technische universität
dortmund

fakultät
statistik

Sequence Data Mining in Cognitive Science

DISSERTATION

in partial fulfillment of the requirements for the degree of
Doktor der Naturwissenschaften
submitted to the

Department of Statistics
TU Dortmund University

by

He Huang

July, 2024

Primary referee:	Prof. Dr. Philipp Doebler
Secondary referee:	Prof. Dr. Markus Pauly
Commission chairperson:	Prof. Dr. Guido Knapp
Assessor:	Dr. Uwe Ligges
Date of the oral examination: 30.09.2024	

Acknowledgments

First of all, I would like to express my deepest gratitude to my supervisor, Prof. Philipp Doebler. I am very grateful for his guidance, help, and encouragement during my PhD research. I met him for the first time when I was in my second year of undergraduate studies at the Technical University of Dortmund, and I am still very grateful to him for inviting me as a student assistant for one of his courses at that time. As a newcomer to Germany, this experience was a great encouragement and boosted my confidence to study in Germany.

In addition to my supervisor, I would like to thank Prof. Barbara Mertins. I am grateful to her for suggesting that I pursue a PhD after my master's degree and for talking with Prof. Philipp Doebler to co-sponsor my PhD research. She offered me a part-time position in her psycholinguistics lab, where I found inspiration for my research and was able to apply my findings to the field.

I would also like to thank my colleagues in the Department of Statistics, the Psycholinguistics Lab, and the FAIR group. I am very grateful for all the help and support they have given me in work and life, and I treasure the friendships I have formed with them.

Finally, I must thank my wife and my daughter. My wife and I have been studying and working in Germany for ten years together and I am very grateful for her company and encouragement. I could not have completed my PhD without her support. My daughter joined our family in the first year of my PhD and has brought infinite happiness to our family. She witnessed my whole PhD life and is currently back in China with my wife. I miss them deeply while completing this dissertation, and I eagerly look forward to completing my defense and reuniting with them soon.

Abstract

This thesis summarizes my research work over a five-year period from February 2020 to August 2024, including all of the papers I published during that time. As it is a cumulative, this thesis provides a concise overview of the contributed articles, omitting exhaustive results and instead referring to the original publications for full details. The main text integrates these publications into a coherent narrative, starting with basic concepts and providing background on the respective research areas. For an in-depth discussion of specific research findings, readers are recommended to consult the relevant articles directly.

This thesis covers the field of sequence data mining (SDM) in cognitive science. Cognitive science increasingly examines sequence data to understand cognitive tasks involving ordered steps or elements, such as language processing, decision-making, and memory formation. SDM techniques are used to uncover patterns and models within sequential data. However, modern data mining techniques like deep learning, which have been broadly applied in other domains, have not been fully integrated into traditional cognitive science tasks. Moreover, cognitive science deals with complex sequence data, such as scanpaths and trajectories, which pose challenges that traditional pattern discovery methods and modern techniques have not successfully overcome.

This thesis aims to extend SDM methods in cognitive science by focusing on the application of advanced techniques and the creation of new methods specifically tailored for handling these complex, domain-specific sequences. For instance, a machine learning-based pipeline for automatic scoring in diversity thinking tasks is proposed in one of my published papers, utilizing algorithms such as Random Forest, XGBoost, and Support Vector Regression. Another two papers introduce novel approaches to analysis scanpaths and handwritten trajectories. Through experimental validation in each paper, the newly developed methods demonstrate superior performance compared to existing approaches.

Overall, my research advances SDM by integrating modern data mining techniques to address the challenges posed by complex sequential data in cognitive science .

Contents

List of Contributed Articles

This thesis is based on the following articles, referred to by their Roman numbers throughout the text.

(I) Buczak, P., Huang, H., Forthmann, B., and Doebler, P. (2022). The Machines Take Over: A Comparison of Various Supervised Learning Approaches for Automated Scoring of Divergent Thinking Tasks. *The Journal of Creative Behavior, 57(1), 17-36, DOI: 10.1002/jocb.559*

Contribution of the author:
Philip Buczak and He Huang have contributed equally to this work and are listed here in alphabetical order. The authors of this paper prepared and structured the manuscript mainly on their own. They conducted extensive simulation studies with helpful comments from the coauthors. Furthermore, they chose and analyzed the data examples independently.

(II) Huang, H., Doebler, P., and Mertins, B. (2024). Short-time AOIs-based representative scanpath identification and scanpath aggregation. *Behavior Research Methods, 1-16.DOI: 10.3758/s13428-023-02332-w*

Contribution of the author:
The author of this thesis prepared and structured the manuscript mainly on his own. He conducted extensive simulation studies with helpful comments by the co-authors. Furthermore, he chose and analyzed the data examples independently.

(III) Huang, H. and Doebler, P. (conditionally accepted) Trajectory-based Handwriting Recognition via Spatiotemporal Convolution on Distance Matrix. This paper has been conditionally accepted by conference *In 2024 7th International Conference on Artificial Intelligence and Pattern Recognition*, which will be held from September 20th to 22nd in Xiamen, China.

Contribution of the author:
The author of this thesis prepared and structured the manuscript mainly on his own. He conducted extensive simulation studies with helpful comments from the coauthors. He chose and analyzed the data examples independently.

Notation

$\mathbb{R}$	Set of real numbers
S	Sequence
n, m	Length of sequences
t	Index of the elements in sequence
$\mathbf{x}_t$	t-th element of a sequence
$\mathbf{x}$	Vector
p	Number of features or dimension of $\mathbf{x}$
N	Number of observations
i	Index for observations
$\mathbf{D}$	Distance matrix of two sequences
f	Supervised machine learning model
L	Loss function
$\mathbf{w}$	Parameter vectors of SVM model
$\mathbf{b}$	Bias vector in SVM model and Neural Network model
$\mathbf{W}, \mathbf{U}$	Parameter matrix of Neural Network model
ξ, ξ^*	Slack variables in SVM model
$\mathcal{D}$	Traning data set

l	Index for the layers in Neural Network model
$\mathbf{h}^l$	Output vector of the l-th layer of Neural Network model

Part I

Summary of Thesis Work

1 Introduction

Cognitive science often deals with how humans and other animals process information, make decisions, learn, and adapt over time (Neisser, 2014). In recent years, cognitive science has increasingly turned its attention to the analysis of sequence data. This shift in focus underscores the recognition that many cognitive tasks involve a series of ordered steps or elements. Understanding and analyzing sequences of cognitive events, such as language processing, decision-making, problem-solving, and memory formation, has become crucial in unraveling the complexities of how the mind operates. By applying sequence analysis techniques, cognitive scientists can uncover patterns, dependencies, and structures within cognitive processes, leading to a deeper understanding of how the human mind represents and processes information.

1.1 Sequence Data in Cognitive Science

By sequence data (also referred to as "sequential data" or "temporal data" in literature), we are referring to data that is arranged or ordered based on a particular index. Given that cognitive science encompasses a diverse range of disciplines including psychology, neuroscience, linguistics, anthropology, artificial intelligence, philosophy, and education, the types of sequence data found in this field are varied. These types include but are not limited to:

- Neural activity data, such as electroencephalography (EEG) sequences, functional magnetic resonance imaging (fMRI) sequences, and magnetoencephalography (MEG) sequences.
- Language and linguistics data, such as speech transcripts, written text, and eye-tracking data in reading.
- Behavioral data, including action sequences used in problem-solving or game playing tasks, motion capture data, and gestural data.
- Developmental study data, particularly in studies related to child language acquisition.
- Human-computer interaction (HCI) sequences, for example, clickstream data.
- Learning and adaptation data, such as sequences of learning activities and outcomes over time.

The diverse types of sequence data within cognitive science pose significant challenges for data analysis. The heterogeneity of these data types requires specialized methods and tools for effective analysis, as each type can have unique characteristics and requirements. While it is not feasible to employ a unified analysis framework to handle all types of sequence data in cognitive science, we can categorize these data into a few fundamental types. Researchers can then select or develop appropriate analytical methods within these fundamental types based on their specific data characteristics and requirements. The five most foundational types of sequence data in cognitive science are (i) time series, (ii) symbolic sequence, (iii) text, (iv) trajectory, and (v) attributed trajectory. These types will be explained as follows:

(i) A time series involves a sequence of real numbers indexed in time order. This form of data is essential for understanding dynamic processes in neural activity and behavioral responses, such as EEG sequences, fMRI sequences, MEG sequences, and response time sequences.

(ii) A symbolic sequence refers to a sequence of symbols, characters, or discrete elements that represent abstract symbolic information, for example, in problem-solving tasks, a symbolic sequence could be {"responding to an email", "seeking help", "keystrokes", "creating new folder", "using the toolbar", "opening folders"}, see Ulitzsch et al. (2022). In addition, in eye-tracking studies, a sequence of letters like {A, C, D, E}, where each letter represents an Area of Interest (AOI), is used to represent the view behavior.

(iii) Texts include sequences of characters, words, or phrases. A text can be a sentence, a paragraph, or a document. It forms the backbone of many cognitive science studies on language processing, reading, and communication. A text can be treated as a special type of symbolic sequence. However, we set it apart from other symbolic sequences due to its following complexities:

- Hierarchical structure: Text tends to have a hierarchical structure; symbols (letters) form tokens (words), which form larger units (sentences, then paragraphs, and so on). This contrasts with many types of symbolic sequence data, where the sequence elements don't normally form larger cohesive groups.

- Grammar and syntax: Text is governed by complex grammatical and syntactic rules. These rules provide structure and impose dependencies between symbols that can extend beyond nearby symbols. Understanding this structure is crucial for the proper interpretation of text data.

- Semantics: Text ferries meaning or semantics, which often requires understanding the context. This rich semantic content often results in high dimensionality and complexity, requiring sophisticated models to capture the context effectively.

(iv) A trajectory refers to a path formed by a sequence of data points, which are represented by two-dimensional (2D) or three-dimensional (3D) coordinates. Common examples of trajectories include GPS paths, handwritten pen strokes captured by electronic devices, and eye movement paths obtained through eye-tracking devices.

(v) Attributed trajectory contains additional variables or attributes linked with each point of the trajectory, such as mouse-tracking data and scanpath. Although the broad concept of a scanpath can encompass the symbolic sequence of AOIs mentioned earlier, in the literature, the term "scanpath" typically refers to a sequence of fixations, which are usually represented as (x, y, d), where (x, y) is the 2D coordinates, d is the duration of the fixation. The duration is an attribute linked with the coordinates.

Among these data types, time series has often been viewed separately from sequence data due to its specialized focus on temporal data and unique methodological approaches, even though it conceptually fits within the broader landscape of sequence data (Fournier-Viger et al., 2017). A time series is an ordered list of numbers, while other sequences are often ordered lists of nominal values, such as categories, symbols, and positions. The foundational methodology for time series is the AutoRegressive Integrated Moving Average (ARIMA, Box et al., 2015) model, which has been extensively utilized to handle non-stationary time series typically observed in neural recordings, examples can be found in Leuthold et al. (2005), Liu et al. (2021), and Tagaris et al. (1997). However, other sequence data are either too large for ARIMA to handle efficiently or nominal-valued, by which is ARIMA inapplicable (Laxman and Sastry, 2006). Thus, this dissertation focuses on the data mining approaches for other sequence data types in cognitive science, namely the type (ii) to type (v) mentioned above.

1.2 Sequence Data Mining

Sequence Data Mining (SDM) involves the extraction of meaningful patterns and models from data that is ordered sequentially. Patterns, as noted by Hand (2007), reveal interesting regularities within the sequence, such as frequent sub-sequences, prototype sequences for a group, and clusters of sequences, while models provide comprehensive representations of the entire dataset, aiming to generalize the processes underlying the sequence data. For example, Hidden Markov Models (HMMs, Rabiner, 1989) have been utilized to model speech sequences, capturing the probabilistic transitions between different phonemes.

Early approaches in SDM primarily focused on discovering frequent sub-sequences, as seen in the pioneering work of the AprioriAll algorithm by Agrawal and Srikant (1995). This laid the groundwork for subsequent algorithms and techniques, gradually

expanding the scope of SDM to intersect with diverse disciplines such as machine learning, bioinformatics, and temporal data analysis. One notable evolution is the shift from mere pattern discovery to the development of comprehensive models encompassing the entire dataset.

In more recent times, the field of SDM has benefited significantly from advancements in machine learning methods, particularly with the burgeoning progress in Text Mining (Jiawei and Micheline, 2006) and the ascendancy of Deep Learning (LeCun et al., 2015). Text Mining, in particular, has contributed to the enrichment of SDM by offering sophisticated techniques for extracting meaningful patterns and insights from textual data. The application of Deep Learning models, such as Recurrent Neural networks (RNNs) and their variants Long Short-Term Memory networks (LSTMs), see Hochreiter and Schmidhuber (1997), has greatly expanded the capacity to capture intricate sequential patterns and dependencies within the data, thereby enhancing the capabilities of SDM in uncovering complex relationships and trends within sequential datasets.

Despite the advancements in SDM, many modern methods, such as machine learning techniques, have yet to be fully integrated into cognitive science applications, particularly in traditional tasks like diversity thinking. Moreover, in cognitive science, dealing with complex sequences, such as attributed sequences, poses a challenge, as conventional SDM methods may not be directly applicable. For instance, when analyzing eye-tracking data, relying solely on symbolic sequences composed of AOIs and employing traditional frequent sub-sequence mining methods may overlook critical information, such as spatial relationships between the AOIs. This highlights the growing need to adapt and develop sequence mining techniques that are specifically tailored to the unique demands of cognitive science research, catering to its diverse and complex data types.

