chatelain, L. Heutte, T. Paquet, and Y. Kessentini. "A deep HMM model for multiple keywords spotting in handwritten documents." In: *Pattern Analysis and Applications* 18.4 (2015), pp. 1003–1015.

[TG14] S. Tulyakov and V. Govindaraju. "Handprinted character and word recognition." In: *Handbook of Document Image Processing and Recognition*. Ed. by D. Doermann and K. Tombre. London: Springer-Verlag, 2014, pp. 359–389.

[TNK05] K. Terasawa, T. Nagasaki, and T. Kawashima. "Eigenspace method for text retrieval in historical document images." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2005, pp. 437–441.

[TPV15] A. H. Toselli, J. Puigcerver, and E. Vidal. "Context-aware lattice based filler approach for key word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 736–740.

[TPV16] A. H. Toselli, J. Puigcerver, and E. Vidal. "Two methods to improve confidence scores for lexicon-free word spotting in handwritten text." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2016, pp. 349–354.

[TT09] K. Terasawa and Y. Tanaka. "Slit style HOG feature for document image word spotting." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. July 2009, pp. 116–120.

[TV13] A. H. Toselli and E. Vidal. "Fast HMM-filler approach for key word spotting in handwritten documents." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 501–505.

[TVR+16] A. H. Toselli, E. Vidal, V. Romero, and V. Frinken. "HMM word graph based keyword spotting in handwritten document images." In: *Information Sciences* 370-371 (2016), pp. 497–518.

[UN98] N. Ueda and R. Nakano. "Deterministic annealing EM algorithm." In: *Neural Networks* 11.2 (1998), pp. 271–282.

[VF08] A. Vedaldi and B. Fulkerson. *VLFeat: An open and portable library of computer vision algorithms*. `http://www.vlfeat.org/`. 2008.

[VTP15] E. Vidal, A. H. Toselli, and J. Puigcerver. "High performance query-by-example keyword spotting using query-by-string techniques." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2015, pp. 741–745.

[Was54] G. Washington. "George Washington papers." In: *Letterbook 1, Aug. 11, 1754 - Dec. 25, 1755*. Series 2, Letterbooks 1754 to 1799. Washington, DC: Library of Congress, 1754. URL: `https://www.loc.gov/item/mgw2.001/`.

[WB15] T. Wilkinson and A. Brun. "A novel word segmentation method based on object detection and deep learning." In: *Int. Symposium of Advances in Visual Computing*. 2015, pp. 231–240.

[WB16] T. Wilkinson and A. Brun. "Semantic and verbatim word spotting using deep neural networks." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2016, pp. 307–312.

[WEG+14a] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, and A. Fornés. "A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance." In: *Proc. of the Int. Conf. on Pattern Recognition*. Aug. 2014, pp. 3074–3079.

[WEG+14b] P. Wang, V. Eglin, C. Garcia, C. Largeron, J. Lladós, and A. Fornés. "A novel learning-free word spotting approach based on graph representation." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2014, pp. 207–211.

[WLB17] T. Wilkinson, J. Lindström, and A. Brun. "Neural Ctrl-F: Segmentation-free query-by-string word spotting in handwritten manuscript collections." In: *Proc. of the Int. Conf. on Computer Vision*. Oct. 2017, pp. 4443–4452.

[WRF16] C. Wieprecht, L. Rothacker, and G. A. Fink. "Word spotting in historical document collections with online-handwritten queries." In: *Proc. of the Int. Workshop on Document Analysis Systems*. Apr. 2016, pp. 162–167.

[ZE02] A. Zomorodian and H. Edelsbrunner. "Fast software for box intersections." In: *Int. Jounal of Comp. Geometry and Applications* 12.01n02 (Feb. 2002), pp. 143–172.

[ZPG14] K. Zagoris, I. Pratikakis, and B. Gatos. "Segmentation-based historical handwritten word spotting using document-specific local features." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. Sept. 2014, pp. 9–14.

[ZPG17] K. Zagoris, I. Pratikakis, and B. Gatos. "Unsupervised word spotting in historical handwritten document images using document-oriented local features." In: *IEEE Trans. on Image Processing* 26.8 (Aug. 2017), pp. 4032–4041.

[ZPJ+16] Z. Zhong, W. Pan, L. Jin, H. Mouchère, and C. Viard-Gaudin. "SpottingNet: Learning the similarity of word images with convolutional neural network for word spotting in handwritten historical documents." In: *Proc. of the Int. Conf. on Frontiers in Handwriting Recognition*. 2016, pp. 295–300.

[ZSH03] B. Zhang, S. N. Srihari, and C. Huang. "Word image retrieval using binary features." In: *Proc. SPIE, Document Recognition and Retrieval XI*. Vol. 5296. 2003, pp. 45–53.

[ZSH08] T. van der Zant, L. Schomaker, and K. Haak. "Handwritten-word spotting using biologically inspired features." In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 30.11 (Nov. 2008), pp. 1945–1957.

[ZT13] X. Zhang and C. L. Tan. "Segmentation-free keyword spotting for handwritten documents based on heat kernel signature." In: *Proc. of the Int. Conf. on Document Analysis and Recognition*. Aug. 2013, pp. 827–831.

LIST OF FIGURES

LIST OF TABLES