This cumulative thesis consists of three articles that extend methods for mining sequence data in cognitive science, including the application of modern sequence data mining techniques to traditional tasks in cognitive science, as well as the development of new methods for handling the unique and complex sequences within this field. In Article (I), a machine learning-based pipeline is proposed for the automatic scoring of Diversity Thinking tasks. Various feature extraction methods and the performance of different machine learning algorithms are evaluated. Additionally, the generalizability of the pipeline across different datasets is validated. In Article (II), a new method to compute a representative scanpath is introduced to identify the pattern of the common view behavior of a group. In article (III), a novel Deep Learning algorithm is developed for extracting semantic information from handwritten trajectories.

In this thesis, the sequence data mining methods are categorized into Pattern Generating Methods and Model Generating Methods depending on their outcomes. An overview of the relevant methods in these two categories is provided in Chapter 2 and Chapter 3 respectively. At the end of these two chapters, a brief introduction

of the contributed publications to that chapter is given. Chapter 4 summarizes the important aspects regarding research motivation, the main methods, and results for each of the three articles. Chapter 5 contains discussions of the results, conclusions, and an outlook on some future research.

2 Pattern Generating Methods

This chapter will introduce Pattern Generating Methods related to sequence data in cognitive science, providing an overview and outlining the contributions of my publications to this area of research. The three primary methods are frequent pattern mining, clustering, and centroid identification, each corresponding to important modes of pattern recognition.

2.1 Frequent Pattern Mining

Frequent pattern mining for sequences involves identifying "interesting" subsequences within a set of sequences. The interesting aspect of a subsequence is evaluated based on various criteria, including its frequency of occurrence, length, and associated value or profit. For example, in the domain of text mining, subsequences of frequently used phrases like "artificial intelligence" or "machine learning" in technical documents could be of interest; In eye-tracking studies, an interesting subsequence might be a pattern of fixations and saccades related to the cognitive process of reading text; In problem-solving tasks, for instance, in the game of chess, an interesting subsequence could be a frequently used series of moves (a sequence of chess notation) that leads to a strategical advantage. Key algorithms to identify frequent subsequence include:

- AprioriAll algorithm (Agrawal and Srikant, 1995): This algorithm extends the Apriori (Agrawal, Srikant, et al., 1994) principle to sequential data, identifying frequent itemsets that appear in a given order. It operates in two primary steps: first identifies the items that meet the minimum support threshold, then combines these items to form larger frequent itemsets. This algorithm is especially suitable for mining symbolic sequences, where the total number of unique symbols is manageable and the itemsets' size is usually small. However, the AprioriAll algorithm can struggle with large datasets because it needs to scan the database multiple times.
- Generalized sequential pattern(GSP, Hirate and Yamana (2006)): It is also an extension of the Apriori algorithm, which iteratively identifies frequent subsequences by generating candidate sequences and then counting their occurrences. This algorithm addresses scalability better than AprioriAll but still suffers from the cost of multiple database scans.

- Prefix-projected Sequential Pattern Mining (PrefixSpan, Han et al., 2001): The PrefixSpan algorithm adopts a different strategy from Apriori-based algorithms. It is a pattern-growth approach that iteratively projects sequence databases based on prefix patterns to discover frequent subsequences efficiently. Each such projection results in a smaller database that retains the essential information about frequent sequences. Because PrefixSpan eliminates the need to generate candidate sequences and performs only a single database scan, it can achieve significantly higher efficiency than Apriori-based algorithms.

In cognitive science, frequent pattern mining techniques are often used for text and symbolic sequences. For text, GSP and PrefixSpan are used to discover frequent phrases across different documents. For a detailed introduction to these algorithms and their application in text data, please refer to Ożdżyński and Zakrzewska (2017). For the symbolic sequence in cognitive science, sequential pattern mining techniques are applied for sequences of AOIs in eye-tracking studies (see Hassani et al. (2015), Matsuda and Takeuchi (2014), McGuire and Chakraborty (2016), and Yu et al. (2023)) and behavior sequences in problem-solving tasks to discover common sequences (see Gadár and Abonyi (2019), Halioui et al. (2015), He et al. (2021), and Lien et al. (2020)).

2.2 Sequence Clustering

Sequence clustering aims to group sequences into clusters based on their similarity (or distance) using clustering algorithms. The terms similarity and distance will be used interchangeably in this thesis, since they can be easily converted into one another. A high similarity corresponds to a small distance, and vice versa.

Two key aspects of sequence clustering are defining an appropriate distance measure for the sequences and selecting a suitable clustering algorithm. Next, I will provide an overview of the relevant distance measures and clustering algorithms.

2.2.1 Distance Measures

Since sequences in cognitive science can vary significantly in their nature and structure, different types of sequences require different distance measures to adequately capture their underlying patterns. The choice of an appropriate distance measure is crucial to the effectiveness of the clustering process and the interpretation of the resulting clusters. In the following subsection, we introduce distance measures suitable for the four types of sequences in cognitive science separately.

Distance Measures for Symbolic Sequences

The Levenshtein distance (Levenshtein et al., 1966), also known as the edit distance, is a commonly used distance measure for symbolic sequences, which quantifies the minimal operations (insertions, deletions, substitutions) required to transform one sequence into another.

Furthermore, the Longest Common Subsequence (LCS, Hirschberg (1977)) finds the length of the longest subsequence present in both sequences.

Additionally, sequence alignment techniques are used to align sequences optimally, considering matches, mismatches, and gaps (delete and insert) operations. A commonly used alignment technique is the Needleman-Wunsch (Needleman and Wunsch, 1970) algorithm, which is originally designed for DNA and RNA sequences and is now also often used for behavior sequences in cognitive science.

Distance Measures for Text

While text can also be viewed as a symbolic sequence, the distance can then be calculated using the methods above. However, considering the special semantic and context properties of text, we focus here on the vector-represenation-based distance measures. Two common vector representations of Text are Bag of Words (BoW, Salton et al., 1975; Sebastiani, 2002) and Word Embeddings (Mikolov et al., 2013; Pennington et al., 2014). In BoW, a text is treated as a bag or set of words, ignoring grammar and word order but keeping track of frequency. For example, consider a simple vocabulary {“apple”, “banana”, “orange”}. The document “apple banana apple” would be represented as the vector $(2, 1, 0)$, indicating “apple” appears twice, “banana” once, and “orange” not at all. This approach captures the frequency of words but loses information about their order and context.

Word Embedding is a more advanced technique where words are mapped to vectors that capture semantic similarity based on the context in which they appear. Examples of such methods include Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Word2Vec is a Neural Network-based approach that consists of two models: Continuous Bag of Words (CBOW) and Skip-Gram.

CBOW predicts a target word given its surrounding context words. For example, given the context “the cat on the”, CBOW predicts the word “sat”. Skip-Gram does the opposite. It predicts the surrounding context words given a target word. For example, given “cat”, it predicts words like “the”, “sat”, and “on”.

These models learn word vectors that words appearing in similar contexts in the training data are close to each other in the vector space, effectively capturing semantic similarity. For instance, vectors for synonyms or related concepts (e.g., “king” and “queen”) will be found close to each other in the embedding space.

GloVe, which is short for Global Vectors for Word Representation, is another popular Word Embedding technique that leverages statistical co-occurrence information from a corpus. Unlike Word2Vec, which is processed locally within small context windows, GloVe builds word vectors based on the global statistical information of word co-occurrences. It constructs a huge co-occurrence matrix where each element represents how often words co-occur within a specified context window in a text corpus. By factoring in the co-occurrence matrix, GloVe learns word vectors such that their dot product is proportional to the logarithm of the probability of co-occurrence. This ensures that the geometric distance in the vector space reflects meaningful word relationships.

While BoW converts the entire text (sequence of words) into one high-dimensional vector, Word Embeddings convert individual words into a sequence of low-dimensional vectors, say $S = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$, where $\mathbf{x}_t = (x_{t1}, \cdots, x_{tp})$, for $t \in \{1, \cdots, n\}$, is the the embedding vector for a single word. When two texts have different numbers of words, each of them has to be composited into a single vector with the same length, to calculate their distance. Commonly used composition methods are additive composition and zero padding. Denote $\mathbf{x}$ resulting vector of the composition, the additive composition is to compute the element-wise sum of all word embedding vectors with

$$\mathbf{x} = \sum_{t=1}^{n} \mathbf{x}_t. \tag{2.1}$$

The zero padding method concatenates all the word vectors into a long vector and pads shorter sequences to a fixed length with zeros with

$$\mathbf{x} = (x_{11}, \cdots, x_{1p}, \cdots, x_{n1}, \cdots, x_{np}, 0, \cdots, 0). \tag{2.2}$$

The additive composition method is straightforward to implement and computationally efficient. However, it ignores the order of words. Important positional and contextual information is lost in the summation, which can be crucial for understanding the meaning of the text. On the other hand, zero padding preserves the order of words but increases the dimensionality of the final vector, especially for longer texts. What's more, padding sequences to a fixed length introduces redundancy and noise into the data. For shorter sequences, a large portion of the final vector might be padding, which does not contribute meaningful information and can dilute the representation. In my PhD research, Article (III) compares the performance of these two methods in the context of Diverse Thinking tasks.

After representing the texts with vectors, two popular distance measures, the Euclidean distance, and the Cosine distance can be used to compute the distance between texts (for other distance measures please refer to Wang and Dong (2020). The Euclidean distance of two text representation $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ is computed with

$$||\mathbf{x}^{(1)} - \mathbf{x}^{(2)}||, \tag{2.3}$$

where $||\cdot||$ represent the Euclidean norms of the vector. Euclidean distance refers to how two texts are close lexically (the words used). Thus, it is usually used to measure the amount of common words of two texts, such as in the plagiarism detection tasks.

The cosine distance is given by

$$1 - \frac{\mathbf{x}^{(1)} \cdot \mathbf{x}^{(2)}}{||\mathbf{x}^{(1)}||||\mathbf{x}^{(2)}||}, \tag{2.4}$$

which measures the cosine of the angle between two vectors and ranges from -1 to 1. Compared to the Euclidean distance, Cosine distance focuses more on the orientation or semantic similarity of text vectors rather than their magnitude. For example, if the word "football" occurs more in Text 1 than in Text 2, it may have a smaller Euclidean distance to the topic "sports" when the term frequency is considered. However, Text 1 might be just longer than Text 2, and this difference in length can disproportionately affect the Euclidean distance calculation. In contrast, Cosine distance normalizes for the length of the vectors, meaning it primarily measures the degree to which the texts share the same topics, independent of their length.

Distance Measures for Trajectories

Commonly used distance measures for trajectories are the Euclidean distance, the Dynamic Time Warping (DTW, Sakoe and Chiba (1978) distance, and the Fréchet Distance (Alt and Godau, 1995).

The Euclidean distance (here is different from the Euclidean distance for text mentioned above) is a basic metric for comparing trajectories, which calculates the sum of the distances of the corresponding points of two trajectories of the same length. Given two trajectories $S^{(1)} = \{\mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_n^{(2)}\}$ and $S^{(2)} = \{\mathbf{x}_1^{(2)}, \cdots, \mathbf{x}_n^{(2)}\}$ with the same length n, where $\mathbf{x}_t^{(i)}$ is the 2D or 3D coordinates of t-th point on i-th trajectory, the Euclidean distance of them is defined by

$$\sum_{t=1}^{n} ||\mathbf{x}_t^{(1)} - \mathbf{x}_t^{(2)}||. \tag{2.5}$$

DTW uses dynamic programming to find an optimal alignment between the sequences by warping (stretching or compressing) the time axis, which addresses the problem of comparing two sequences that may be out of phase. For example, in speech recognition or activity recognition, the same word or movement might be spoken or performed at different speeds. Denote $\mathbf{D} \in \mathbb{R}^{n \times m}$ the distance matrix of the two trajectories $S^{(1)} = \{\mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_n^{(1)}\}$ and $S^{(2)} = \{\mathbf{x}_1^{(2)}, \cdots, \mathbf{x}_m^{(2)}\}$, where

$$\mathbf{D}_{st} = ||\mathbf{x}_s^{(1)} - \mathbf{x}_t^{(2)}||, \tag{2.6}$$

for $s \in \{1, \cdots, n\}$ and $t \in \{1, \cdots, m\}$, DTW aims to find a path from $\mathbf{D}_{11}$ to $\mathbf{D}_{nm}$ such that each point of both sequences is matched to at least one point of the other sequence and the cumulative distance on the path is minimized. The path on the distance matrix is a sequence of indices $\{(s_1, t_1), (s_2, t_2), \ldots, (s_K, t_K)\}$, where each pair (s_k, t_k) corresponds to matching $\mathbf{x}^{(1)}_{s_k}$ with $\mathbf{x}^{(2)}_{t_k}$. This path must satisfy the following requirements:

- The path must start at $\mathbf{D}_{11}$ and end at $\mathbf{D}_{nm}$.
- The steps in the path must be adjacent in the matrix. This means that for any two consecutive points (s_k, t_k) and (s_{k+1}, t_{k+1}) in the path, the indices should satisfy: $(s_{k+1} - s_k, t_{k+1} - t_k) \in \{(1, 0), (0, 1), (1, 1)\}$. This ensures that the path progresses either right, up, or diagonally (we assume that $\mathbf{D}_{11}$ is in the lower-left corner of the matrix).
- The indices must increase monotonically, ensuring that time does not move backward in either sequence. This means: $s_{k+1} \geq s_k \quad \text{and} \quad t_{k+1} \geq t_k$.

The objective of DTW is to find a warping path that minimizes the total cumulative distance:

$$\sum_{k=1}^{K} \mathbf{D}_{s_k t_k}. \tag{2.7}$$

By solving this optimization problem, DTW provides an alignment that can handle variations in speed and timing between the sequences.

Fréchet distance, also known as "dog-walking distance", measures how a leash (representing the distance between two trajectory points) varies as two entities (a person and a dog, representing points on two trajectories) walk along their respective paths. The Fréchet distance is the minimum necessary leash length required for walking the two trajectories synchronously from start to end, allowing for different speeds. For a detailed description of the calculation Fréchet distance, please refer to Alt and Godau (1995).

All the computation of these three trajectory distances can be initiated from the distance matrix $\mathbf{D}$ of two trajectories. Figure 2.1 shows the cases for Euclidean distance and DTW distance. The left-hand side shows the different ways the matching of two trajectories. Trajectories are represented with solid lines in different colors. Dashed lines are used to connect the points of the matches. The right-hand side shows the distance matrix. Euclidean distance of the two trajectories is the sum of the elements on the diagonal of the distance matrix. It is equivalent to matching two points from different trajectories with the same index and then summing the distances of the matched pairs. On the other hand, DTW searches a path from $\mathbf{D}_{11}$ to $\mathbf{D}_{nn}$ on the distance matrix, so that of elements on the path is minimized. Then the points of the two trajectories are matched according to the corresponding index of the path, and the DTW distance is also the sum of the distances of the matched points.

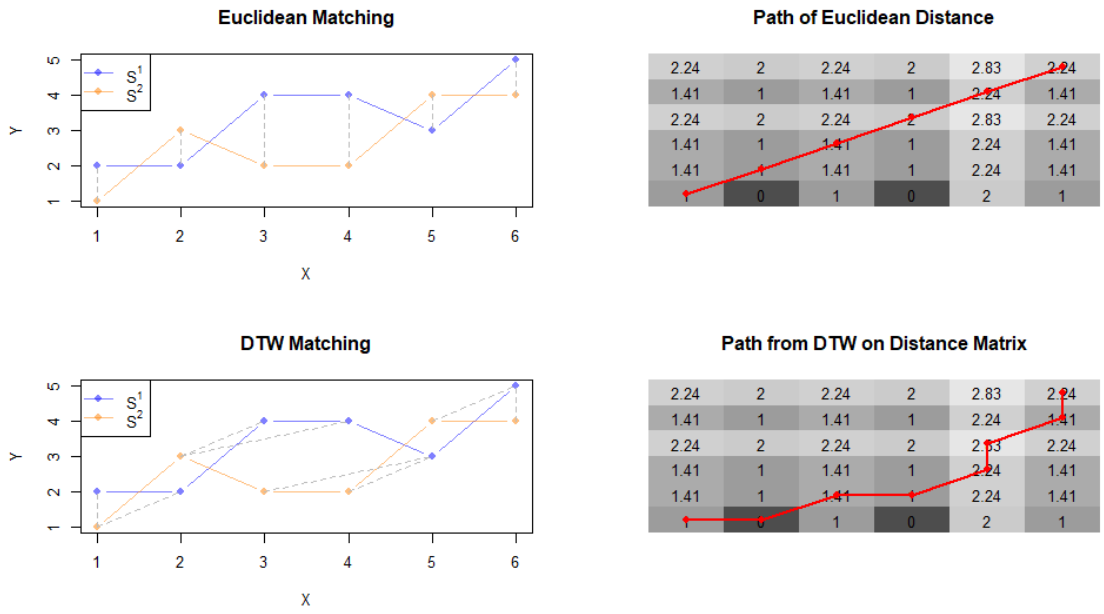

Figure 2.1: Computation of Euclidean distance and DTW distance of two Trajectories starting with the distance matrix.

Fréchet distance can also be computed from the distance matrix of two Trajectories. It seeks a path that minimizes the maximum distance encountered along the path. The difference with DTW is that it used the maximum instead of the sum on the path as the distance.

Theoretically, the distance matrix between two trajectories contains all the information about the distances between the two trajectories. Taking inspiration from this, in Article (III), we extract features from the distance matrix and employ a machine learning model to compute the distance between the trajectories. These extracted features then act as the input for a machine learning model. By training this model on a set of trajectory pairs along with their corresponding labels (0 or 1 indicating whether a pair of trajectories belong to the same class or not), the model can learn how to predict the distance between two new trajectories based only on features extracted from their distance matrix. As such, this method transforms the problem of calculating trajectory distances into a machine learning problem, allowing us to leverage the power of machine learning algorithms to make this calculation more efficient and potentially more accurate.

Distance Measures for Attributed Trajectories

For attributed trajectories we focus on the scanpath in eye-tracking studies. There are distance measures available for other types of attributed trajectories. For example,

the Area Under Curve (AUC) and Maximum Deviation (MD) for the mouse-tracking sequence, see Maldonado et al. (2019). However, these measures may not be as specialized or extensive as those developed for scanpaths, given that eye-tracking data has been the subject of intensive investigation for a longer period.

Three commonly used distance measurements for scanpath are Scansim (Von der Malsburg and Vasishth, 2011), ScanMatch (Cristino et al., 2010), and MultiMatch (Jarodzka et al., 2010). A brief description of them is as follows.

Scansim calculates scanpath dissimilarity based on the Levenshtein distance. It takes into account the variations in acuity across the visual field, penalizing distances between fixations by considering the high acuity in the fovea and the drop in resolution towards the periphery. Unlike other methods, ScanSim focuses directly on pixel coordinates of fixations, providing a nuanced assessment of dissimilarity within scanpaths.

In contrast, ScanMatch transforms scanpaths into strings by segmenting the stimulus into grids and labeling the grids with letters. Each letter repetition is based on the duration of the fixation on the corresponding grid cells. The resulting string sequence incorporates spatial location, sequential information, and temporal duration of the fixations on the scanpath. ScanMatch aligns and compares these strings using the Needleman-Wunsch algorithm, providing a comprehensive overall similarity score. It leverages a substitution matrix to encode information about the relationship between each grid cell, enhancing the method's sensitivity to spatial and temporal characteristics within the scanpaths.

MultiMatch represents two scanpaths as sequences of saccade vectors and compares them from five perspectives: vector, direction, length, position, and duration similarities. The measure simplifies scanpaths based on saccade angles and amplitudes before evaluating their similarities, also considering spatially close fixations. MultiMatch involves a more detailed analysis of scanpath characteristics, providing five distinct scores for the five comparison aspects. This provides a nuanced and multifaceted assessment of similarities within the scanpaths, enabling a more granular understanding of their differences across various dimensions.

These distance measures offer distinct strategies for evaluating similarities for scanpaths, each tailored to capture specific nuances and characteristics of eye movement. All three measurements utilize both types of information, the coordinates of the attributed trajectory and the attributes corresponding to each coordinate point.

In addition to the three distance measures above, some studies utilize only part of the information from scanpaths, such as treating the scanpath only as a trajectory, or only as a symbolic sequence to compute their distances. For detailed descriptions and comparisons of these distance measurements please refer to Fahimi and Bruce (2021).

In my PhD research, a pattern recognition method for scanpath is developed, which can adapt to different distance measures above, see Article (II).

2.2.2 Clustering Methods

Popular clustering approaches include k-Centroids clustering, Hierarchical clustering (Johnson, 1967), and Density-Based Spatial Clustering of Applications with Noise (DBSCAN, Ester et al., 1996), all of which have been applied to cluster sequence data in cognitive science.

k-Centroids represents a class of clustering algorithms, that aims to partition the data into k clusters by iteratively reassigning data points to the nearest cluster centroids and recalculating the centroids until convergence is achieved. These clustering algorithms find centroids that optimize a certain criteria by considering the distance between the points within a cluster and the distance between the clusters. Each algorithm differs in the definition of cluster centroid and the distance measures. For example, k-Means (MacQueen et al., 1967) and k-Median (Shmoys et al., 1997) are two specific types of k-Centroids clustering algorithms. k-Means uses the mean vector as the centroid and k-Median uses the median as the cluster centroid, where the median is computed in each single dimension. While k-Means uses Euclidean distance as the distance measure, k-Median uses the Manhattan distance ($L1$ distance). Both methods are used in the cases, where the sequences are represented as vectors, e.g. Word2Vec for text. However, for other sequences in cognitive science, it is a challenge to define the centroid for a cluster. Furthermore, a key challenge lies in determining the value of k, which involves a comprehensive consideration of domain knowledge, practical application requirements, and algorithm performance.

In contrast to k-Centroids clustering, Hierarchical clustering does not require the predefined number of clusters or the computation of centroids. Instead, it relies on the definition of distance measure for the sequences and seeks to build a hierarchy of clusters. The hierarchy structure is formed by either iteratively merging smaller clusters into larger ones (agglomerative) or by subdividing larger clusters into smaller ones (divisive). In agglomerative hierarchical clustering, each data point initially represents its own cluster. At each step, the two closest clusters are merged. This process continues until all data points belong to a single cluster, resulting in a hierarchy of clusters. In divisive hierarchical clustering, all data points start in a single cluster. At each step, the algorithm progressively divides the clusters into smaller ones until each data point is in its own cluster.

The advantage of Hierarchical clustering lies in its ability to create a tree-like visual representation (dendrogram) that displays the order in which the clusters were merged. This can provide valuable insights into the relationships between the data points and the natural groupings within the data. However, Hierarchical clustering can be computationally expensive, especially when dealing with large data sets, as

the memory and time complexity grows quadratically with the number of data points. Furthermore, the determination of the optimal number of clusters can be subjective when using the dendrogram.

DBSCAN is a another popular clustering algorithm. It works by grouping data points that are closely packed and have a minimum number of neighboring points within a specified radius. The algorithm has two parameters: ϵ, which defines the radius within which to search for neighboring points, and "minPoints", which specifies the minimum number of points within the radius for a point to be considered a core point. Given the two parameters ϵ and "minPoints", DBSCAN proceeds as follows:

- It starts by randomly selecting a data point that has not been visited.
- For this point, it identifies all its neighboring points within the specified radius ϵ.
- If the number of points in the ϵ-neighborhood is greater than or equal to "minPoints", the point is labeled as a core point, and all of its neighbors are added to the cluster.

Points that are not core points but are reachable from a core point (i.e., they are within the radius ϵ of a core point) are considered border points and are assigned to the cluster of the core point. Points that are neither core points nor reachable from a core point are considered noise .

One advantage of DBSCAN is its ability to identify clusters of arbitrary shapes and handle outliers effectively. Additionally, it does not require the user to specify the number of clusters beforehand. However, DBSCAN requires the definition of two hyperparameters, which can be non-trivial to set optimally. Choosing appropriate values for these hyperparameters involves understanding the density distribution within the dataset and may require trial and error or domain knowledge.

2.3 Group Centroid Identification

Calculating a centroid for a group of sequences is an important task in sequence data mining, primarily aimed at describing, summarizing, or visualizing a group of sequences. This is akin to using a single number, such as the mean or median, to describe a set of numbers in statistics. However, for sequences, calculating an average or median sequence becomes challenging. A commonly used method is to calculate the medoid to respresent a group of sequences. The medoid sequence is the sequence from the group with the smallest average or total distance to all others in the group. For example, in the eye tracking studies, the medoid scanpath are used as the prototypical view pattern for a group, see Paape et al. (2022), Parshina et al. (2022), and Von der Malsburg and Vasishth (2013).

Instead of identifying an existing representative sequence from the sample, some methods generate an artificial sequence as the centroid of the group. For example, consensus sequences are widely used in molecular biology to summarize and visualize sequence datasets. The consensus sequence represents the most prevalent nucleotide or amino acid at each position in an alignment of DNA or RNA sequences. This method has also been applied to the symbolic sequence in cognitive science, see Hinsz (1990), Mohammed (2001), and Mohammed and Ringseis (2001). Additionally, in the field of cognitive science, there are also specialized methods, such as the Dynamic Time Warping Barycenter (Li and Chen, 2018) method for scanpath. This method generates an average sequence by iteratively adjusting a candidate sequence to minimize the DTW distance to all sequences in the set.

2.4 Contributed Publications

My contributed publications to this Chapter are Article (II) and Article (III).

In Article (II), a new method to compute the group centroid is developed for one of the intensively studied attributed trajectories in cognitive science, namely the scanpath. Experiments show that the centroids detected by this new method are more representative than the ones detected by existing methods. What's more, a software package is developed for calculation and visualization purposes. A summary of Article (II) will be provided in Section 4.2.

In Article (III), a model is developed and learned to compute the distance for trajectories, instead of using a predefined one, such as the DTW distance. The computed distance from the model is then used in an algorithm to classify the trajectories. Experiments show that the accuracy of classification is improved by using the distance calculated by this model. A summary of Article (III) will be provided in Section 4.3.

3 Model Generating Methods

Model generation methods utilize techniques from both supervised and unsupervised learning in the field of machine learning. The primary objective is to extract a model that can characterize and forecast sequences by analyzing the given sequential data.

3.1 Supervised Learning Models

In the case of supervised learning, a model is trained using a known dataset (training data), which includes input data and corresponding responses, to make predictions on unseen or future data. A supervised model can be formulated as:

$$y = f(\mathbf{x}), \tag{3.1}$$

where $\mathbf{x} \in \mathcal{X}$ represents the input vector, $y \in \mathcal{Y}$ is the output or target variable that the model aims to predict, and $f : \mathcal{X} \to \mathcal{Y}$ refers to the function or model that captures the relationship between $\mathbf{x}$ and y. If $\mathcal{Y} \subseteq \mathbb{R}$, then it constitutes a regression task, where the goal is to predict a continuous value. On the other hand, if $\mathcal{Y}$ is a finite set, the task becomes a classification problem, aiming to assign input data into predefined classes. The training of a supervised model involves the following steps:

- Training data collection and feature extraction: This step is to construct the examples with associated true outcomes, which is denoted as $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$. For instance, it may include trajectories of handwritten digits and their true labels. For the application of supervised models to sequence data, the primary challenge lies in constructing the input $\mathbf{x}$ in a manner that can be digested and processed effectively by these machine learning algorithms. There are two primary methods for constructing such an input vector $\mathbf{x}$. The first method is to extract features from the sequence manually. For example, for text data, features could include the number of words or the frequency of certain key terms. For trajectory data, manual features might include the total length of the trajectory, the average speed, or the change in direction. These manually selected features help to distill meaningful information from the raw sequences, making them easier for machine learning models to interpret. The second method is automatic feature extraction. Automatic feature extraction can

be performed as a separate step, such as using word2vec to transform words into vectors. Alternatively, automatic feature extraction can be integrated into the model training process. In this approach, models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) are designed to learn and extract relevant features directly from the raw sequence data during training. CNNs can automatically identify and learn spatial hierarchies in image data, while RNNs can effectively learn temporal dependencies in sequence data such as text or trajectories. Combining both manual and automatic feature extraction methods can also be an option. Manual features provide domain-specific insights, while automatic methods can uncover hidden patterns within the data, thereby improving the overall performance of the sequence modeling task.

- Model selection: select the model that fits best with the nature of the data and the problem at hand, such as linear regression, decision tree, the ensembles of decision trees (e.g., Random Forest and XGBoost), Support Vector Machines (SVM), and various Neural Network models.
- Model training: This step involves iteratively adjusting the model parameters to minimize the discrepancy between the model's predictions and the true outcomes of the training data. This discrepancy is quantified by a loss function $L(y, f(\mathbf{x}))$, and the goal of training is to find the parameters that minimize this loss.
- Validation and hyperparameter tuning: Apply the model to a validation set (data that the model has not been trained on) to evaluate the performance. Tune hyperparameters and make necessary adjustments to the model to optimize generalization.
- Evaluation: Finally, the model's performance will be tested using a test dataset that has not been seen by the model during training or validation.
- Prediction: Once trained, validated, and tested, the model can then be used to predict the outcomes of new, unseen data.

Some commonly used supervised learning models are introduced below.

3.1.1 k Nearest Neighbors

k Nearest Neighbors (kNN, Mucherino et al., 2009) algorithm is a simple and intuitive non-parametric classification and regression method. In kNN, the prediction for a new data point is made based on the majority class of its k nearest neighbors in the dataset. For regression, the prediction is based on the average of the target values of the k nearest neighbors. The algorithm is straightforward to comprehend, making it a good candidate for initial exploration or baseline comparison. However,

The algorithm needs to compute distances for each new data point, making it computationally expensive, especially for large datasets. What's more, the choice of the parameter k may influences the performance, and selecting the optimal k can be challenging. The optimal value of k may vary for different datasets.

3.1.2 Support Vector Machine

Support Vector Machine (SVM, Boser et al., 1992) is another supervised learning model used for classification and regression. In classification cases, the primary goal of SVM is to find the optimal hyperplane that categorizes data points into different classes while maximizing the margin between the classes. Formally, given a set of training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, where $y^{(i)} \in \{-1, 1\}$ represents the class label, the goal is to find the optimal hyperplane defined by:

$$\mathbf{w} \cdot \mathbf{x} - \mathbf{b} = 0, \tag{3.2}$$

where $\mathbf{w}$ is the weight vector, $\mathbf{x}$ is the input vector, and $\mathbf{b}$ is the bias term.

The optimization problem for finding the optimal hyperplane can be formulated as:

$$\min_{\mathbf{w},\mathbf{b},\xi} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi^{(i)}, \tag{3.3}$$

subject to the constraints:

$$y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} - \mathbf{b}) \geq 1, \tag{3.4}$$

and

$$\xi^{(i)} \geq 0, \tag{3.5}$$

for $i = 1, 2, \ldots, N$. Here, $||\mathbf{w}||$ denotes the Euclidean norm of the weight vector $\mathbf{w}$. Thus, the above optimization problem aims to maximize the margin while minimizing the norm of the weight vector. $C\sum_{i=1}^{N} \xi^{(i)}$ is a regularization term used in the cases that the data is not linearly separable. $\xi^{(i)}$ is called slack variable of "soft-margin" for each sample to tolerate misclassifications.

SVM can also handle non-linear decision boundaries by using the "kernel trick," where the input features are mapped into a higher-dimensional space, allowing for non-linear classification. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid kernels. For details please refer to Boser et al. (1992).

In regression cases, SVMs are used to find the best-fit hyperplane to predict continuous outcomes. The objective of SVM regression is to find a function that best fits the data while minimizing the error, known as the ε-insensitive loss function.

For a set of training data $\mathcal{D}$, SVM regression aims to find the optimal hyperplane defined by:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - \mathbf{b}. \tag{3.6}$$

The optimization problem for SVM regression is formulated as:

$$\min_{\mathbf{w},\mathbf{b},\xi,\xi^*} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n}(\xi^{(i)} + \xi^{*(i)}), \tag{3.7}$$

subject to the constraints:

$$y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} - b \leq \varepsilon + \xi^{(i)}, \tag{3.8}$$

$$\mathbf{w} \cdot \mathbf{x}^{(i)} + \mathbf{b} - y^{(i)} \leq \varepsilon + \xi^{*(i)}, \tag{3.9}$$

and

$$\xi^{(i)}, \xi^{*(i)} \geq 0, \tag{3.10}$$

for $i \in \{1, 2, \cdots, N\}$, where ε represents the margin of tolerance, allowing deviations within this margin to be ignored from the loss function. $\xi^{(i)}$) represents the deviation of the predicted value below the actual target value, while $\xi^{*(i)}$ denotes the deviation of the predicted value above the actual target value.

SVM is less sensitive to noisy data compared to kNN, as it focuses on defining a maximal margin boundary, reducing the impact of individual outliers. However, SVM can be computationally expensive to train the model, since it has a large number of parameters.

3.1.3 Decision Tree and Decision Tree-Based Ensembles

Decision Tree is another type of widely used non-parametric supervised learning model, which constructs a tree-like structure by recursively splitting the dataset based on the most significant feature at each node. There are different Decision Tree algorithms. Their difference lies in the rules of the feature selection for the splitting at each node. For example, C4.5 (Quinlan, 2014) uses Information Gain for decision tree generation, and Classification and Regression Tree (CART, Breiman, 2017) uses Gini Impurity for classification and Mean Squared Error for regression. This thesis focuses on regression trees from the CART class since it has been applied in Article (I). Given a set of training data $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with $\mathbf{x}^{(i)} = (x_1^{(i)}, \cdots, x_p^{(i)})$ representing the feature vector of the i-th observation. The fundamental idea is to iteratively partition the feature space into separate non-overlapping regions, say $(R_1, ..., R_J)$, until a specific termination condition is met, such as the maximum depth of the tree. Subsequently, for each of these distinct regions, also known as terminal

nodes, an individual estimation of the target variable is carried out by calculating

$$\hat{c_j} = \frac{1}{N_j} \sum_{x_i \in R_j} y_i, \tag{3.11}$$

where N_j corresponds to the number of data points in region R_j.

Decision trees are particularly prone to overfitting the training data when the data is noisy or contains irrelevant features. To address this issue, ensemble techniques, such as Bagging and Boosting, have been developed. Bagged ensembles, such as Random Forest (Random and Leo, 2001) and Extra Trees (Geurts et al., 2006), train multiple trees on random subsets of the features and combine their predictions to improve overall performance. The ensemble prediction for classification tasks is determined by majority voting, while for regression tasks, it is based on averaging the individual tree predictions.

Despite Bagging, further improvements in learning power and performance are achievable with Boosting methods. Boosting Trees are iterative ensemble algorithms that aim to reduce model error progressively. One of the most effective boosting algorithms is XGBoost (Chen and Guestrin, 2016), which employs an iterative optimization method to enhance a model's generalization ability. The process involves initializing a simple Decision Tree model and then iteratively adding new trees, where each new tree is trained to minimize the residual errors of the combined previous trees. XGBoost holds a prominent position in the machine learning field, especially before the widespread adoption of Deep Learning models. It gained recognition for its exceptional performance in structured (tabular) data analysis, where it outperformed many traditional machine learning algorithms, which is also validated in the earlier publication of my PhD research, see Article (I).

3.1.4 Neural Networks Models

As Deep Learning continues to find widespread application across various domains, including cognitive science, I shifted the focus of my PhD research in the last year towards this transformative technology. The terms "Neural Networks" and "Deep Learning" are often used interchangeably in literature. However, Deep Learning specifically refers to Neural Network models that are deeper, meaning they contain a large number of hidden layers. A Neural Network typically consists of layers of interconnected nodes or neurons, where each connection has an associated weight that is adjusted during training. The network learns by minimizing a loss function and adjusting these weights to improve its performance on a given task.

The fundamental structure of Neural Networks is the Multilayer Perceptrons (MLPs, Rumelhart et al., 1986), which consist of multiple layers of interconnected neurons. The typical architecture of an MLP includes an input layer, one or more hidden layers,

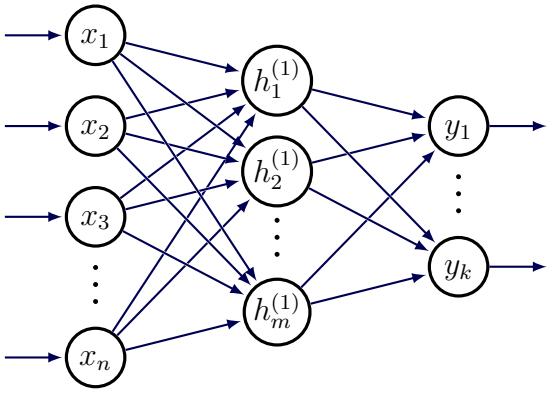

Figure 3.1: An example of MLP with a single hidden layer.

and an output layer. Figure 3.1) shows a simple MLP with only one hidden layer. The structure of an MLP can be described as follows. The input layer receives the data and forwards it to the hidden layers, where each neuron calculates a weighted sum of its inputs, adds a bias, and applies an activation function to determine its output. Common activation functions include the sigmoid, tanh, and ReLU (Rectified Linear Unit) functions. These activation functions introduce non-linear transformations to the data, which are crucial for enabling the model to capture and represent complex non-linear relationships within the data. For a more detailed overview of these activation functions and their properties, please refer to Rasamoelina et al. (2020).

The output layer produces the final prediction of the network. The forward computation in an MLP is expressed mathematically as:

$$\mathbf{h}^{(l)} = \sigma(\mathbf{W}^{(l)}\mathbf{h}^{(l-1)} + \mathbf{b}^{(l)}), \tag{3.12}$$

where $\mathbf{h}^{(l)}$ denotes the output of the l-th layer, $\mathbf{W}^{(l)}$ represents the weight matrix, and $\mathbf{b}^{(l)}$ the bias vector, and σ the activation function.

Training the MLP involves adjusting the weights $\mathbf{W}^{(l)}$ and biases $\mathbf{b}^{(l)}$ using the backpropagation algorithm. The key steps in backpropagation include:

- Computing the gradients of the loss function with respect to each weight by applying the chain rule of calculus.
- Updating the weights in the direction that reduces the loss function, typically through a method like gradient descent:

$$\mathbf{W}^{(l)} := \mathbf{W}^{(l)} - \eta \frac{\partial L}{\partial \mathbf{W}^{(l)}}, \tag{3.13}$$

  where η is the learning rate, and L is the loss function.

Although MLPs are versatile and effective for many tasks, they do not scale well to grid-like data (such as image) or sequence data due to their fully connected

architecture. To address these limitations, specialized Neural Network architectures have been developed to handle the nuances of different types of data. Convolutional Neural Networks (CNNs, LeCun et al., 1998) are particularly efficient for grid-like data structures, such as images and videos, by preserving spatial hierarchies. Similarly, Recurrent Neural Networks (RNNs, Elman, 1990) and their advanced variants like Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber, 1997) networks are well-suited for sequence data, such as time series and natural language, by maintaining temporal dependencies.

CNNs

A CNN consists of several types of layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers apply convolution operations to the input, using filters or kernels that slide over the input data to produce feature maps. CNNs are designed for processing grid-like data, such as matrices. Given an input matrix $\mathbf{D} \in \mathbb{R}^{m \times n}$ and a kernel with learnable parameters (also known as a filter) $\mathbf{K} \in \mathbb{R}^{u \times v}$. The element at s-th row and t-th column of the resulting matrix $\mathbf{C}$ of the convolution operation between $\mathbf{D}$ and the kernel $\mathbf{K}$ is caculated with

$$\mathbf{C}_{st} = \sum_{\mu=0}^{u-1} \sum_{\nu=0}^{v-1} \mathbf{D}_{s+\mu,t+\nu} \mathbf{K}_{\mu+1,\nu+1}, \tag{3.14}$$

for $s \in \{1, \cdots, m-u+1\}$ and $t \in \{1, \cdots, n-v+1\}$.

Pooling layers follow convolutional layers to reduce the spatial dimensions of the feature maps (the resulting matrix $\mathbf{C}$), typically using operations like max pooling or average pooling, which help reduce computational complexity and improve robustness to spatial transformations. The outputs of the pooling layers are then fed into an MLP (fully connected layers) to get the model predictions.

Although CNN is used for grid-like data and not for sequence data, we have transformed a pair of sequences into grid-like data in (III) by using their distance matrix to compute the distances of paired sequences, allowing the CNN to be applied.

RNNs

Recurrent Neural Networks (RNNs) are a class of neural networks designed to process sequential data. Unlike CNNs, RNNs can maintain a "memory" of previous inputs via their internal state, allowing them to capture temporal dependencies within the data. This makes RNNs exceptionally powerful for tasks where the order of inputs matters, such as time series prediction, natural language processing, and speech recognition.

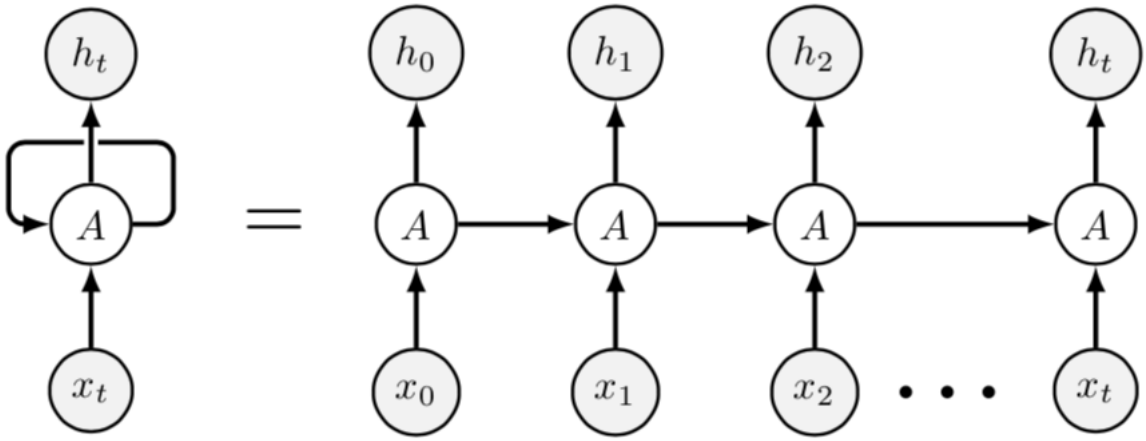

Figure 3.2: The architecture of RNN. The left-hand side shows the concise architecture, which can be unfolded by the time steps as on the right-hand side. The symbol A represents an MLP.

Figure 3.2 shows the classical architecture of RNN. RNNs are often succinctly represented by the diagram on the left-hand side. However, the actual computational process of an RNN can be unfolded across time steps. As illustrated on the right-hand side of the equal sign, the sequence is fed step-by-step into the network over time. Each time step's input $\mathbf{x}_t$, along with the output $\mathbf{h}_{t-1}$ generated from the previous time step, are processed through an MLP network A, to generate a new output $\mathbf{h}_t$ by the following way:

$$\mathbf{h}_t = \sigma(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1} + \mathbf{b}), \tag{3.15}$$

where $\mathbf{U}$ and $\mathbf{W}$ are the weight matrices of the network, and $\mathbf{b}_h$ and $\mathbf{b}_y$ are the bias vectors, similar as in an MLP. $\mathbf{U}$ is the weight matrix for the neurons connected from input $\mathbf{x}_t$ to output $\mathbf{h}_t$, while $\mathbf{W}$ is the weight matrix for the neurons connected from previous output $\mathbf{h}_{t-1}$ to current output $\mathbf{h}_t$.

Depending on the specific application and the desired output length, RNNs can be structured in several different ways. The two foundational architectures are the many-to-one (n-to-1) and many-to-many (n-to-n) architecture.

In an n-to-1 RNN architecture, only the output of the last time step is used as the model's final output y. This structure is typically used for classification or regression tasks. In an n-to-n architecture, the outputs of all steps are used as the model's final output. An n-to-n RNN processes a sequence of inputs and generates a sequence of outputs with the same length of input, making it suitable for applications like machine translation, where each word in the input sentence corresponds to a word in the output sentence.

However, the requirement that the input and output sequences must have the same

length limits the flexibility of this architecture. There are many scenarios where the lengths of input and output sequences do not match. For example, in Text summarization, the input is a long passage of Text and the expected output is a shorter summary. To address such scenarios, a more general approach is needed, which is referred to as the n-to-m architecture or Sequence-to-Sequence (Seq2Seq, Sutskever et al., 2014) model. The Seq2Seq model consists of two RNNs, the one is called Encoder and the other is called Decoder. The former encodes the input data into a context semantic vector $\mathbf{c}$. The semantic vector $\mathbf{c}$ can be obtained in several ways. The simplest method is to assign its output of the last steps (or a transformation of the last steps) to $\mathbf{c}$, e.g., the Entcoder is an n-to-1 RNN. Another option is to use the output of its all steps to compute $\mathbf{c}$, e.g., the Entcoder is an n-to-n RNN. After obtaining $\mathbf{c}$, the Decoder, another n-to-n RNN network use $\mathbf{c}$ as its input to compute an output sequence.

Training an RNN involves tuning its weights and biases to minimize a loss function, just like other types of neural networks. However, the recurrent nature of RNNs requires a specialized training algorithm known as backpropagation through time (BPTT, Werbos, 1990). During BPTT, the RNN is unfolded across time steps, and gradients are computed for each time step, treating the series of recurrent layers as a single feedforward MLP network. RNNs face challenges with long-term dependencies due to vanishing and exploding gradient problems. LSTM networks address these issues by introducing a more complex architecture.

In LSTM networks, the traditional MLP network "A" shown in Figure 3.2 is replaced with a more complex structure called a cell, as depicted in Figure 3.3. An LSTM cell consists of several gates that regulate the flow of information, ensuring that important information is retained over long periods. Unlike the MLP network in a classical RNN, which generates a single output, the LSTM cell produces two outputs at each time step: $\mathbf{C}_t$ and $\mathbf{h}_t$. $\mathbf{C}_t$ serves as the long-term information (memory) of the LSTM, retaining important information across many time steps. The mechanisms within the cell ensure that $\mathbf{C}_t$ can carry and transfer relevant information, allowing the network to avoid the vanishing gradient problem. $\mathbf{h}_t$ functions as the short-term memory, capturing the recent context that is directly connected to the current output. For a detailed description of the LSTM cell, please refer to Hochreiter and Schmidhuber (1997).

3.2 Unsupervised models

In addition to supervised learning, unsupervised learning models such as Hidden Markov Models (HMMs, Rabiner, 1989) and Latent Dirichlet Allocation (LDA, Blei et al., 2003) are also used in cognitive science. Hidden Markov Models (HMMs) are statistical models used to describe systems where the modeling requires assumptions about unobservable (hidden) states and the observable outputs or observations.

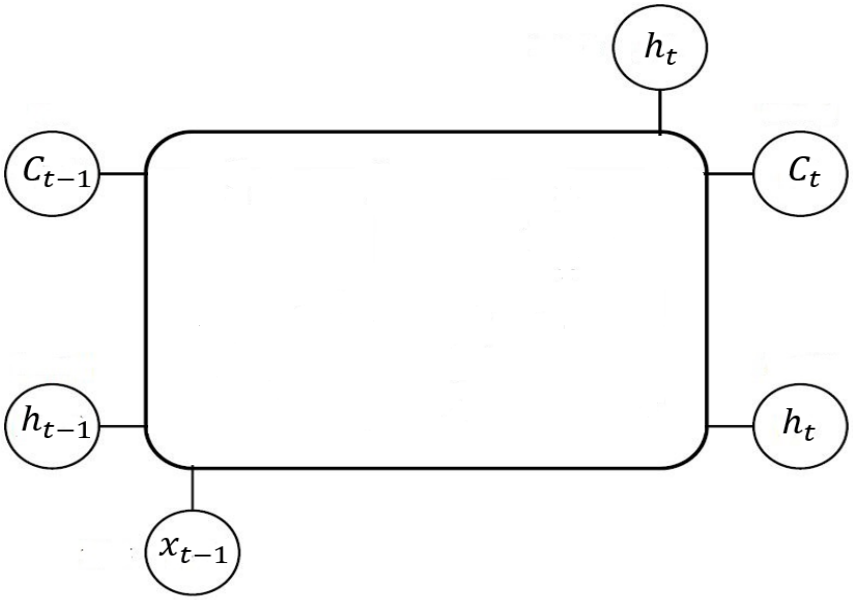

Figure 3.3: A cell of LSTM model.

Latent Dirichlet Allocation (LDA) is for topic modeling in natural language processing. It assumes that each document in a corpus is a mixture of a small number of topics, and each topic is characterized by a distribution of words.

Since my PhD research did not delve into the application of unsupervised learning models, no further discussion on these topics will be provided here. For a detailed description of the unsupervised models, please refer to Ghahramani (2003).

3.3 Contributed publications

My contributed publications to this Chapter are Article (I) and Article (III).

In Article (I), we examined the rating performance of a supervised learning pipeline on creative thinking tasks, designed to measure an individual's ability to generate original and useful ideas. Creative thinking tasks often involve divergent thinking, which is commonly assessed through Alternate Uses Task. This study emphasizes the importance of properly selecting and composing features for automated scoring models in creative tasks. A summary of Article (I) will be provided in Section 4.2.

In Article (III), a novel Trajectory classification model is proposed. This model using the spatial and temporal distance matrices of trajectories as its inputs. This method utilizes a kNN-based approach to classify trajectories by recognizing shape

similarities, even when the order of points varies significantly. For the distance of Trajectories used in kNN, a CNN model with spatially designed convolution Kernels is learned from the labeled data, instead of using a pre-defined distance mseasure. The model demonstrated superior performance compared to traditional DTW+kNN and LSTM models on a benchmark dataset. A summary of Article (II) will be provided in Section 4.3.

4 Summary of the Articles

4.1 Article (I)

The article summarized in the following is published in The Journal of Creative Behavior: Buczak et al. (2023). The machines take over: A comparison of various supervised learning approaches for automated scoring of divergent thinking tasks. *The Journal of Creative Behavior*, 57(1), 17-36. https://doi.org/10.1002/jocb.559.

4.1.1 Motivation

The motivation for this research is based on the fact that scoring creative thinking tasks, such as the Alternate Uses Task (AUT), is a labor-intensive process requiring human and time resources. This has driven interest in automated scoring methods for creative thinking tests, as documented by several recent studies (e.g., Beaty and Johnson, 2021; Dumas et al., 2021; Stevenson et al., 2020). Previous research has primarily utilized semantic distance approaches, which are examples of unsupervised machine learning, to quantify originality in responses. These approaches have shown strong correlations between semantic distance scores and scores provided by human raters. This article seeks to extend and complement existing AUT scoring research by focusing on supervised learning approaches, which have been less extensively studied. This article specifically examines the potential of various supervised learning algorithms (e.g., Random Forest, XGBoost, and Support Vector Regression) and different feature set compositions to improve rating performance and identify key features relevant to predicting human ratings.

4.1.2 Pipeline for Automated Scoring

To achieve the goals described above, a structured machine learning pipeline for automated scoring (as shown on the right-hand side of Figure 4.1) is implemented. In contrast to the traditional approach where human raters directly evaluate the raw response data (left-hand side), the machine learning approach involves additional data pre-processing step and feature engineering step. Pre-processing includes removing excess spacing and punctuation as well as correcting coding errors. After pre-processing, a set of informative features is generated from the responses. This

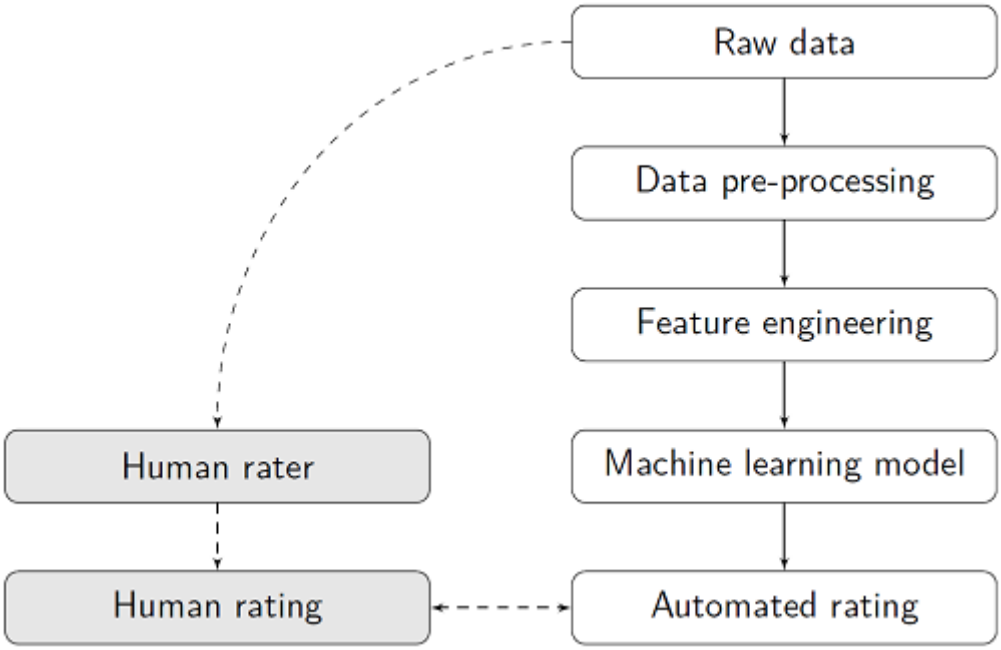

Figure 4.1: A machine learning Pipeline for automatically scoring of DT tasks.

prepared data set is then used to train a machine learning model. The training process includes steps like model selection, feature selection, and hyperparameter optimization. Once the final model is trained and optimized, it can be used to predict scores for new response data.

Two Divergent Thinking (DT) datasets from studies by Silvia et al. (2008) and Hofelich-Mohr et al. (2016), both of which include rated responses from Alternate Uses Tasks (AUTs) are used to evaluate the pipeline. The generated features are as shown in Figure 4.2. These features can be categorized based on their type and interpretability.

The "meta-features" category includes features such as the "number of words", "average word length", and "maximum word length" in a response. These features are interpretable and can be directly extracted from the responses.

Another category involves Word Embeddings (WE, see Section 2.2.1 of this thesis), which forms the word vector representation of a response in a pre-trained semantic space. Pre-trained semantic spaces usually offer vector representations for individual words, whereas responses in DT tasks are typically phrases or sentences. To generate sentence embeddings, we employed two different strategies as described in Section 2.2.1 of this thesis. The first strategy involves summing the Word Embeddings of the individual words contained in the response. The second strategy uses zero padding to align sentence lengths.

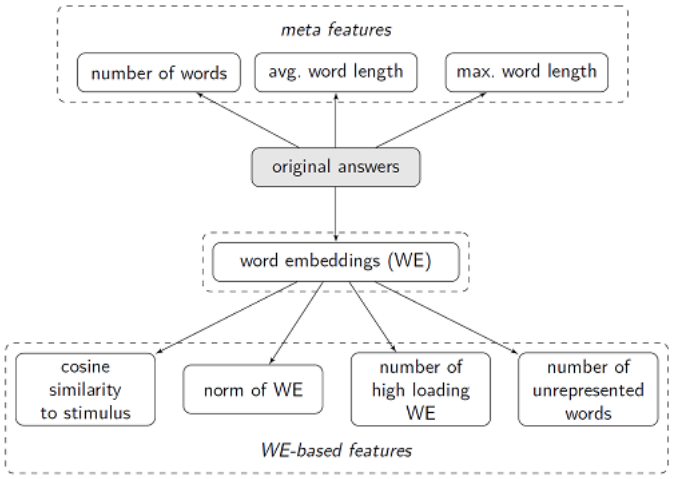

Figure 4.2: Features from the responses in DT tasks.

4.1.3 Results

In Article (I), we compared the predictive performance of three supervised machine learning algorithms, the Random Forest (RF), XGBoost (XGB), and Support Vector Regression (SVR) for the automated prediction of creative score in AUTs on the data sets under different feature sets. The key findings of this study are as follows:

- Algorithm Performance: RF and XGB generally outperformed SVR, especially when combining meta-information with WEs and WE-based features. The best model, an RF, achieved high person-level correlations for two different datasets.

- Cross-Sample Challenges: Predicting ratings across different datasets revealed generalizability issues due to rating mean differences. Models trained on one dataset often did not perform well on another, highlighting challenges in predicting mean ratings across samples.

- Feature Analysis: Including WEs significantly improved model performance. However, the WEs are not interpretable. Thus, there was a trade-off between feature interpretability and predictive performance. Reducing feature set dimensionality using Principal Component Analysis (PCA) maintained performance, pointing to a potential for reducing computational complexity.

- Semantic Space and Word Embedding Composition: The choice of semantic space (GloVe vs. Word2Vec) did not significantly affect performance, but the method of composing WEs (additive vs. zero-zero padding) did. Zero-padding improved predictive performance but also increased computational costs.

4.2 Article (II)

The article summarized in the following is published in Behavior Research Methods: Huang et al. (2024). Short-time AOIs-based representative scanpath identification and scanpath aggregation. *Behavior Research Methods*, 1-16. https://doi.org/10.3758/s13428-023-02332-w.

4.2.1 Motivation

Eye-tracking technology has seen widespread application in both applied and scientific research over the past few decades. It offers valuable insights into the cognitive processes underlying human behaviors by tracking where and how long participants focus their visual attention during various activities. However, due to the high degree of freedom in viewing choices, understanding the scanning strategies of a sample remains challenging. Article (II) introduces a novel method for identifying an aggregated scanning strategy at the sample level using a representative scanpath.

Previous research has generally used two methods to calculate a representative scanpath for a group. One method is to identify an existing scanpath within the sample that has the highest average similarity to the other scanpaths. However, this method heavily relies on the definition of scanpath similarity and is less robust to noise. The other method calculates an artificial scanpath where the fixations are derived from Areas of Interest (AOIs). The challenge with this method lies in the definition or computation of AOIs. Some studies predefine AOIs based on the semantic information of the stimuli, while others, after collecting a set of scanpaths, derive AOIs by clustering the fixations on the scanpaths. We refer to the latter as the "global AOI-based" method because these AOI calculations do not take into account the local temporal information of the fixations.

To better utilize the temporal and sequential information of fixations, Article (II) introduced a novel method for calculating a representative scanpath based on short-time AOIs. The short-time AOIs are obtained from the fixations by considering their period of the view process.

4.2.2 Short-Time AOIs-Base Scanpath Aggregation

The principal idea of STASA is to first identify short-time AOIs and then derive a representative scanpath by linking these AOIs in temporal order. As shown in Figure 4.3, scanpaths are first represented in a 3D coordinate system by adding a time axis to the 2D space. Fixations during specific time frames are clustered using the DBSCAN algorithm. The optimal DBSCAN parameters (ϵ and minPt) are determined iteratively to find the densest and largest clusters in each time frame.

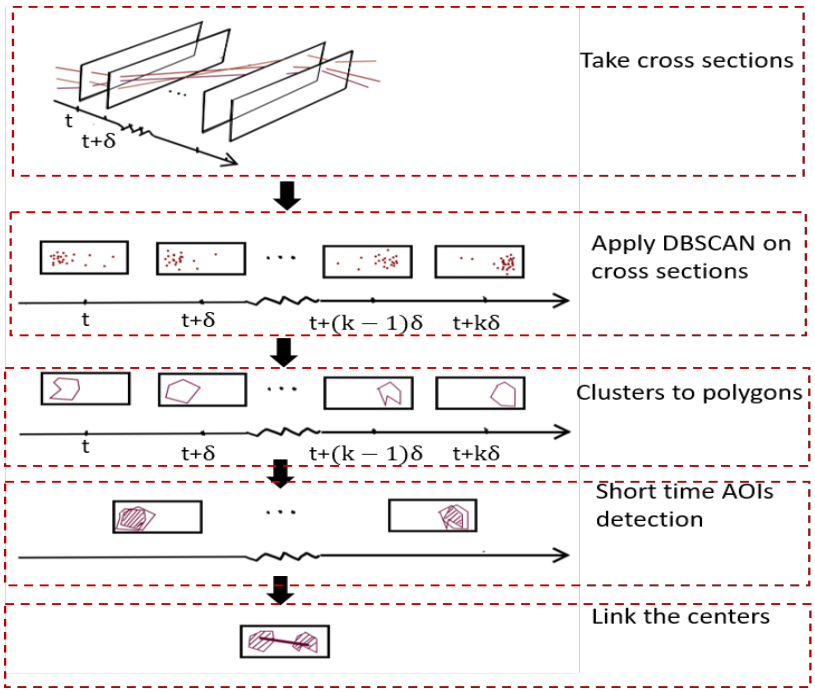

Figure 4.3: Overview of the STASA method.

Once clusters are formed in each time frame, their spatial boundaries are estimated using the concave hulls of the clusters. The similarity of clusters on adjacent time frames is calculated to determine the spatial and temporal boundaries of short-time AOIs. The representative scanpath is constructed by linking the centers of these short-time AOIs. The space coordinates of the fixations are the geometric centers of the AOIs, and their temporal duration is defined by the start and end times of the short-time AOIs.

4.2.3 Evaluation of STASA

To evaluate the STASA method, Article (II) compared its performance, which is measured by the representativeness of the computed scanpath for a group, with the CDBA method and Heuristic method from Li and Chen (2018). Theses methods are based on the global AOIs obtained by clustering the fixations on scanpaths without considering their time information. The comparison are conducted on two large public available eye-tracking datasets: MIT1003 (Judd et al., 2009) and OSIE (Xu et al., 2014). The representativeness of the computed scanpath for a group is evaluated by its average similarity with the group members. To calculate the representativeness, three different similarity measures are employed: Scasim, MultiMatch, and ScanMatch. MultiMatch evaluates the similarity of two scanpaths across five dimensions: vector, direction, length, position, and duration, while ScanMatch and Scasim offer an

overall similarity score. Results show that STASA outperformed the CDBA and Heuristic methods on both datasets, particularly in terms of overall similarity scores calculated by ScanMatch. Detailed analysis using MultiMatch revealed that STASA exhibited superior vector similarity and length similarity to the group members. Additionally, the positional similarity of fixations derived from STASA proved more representative compared to the center-based approach on global AOIs used in the other two methods.

Additionally, Article (II) compared STASA with a baseline method of identifying the most representative scanpath directly from the raw sample. The measures Scasim and ScanMatch were used here to identify the scanpath with the smallest dissimilarity to other sample members. Overall, the results affirm that STASA offers a more accurate and representative scanpath by effectively incorporating both temporal dynamics and spatial information.

4.2.4 Implementation of STASA

STASA has been implemented as an R (R Core Team, 2021) package named *scanpathAggr*, which is avaliable on Github (https://github.com/HeHuangDortmund/scanpathAggr). *scanpathAggr* provides functions for computing representative scanpath and visualization functions. As shown in Figure 4.4, given a set of scanpaths, the functions in this package compute a representative scanpath, and use the visualization functions to display all the scanpaths and the computed representative scanpaths on stimuli, and also includes the computed short-time AOIs.

4.3 Article (III)

The article summarized in the following has been conditionally accepted by conference *In 2024 7th International Conference on Artificial Intelligence and Pattern Recognition*, which will be held from September 20th to 22nd in Xiamen, China.

4.3.1 Motivation

Automatic handwriting recognition has emerged as a pivotal research area intersecting cognitive science, machine learning, computer vision, and pattern recognition. It can be broadly classified into online and offline recognition. Online systems use a tablet digitizer to capture dynamic pen tip movements, benefiting from temporal information such as pen pressure and depth captured by special devices. However, these systems require specialized equipment, limiting their accessibility. Offline systems, in contrast,

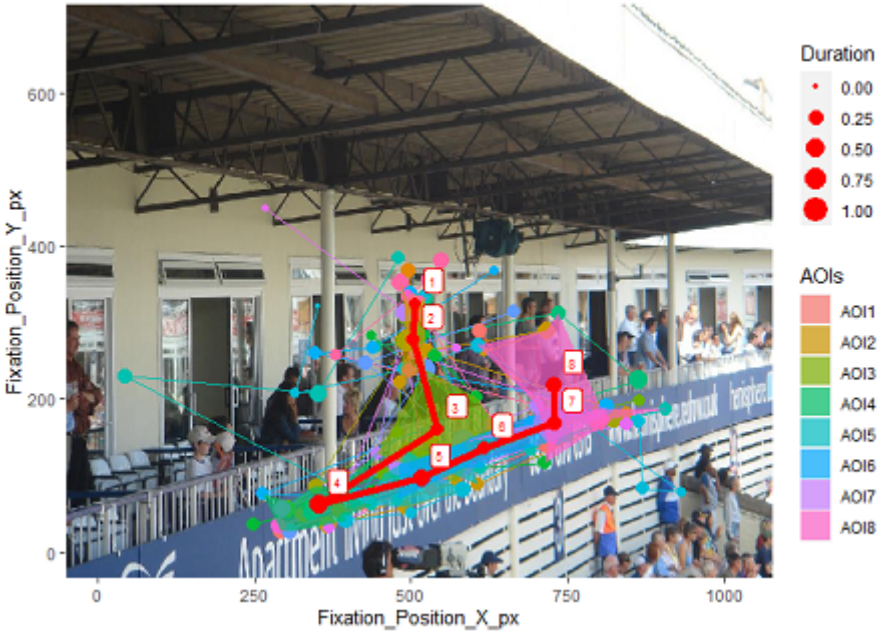

Figure 4.4: An example of the computed representative scanpath by the R package *scanpathAggr*. The thick red line represents the representative scanpath. The rest of the thin lines represent the original scanpaths, different colors represent different scanpaths. The dots on the scanpath represent fixations, the diameter of the dots is proportional to the duration of the fixation. In addition, short-term AOIs are also represented by different colors.

analyze static images from scanners or cameras, offering more versatility but facing challenges in segmenting individual characters from the images.

The recognition process generally involves two steps: segmenting continuous text into individual characters and then classifying these characters. For online recognition, the availability of temporal information aids in more precise segmentation. In offline recognition, segmentation and classification are more challenging due to the static nature of the input. The state-of-the-art models for both online and offline recognition harness Deep Learning techniques, with CNN excelling in image classification tasks and LSTMs proving effective for trajectory classification in online systems.

To bridge the gap between the advantages of online and offline recognition, researchers have explored techniques to recover temporal information from static images of handwriting, thereby improving offline recognition performance. However, this process can produce artifacts such as zigzagged strokes and noise sensitivity, complicating character classification. While some studies focus on developing more accurate trajectory recovery techniques, Article (III) focuses on improving the classification accuracy of trajectories obtained through existing recovery techniques. To achieve this goal, a novel trajectory classification model is proposed in Article (III).

4.3.2 Spatiotemporal Kernel Convolution Network

The proposed model in Article (III) is a CNN model called Spatiotemporal Kernel Convolution Network (STKNet). This model takes two handwriting trajectories as input and calculates the distance of them. After that, the kNN method is used to assign labels for new data.

The architecture of STKNet is shown in Figure 4.5. STKNet is a CNN but has two main differences from conventional CNNs. One is that it takes two matrices as input. Spatially, given two trajectories $Tr^{(1)} = \{\mathbf{x}_1^{(1)}, \cdots, \mathbf{x}_n^1\}$ and $Tr^{(2)} = \{\mathbf{x}_1^{(2)}, \cdots, \mathbf{x}_n^{(2)}\}$, STKNet computes first their spatial distance matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ and temporal distance matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ as the input of the convolution layer, where $\mathbf{D}_{st} = d(\mathbf{x}_s^{(1)}, \mathbf{x}_t^{(2)})$ is the spatial distance between s-th point on $Tr^{(1)}$ and t-th point on $Tr^{(2)}$, and $\mathbf{T}_{st} = |s - t|$. The temporal distance matrix $\mathbf{T}$ is constructed from the differences in the indices of the points on the two trajectories, implying that the points on the trajectories are obtained at equal time intervals. If the timestamps of the points timestamp were known, then $\mathbf{T}_{st}$ could be calculated as the absolute difference between the timestamp of the s-th point on the first trajectory and the timestamp of the t-th point on the second trajectory. The other difference between STKNet and a conventional CNN is that a so-called spatiotemporal convolution operation is conducted simultaneously on $\mathbf{D}$ and $\mathbf{T}$ to extract both spatial and temporal similarity features. Denote $\mathbf{Y}$ the resulting matrix of the spatiotemporal convolution of $\mathbf{D}$ and

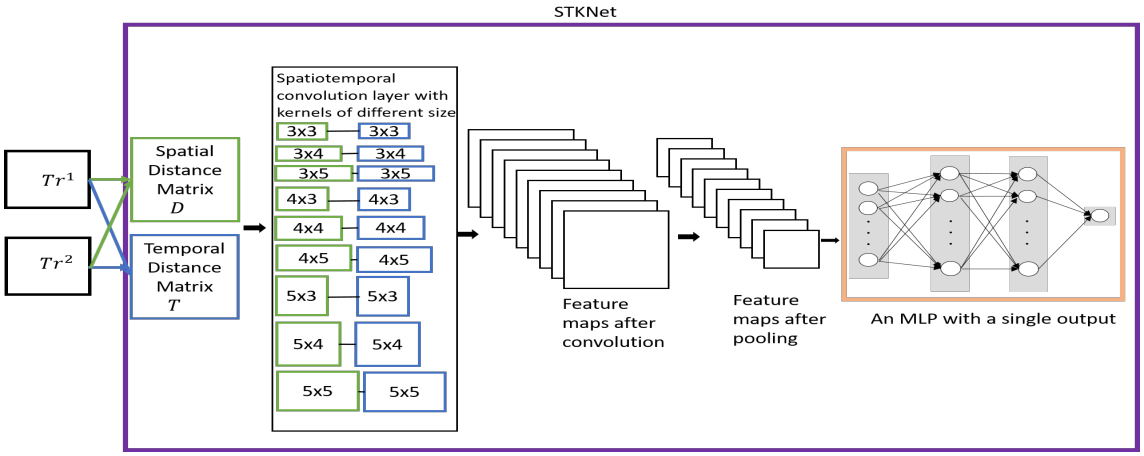

Figure 4.5: An Overview of the STKNet model.

$\mathbf{T}$ with a convolution Kernel $\mathbf{K} \in \mathbb{R}^{u \times v}$, its st-th element is computed with

$$\mathbf{Y}_{st} = \mathbf{C}_{st} \cdot \mathbf{W}_{st}, \tag{4.1}$$

where $\mathbf{C}_{st}$ is the output of the conventional convolution of $\mathbf{D}$ and $\mathbf{K}$ which computed using Equation 3.14, and $\mathbf{W}_{st}$ is the weight of the spatial convolution $\mathbf{C}_{ij}$ and computed from the temporal distance matrix with

$$\mathbf{W}_{st} = \sum_{\mu=0}^{u-1} \sum_{\nu=0}^{v-1} c\mathbf{T}_{s+\mu,t+\nu}, \tag{4.2}$$

where $c = \frac{1}{uv}$ is a constant decided by the size of the convolution kernel $\mathbf{K}$. This is equivalent to using two convolutional kernels of the same size on the spatial distance matrix and temporal distance matrix at the same time; the convolutional kernel on the spatial distance matrix has learnable parameters, while the convolutional kernel on temporal distance has fixed parameters c. While the spatial convolution on systematically extracts features of various combinations of sub-trajectories from the two input trajectories, the temporal convolution on weights the extracted features by the temporal distance of the sub-trajectories.

4.3.3 Evaluation of STKNet

The proposed model was evaluated on the MNIST stroke sequence dataset (de Jong, 2016) and the performance is compared with other two models, e.g., the DTW+kNN and the LSTM. DTW+kNN and the methods in Articel (III) share a common methodology that involves computing the distance between each trajectory in the test set and all trajectories in the training set, followed by label assignment based on the kNN algorithm. LSTM, recognized for its superior performance on the MNIST

stroke sequence dataset, is considered one of the best-performing models in this context.

In the conducted experiments, the models are not trained on the full set of 60,000 samples of the MNIST stroke sequence dataset. Instead, the experiments begin with 100 randomly selected samples, and the size of the training set is gradually increased. During this process, the prediction accuracy of different models is observed on the entire 10,000 test samples with varying training set sizes. Results show that STKNet outperforms other models for all cases and achieves state-of-the-art prediction accuracy with only 1,200 trajectories in the training sample, a relatively small number of labeled samples compared to 60,000 MNIST training images.

The MNIST stroke sequence dataset contains handwriting trajectories of digits from 0 to 9. When analyzing the recognition performance for individual digits, all models exhibited high accuracy for digits 0 and 1. However, STKNet outperformed DTW+kNN and LSTM for digit 2, which those models often misclassified as digits 3 and 7. Digits 8 posed the most significant challenge for DTW+kNN and LSTM, as they showed the worst accuracy for this digit. In contrast, STKNet maintained consistent accuracy across all digits, including digit 8.

One of the critical challenges in handwriting recognition is the variability in writing orders. STKNet demonstrated its strength in classifying trajectories with varying writing orders. DTW and LSTM models were particularly affected by different writing orders in small training sets due to their sensitivity to trajectory order, leading to significant errors. In contrast, STKNet exhibited robustness in recognizing similarly shaped trajectories regardless of the writing order. This characteristic enabled the model to maintain high accuracy even for complex or irregular trajectories, as long as the general shape was retained.

5 Discussion

This cumulative doctoral thesis integrates contributions from three distinct research publications into the fields of sequence data mining for cognitive science. Each publication addresses unique challenges and introduces innovative methodologies, contributing to their respective chapters. Article (I) focuses on supervised learning models for assessing creative thinking tasks, employing machine learning algorithms such as Random Forest, XGBoost, and Support Vector Regression. It highlights the effectiveness of combining meta-information with Word Embeddings to predict mean creative quality ratings. Article (II) introduces a novel approach to computing group centroids for scanpaths, improving its representativeness compared to traditional methods. Article (III) presents a trajectory classification model based on spatial and temporal distance matrices, demonstrating robust performance in classifying handwritten trajectories. However, each publication has its limitations, the discussion of which will be summarized as follows.

5.1 Article (I)

In Article (I), we aim to improve the automated scoring of responses in Divergent Thinking (DT) tasks, specifically using data from Alternate Uses Tasks (AUTs) by Silvia et al. (2008) and Hofelich-Mohr et al. (2016). Three machine learning algorithms, Random Forest (RF), XGBoost (XGB), and Support Vector Regression (SVR) are compared for their predictive performance. The study developed a pipeline that includes preprocessing of raw response data and generating interpretable and non-interpretable features, such as Word Embeddings (WE) using pre-trained models like GloVe and Word2Vec. The analysis shows that RF and XGB slightly outperformes SVR. The best models achieved person-level correlations from 0.72 to 0.75 for Hofelich-Mohr et al.'s data and 0.58 to 0.65 for Silvia et al.'s data.

However, challenges in cross-sample predictions are identified due to mean rating differences by human raters. The models struggled to maintain performance when applied to a different dataset than the one on which they were trained. This limitation is predominantly due to variations in the mean ratings assigned by different groups of raters, which introduced a significant bias.

Another critical limitation was the lack of demographic data, which prevented an analysis of potential biases. This absence poses a risk of unacknowledged biases affecting model predictions.

Moreover, the two data processing techniques, zero padding and additive composition for sentence embeddings, each presented their own drawbacks, including information loss and increased computational demands. Additionally, relying on non-interpretable features like Word Embeddings posed challenges in understanding and interpreting model predictions, which is essential for practical applications.

To address these limitations, future research should incorporate several key improvements. Improving cross-sample robustness is a crucial aspect, requiring methods to standardize rating scales through normalization techniques applied post hoc or calibration studies to align rater scales ad hoc. Normalization techniques might involve Z-Score Normalization (Standardization) and Min-Max Normalization. The former converts the ratings to a standard score by subtracting the mean and dividing it by the standard deviation of the ratings in a given dataset. The latter subtracts the minimum from the raw score and divides it by the possible scoring range. Although normalization converts the scores of different samples to the same range, the interpretation of the scores remains sample-specific. This is because the position of each individual is evaluated against all others in the same sample. On the other hand, calibration studies seek to align the scales used by different raters through systematic methods, making it possible to directly compare scores across different samples. Possible calibration methods may include:

- Anchor Items: Introduce a set of pre-determined items with known difficulty or quality levels that are rated by all raters. These items serve as benchmarks or anchors in the rating process.
- Consensus Building Sessions: Organize sessions where raters collaboratively discuss and rate several example responses until a high consensus level is reached. These sessions help standardize the interpretation of rating criteria.

In addition, incorporating demographic data in future studies would enable an analysis of biases and their impact on model predictions. A classic statistical model that can handle demographic information is the Mixed-Effects Model (see Pinheiro and Bates, 2006). This model includes random effects to account for variations across different demographic groups, which can capture group-level variance and adjust predictions accordingly. However, this approach faces significant computational complexity, particularly when dealing with high-dimensional variables like the WE representation. Therefore, it is necessary to reduce data complexity during the pre-processing stage. To addressing the complexity of data processing, future research should investigate how dimensionality reduction techniques, such as Principal Component Analysis (PCA) on Word Embeddings, can reduce computational complexity without sacrificing accuracy.

Moreover, hybrid models that integrate unsupervised, semi-supervised, and supervised methods could leverage the strengths of each approach, potentially bringing machine predictions closer to human ratings. Expanding the dataset variety and size would also be beneficial, enabling evaluations of model robustness and generalizability across different populations and tasks. By addressing these areas, future research has the potential to significantly improve the accuracy, fairness, and generalizability of automated scoring systems for creative thinking tasks.

5.2 Article (II)

Article (II) introduces a novel method (called STASA) for identifying a representative scanpath within a sample. This article proposes the concept of short-time AOIs, derived from scanpath fixations over short periods, to determine the representative scanpath fixations. This contrasts with previous methods that rely on global AOIs, which aggregate all scanpath fixations without considering their sequence or duration. By comparing with the global AOI-based approach across two publicly available eye-tracking datasets, it is found that the proposed method yields scanpaths with higher representativeness. However, the absence of publicly available code and data for the global AOI-based methods precludes statistical testing, limiting their conclusions to average performance comparisons. Future work could focus on developing and releasing an open-source implementation of both the proposed short-time AOI method and the traditional global AOI-based methods. This would allow for thorough statistical comparisons and enable other researchers to replicate and build upon the findings.

Article (II) also discusses the efficiency of STASA against simpler approaches that identify an existing scanpath as the representative scanpath. While the proposed method shows higher representativeness, it is noted to be slower compared to simpler approaches to identifying a representative scanpath. Future research could investigate optimization techniques and parallel computing strategies to improve the speed and efficiency of the method, making it more suitable for real-time applications and larger datasets.

This study aims to establish a general methodology for generating representative scanpaths for a group and proposes a parameter-tuning strategy for STASA to accommodate different scanpath similarities specified by users. Reflecting on the inherent subjectivity in choosing similarity measures, it is acknowledged that no universally unbiased scanpath similarity exists due to the diverse aspects covered, such as temporal, spatial, and semantic attributes. Nonetheless, the method can be adapted to the user's specific requirements by adjusting the parameter. Reflecting on the proposed parameter-tuning strategy (grid search), future work could extend this to develop an adaptive, possibly automated, parameter optimization system that

adjusts based on specific datasets and user-defined similarity metrics. This could make the method more versatile and user-friendly.

Furthermore, the proposed method is distinct from scanpath clustering or classification models, which either identify visual strategies across multiple viewer groups or classify viewers based on scanpath similarities. While STASA is designed for single-group analysis, it could theoretically be extended to scanpath classification by computing a representative scanpath for each group and assigning new scanpaths to the most similar group. It could also serve as a building block for scanpath clustering, providing a representative scanpath as a cluster centroid.

Article (II) additionally considers the impact of noisy scanpaths and the presence of salient objects on stimuli. The relationship between the number of objects in an image and the representativeness score of the scanpath is explored, hypothesizing that fewer salient objects lead to higher representativeness due to more uniform gaze strategies among observers.

5.3 Article (III)

In Article (III), a novel trajectory classification model (named STKNet) was proposed, utilizing the spatial and time distance matrix between two trajectories as input. This model extracts information from these distance matrices to calculate the distances between trajectories and subsequently uses kNN for classification. This approach proves particularly adept at recognizing shape similarities in trajectories, even when there are significant differences in the order of points along these trajectories.

The validation experiments demonstrated that for small sample sizes, using recovered handwritten trajectories leads to higher recognition accuracy compared to using the original images. The model's performance was compared against the classical DTW+kNN and the state-of-the-art deep learning LSTM model on the MNIST sequence dataset, derived from recovered trajectories from images. It consistently outperformed these models, achieving state-of-the-art performance with a limited amount of training data, enhanced by data augmentation techniques applied to the small set of raw labeled data.

The superior performance of STKNet can be attributed to its utilization of both temporal and spatial distance matrices of trajectories as inputs, which theoretically encompass all similarity information between the two trajectories. Unlike DTW, which uses linear programming to derive distances from the distance matrix without training using labeled data, STKNet adjusts its parameters based on label information, adapting to the current data, such as noise levels and zigzags in recovered trajectories. Moreover, STKNet is less sensitive to variations in writing order compared to DTW and LSTM, which are designed for sequential data and can struggle when the test sample's writing order significantly differs from the training set, an issue particularly

prevalent when training sets are small. Additionally, DTW and LSTM struggle with recognizing recovered hand-written trajectories containing noise-induced anomalies such as loops or outlying points. Our method overcomes this by using the distance matrix as input and employing convolutional operations to extract local shape similarities, effectively recognizing trajectories with similar overall shapes.

Theoretically, the proposed model could be applied to online handwriting trajectory recognition. A limitation of this study is the absence of benchmark datasets for online handwriting trajectory. Future research could focus on creating labeled datasets for both offline and online handwriting trajectories. Additionally, the MNIST dataset used in this study includes only digits and does not cover alphabetic characters. In future work, efforts should be made to create datasets that include alphabetic characters. This would involve collecting handwriting samples that encompass a full set of letters (both uppercase and lowercase), numerals, and common symbols, ensuring a comprehensive evaluation of STKNet in recognizing a broader range of handwriting. The potential of STKNet can be further tested by validating it on handwritten text in various languages, such as Arabic script and Chinese characters. This involves collecting and annotating datasets that contain diverse handwriting styles and scripts. Conducting experiments across multiple languages will help establish the model's effectiveness and generalizability in handling different writing systems, which have unique structural and stylistic characteristics.

Moreover, like DTW and LSTM, STKNet could potentially be used for other trajectory classification applications, such as traffic trajectories, animal movement trajectories, or eye movement trajectories, which will be explored in future research. Additionally, STKNet may also be applied to attributed trajectories, such as scanpaths. For attributed trajectories, in addition to spatial and temporal distance matrices, the differences in the attributes of points can also form an attribute difference matrix. These matrices (spatial, temporal, and attribute difference matrices) can be simultaneously input into the CNN. However, designing suitable convolutional kernels is essential to extract features from different distance matrices and handle the relationships between them.

In addition, there is another notable limitation to the proposed method. That is, the STKNet model, although designed to calculate distances between trajectories, does not serve as a metric d, which should satisfy the following axioms for all samples A, B, C in the sample space:

- Identity: $d(A, A) = 0$.
- Positivity: if $A \neq B$, then $d(A, B) > 0$.
- Symmetry: $d(A, B) = d(B, A)$.
- Triangle inequality: $d(A, B) \leq d(A, C) + d(B, C)$.

These axioms ensure that the distance function d behaves consistently with our intuitive understanding of distance in Euclidean space, making the results of analyses more interpretable.

To approximate the Identity property during the training of STKNet, one approach is to include pairs of each sample with itself,i.e., (A, A), in the training dataset with a label of 0. This teaches the model that the distance between identical samples should be zero.

To ensure Positivity, a possible technique is to impose constraints on the model to restrict all parameters to non-negative values. For STKNet, when all the parameters are non-negative, its output will also be non-negative.

In terms of achieving Symmetry, since the final output of the STKNet is determined by a fully connected layer and the distance matrix of (A, B) is the transpose of the distance matrix (A, B), a potential method to obtain the same feature vector from the fully connected layer is to use a global max pooling layer after the convolutional layers. This technique, however, might result in a significant loss of information. Therefore, consideration and balance need to be made between ensuring Symmetry and preserving necessary trajectory information.

To satisfy the triangle inequality, a possible implementation method involves adding a penalty term to the loss function. This term can impose a non-zero penalty on triplets failing to satisfy the triangle inequality. However, this approach presents a significant computational challenge. The value of the loss function cannot be calculated easily by summing individual sample pairs; instead, it requires considering all sample triplets simultaneously. This necessity involves using all samples when updating model parameters in each iteration (batch learning). For large datasets, this requirement may exceed the available memory capacity of the computer.

Overall, ensuring that the model approximately satisfies these metric axioms can significantly increase the difficulty of model training. It requires careful design of training strategies, loss functions, and model architecture adjustments to respect these complex constraints. Furthermore, extensive experimentation will be necessary to balance these constraints with maintaining the model's performance and generalizability.

Finally, the experiments in Artikel (III) are limited to the recognition of single characters. However, the complete handwriting recognition process involves segmenting a document into individual characters before recognition. The integration of STKNet into the entire recognition process and its performance in such a context have not been validated and need to be explored in future research. This will involve extensive testing and possibly refining both the model and the pipeline to ensure seamless and accurate character recognition in real-world applications.

5.4 General Limitations

In the following, I will discuss some general limitations of this thesis and the overall work, beyond the specific limitations mentioned in each article.

Generalizability

Generalizability is a primary challenge faced by the methods proposed in this thesis. There are two levels of generalizability: the first pertains to the application of the same method across different datasets of the same type, and the second involves the application of the same method across different types of sequence data. On the one hand, the models may perform well on hand-curated datasets but may not generalize to noisy, real-world data that exhibit different characteristics. On the other hand, while these methods have demonstrated success in specific domains such as text processing, eye-tracking analysis, and handwriting recognition, their applicability to other types of sequence data remains to be validated.

To address this limitation, future research should:

- Conduct extensive cross-domain evaluations by testing the models on a wide variety of sequence data from different fields.
- Implement domain adaptation techniques to improve the models' ability to generalize across different types of data.
- Generate synthetic data that mimic real-world variances to test the models' robustness and adaptability.
- Explore methods to integrate these diverse data sources within a unified framework. This integration can harness the complementary strengths of different data types, potentially leading to improved model accuracy and broader applicability in real-world contexts.

Computational Demands

The computational efficiency of the proposed methods is another critical limitation, particularly when dealing with large-scale datasets. Methods like the trajectory classification model and the representative scanpath aggregation method can become computationally intensive. To mitigate this, future work could explore:

- Optimizing algorithms for parallel processing to take full advantage of multi-core processors and GPUs.
- Implementing more efficient data structures and algorithms for faster computation.

- Exploring approximate methods that provide a good trade-off between accuracy and computational cost.

Benchmark Datasets

Another significant limitation is the absence of benchmark datasets for testing certain proposed methods, such as the trajectory classification model for online handwriting recognition. The lack of standardized datasets makes direct comparisons with existing methods challenging and slows the validation process. To address this gap, future work should focus on:

- Creating and sharing publicly accessible benchmark datasets for various applications, including online handwriting recognition, scanpath analysis, and other sequence data tasks. These datasets should be comprehensive and include a wide range of samples to facilitate thorough testing and validation.
- Establishing standardized evaluation protocols and metrics to enable consistent comparisons across different models and studies.

Interpretability

Additionally, while efforts have been made to incorporate interpretable features into the supervised models, some aspects remain challenging. This is particularly true for models employing complex techniques like Word Embeddings. Ensuring that model predictions are understandable and trustworthy for end-users is crucial for practical adoption.

Besides using interpretable features, interpretability can also be improved through modifications to the model structure and training process. For instance, as discussed in Section 5.3, the model structure of STKNet and its training process could be modified to meet the requirements of a metric.

5.5 Summary

In summary, this cumulative doctoral thesis contributes to the fields of sequence data mining for cognitive science through the integration of three distinct research publications. Each article addressed unique challenges with innovative methodologies: Article (I) focused on enhancing automated scoring of creative thinking tasks with supervised learning models, Article (II) introduced a new method for representative scanpath aggregation based on short-time Areas of Interest (AOIs), and Article (III) proposed an advanced trajectory classification model utilizing spatiotemporal convolution on distance matrices for handwriting recognition.

The empirical validation of these methods demonstrated improvements in performance over traditional approaches. However, several broader limitations were identified, including difficulties with cross-sample generalizability, computational efficiency, and the integration of diverse data types. Concerns about model interpretability and the absence of benchmark datasets for certain applications were also highlighted.

Looking forward, future research should address these limitations by developing more efficient algorithms, improving model interpretability, including demographic data, and creating standardized datasets for comprehensive testing. Integrating diverse data types within a unified framework could further enhance the robustness and applicability of these methods.

Bibliography

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB*, *1215*, 487–499.

Agrawal, R., and Srikant, R. (1995). Mining sequential patterns. *Proceedings of the eleventh international conference on data engineering*, 3–14.

Alt, H., and Godau, M. (1995). Computing the fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, *5*(01n02), 75–91.

Beaty, R. E., and Johnson, D. R. (2021). Automating creativity assessment with semdis: An open platform for computing semantic distance. *Behavior research methods*, *53*(2), 757–780.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, *3*(Jan), 993–1022.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: Forecasting and control*. John Wiley and Sons.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Buczak, P., Huang, H., Forthmann, B., and Doebler, P. (2023). The machines take over: A comparison of various supervised learning approaches for automated scoring of divergent thinking tasks. *The Journal of Creative Behavior*, *57*(1), 17–36.

Chen, T., and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.

Cristino, F., Mathôt, S., Theeuwes, J., and Gilchrist, I. D. (2010). Scanmatch: A novel method for comparing fixation sequences. *Behavior research methods*, *42*, 692–700.

de Jong, E. D. (2016). Incremental sequence learning. *arXiv preprint arXiv:1611.03068*.

Dumas, D., Organisciak, P., and Doherty, M. (2021). Measuring divergent thinking originality with human raters and text-mining models: A psychometric comparison of methods. *Psychology of Aesthetics, Creativity, and the Arts*, *15*(4), 645.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *kdd*, *96*(34), 226–231.

Fahimi, R., and Bruce, N. D. (2021). On metrics for measuring scanpath similarity. *Behavior Research Methods*, *53*, 609–628.

Fournier-Viger, P., Lin, J. C.-W., Kiran, R. U., Koh, Y. S., and Thomas, R. (2017). A survey of sequential pattern mining. *Data Science and Pattern Recognition*, *1*(1), 54–77.

Gadár, L., and Abonyi, J. (2019). Frequent pattern mining in multidimensional organizational networks. *Scientific reports*, *9*(1), 3322.

Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, *63*, 3–42.

Ghahramani, Z. (2003). Unsupervised learning. In *Summer school on machine learning* (pp. 72–112). Springer.

Halioui, A., Valtchev, P., and Diallo, A. B. (2015). Acquisition of generic problem solving knowledge through information extraction and pattern mining. *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, 583–590.

Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. *proceedings of the 17th international conference on data engineering*, 215–224.

Hand, D. J. (2007). Principles of data mining. *Drug safety*, *30*, 621–622.

Hassani, M., Beecks, C., Töws, D., Serbina, T., Haberstroh, M., Niemietz, P., Jeschke, S., Neumann, S., and Seidl, T. (2015). Sequential pattern mining of multimodal streams in the humanities.

He, Q., Borgonovi, F., and Paccagnella, M. (2021). Leveraging process data to assess adults' problem-solving skills: Using sequence mining to identify behavioral patterns across digital tasks. *Computers and Education*, *166*, 104170.

Hinsz, V. B. (1990). Cognitive and consensus processes in group recognition memory performance. *Journal of Personality and Social psychology*, *59*(4), 705.

Hirate, Y., and Yamana, H. (2006). Generalized sequential pattern mining with item intervals. *J. Comput.*, *1*(3), 51–60.

Hirschberg, D. S. (1977). Algorithms for the longest common subsequence problem. *Journal of the ACM (JACM)*, *24*(4), 664–675.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huang, H., Doebler, P., and Mertins, B. (2024). Short-time aois-based representative scanpath identification and scanpath aggregation. *Behavior Research Methods*, 1–16.

Jarodzka, H., Holmqvist, K., and Nyström, M. (2010). A vector-based, multidimensional scanpath similarity measure. *Proceedings of the 2010 symposium on eye-tracking research and applications*, 211–218.

Jiawei, H., and Micheline, K. (2006). *Data mining: Concepts and techniques*. Morgan kaufmann.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*(3), 241–254.

Judd, T., Ehinger, K., Durand, F., and Torralba, A. (2009). Learning to predict where humans look. *2009 IEEE 12th international conference on computer vision*, 2106–2113.

Laxman, S., and Sastry, P. S. (2006). A survey of temporal data mining. *Sadhana*, *31*, 173–198.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Leuthold, A. C., Langheim, F. J., Lewis, S. M., and Georgopoulos, A. P. (2005). Time series analysis of magnetoencephalographic data during copying. *Experimental brain research*, *164*, 411–422.

Levenshtein, V. I., et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, *10*(8), 707–710.

Li, A., and Chen, Z. (2018). Representative scanpath identification for group viewing pattern analysis. *journal of eye movement research*, *11*(6).

Lien, Y.-C. N., Wu, W.-J., and Lu, Y.-L. (2020). How well do teachers predict students' actions in solving an ill-defined problem in stem education: A solution using sequential pattern mining. *IEEE Access*, *8*, 134976–134986.

Liu, Y., Zhang, Z., Xing, B., Yuan, J., Feng, C., and Zhang, H. (2021). An enhanced arima model for eeg classification. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 226–230.

MacQueen, J., et al. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, *1*(14), 281–297.

Maldonado, M., Dunbar, E., and Chemla, E. (2019). Mouse tracking as a window into decision making. *Behavior Research Methods*, *51*, 1085–1101.

Matsuda, N., and Takeuchi, H. (2014). Frequent pattern mining of eye-tracking records partitioned into cognitive chunks. *Applied Computational Intelligence and Soft Computing*, *2014*(1), 101642.

McGuire, M. P., and Chakraborty, J. (2016). Directional scan path characterization of eye tracking sequences: A multi-scale approach. *2016 Future Technologies Conference (FTC)*, 51–61.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mohammed, S. (2001). Toward an understanding of cognitive consensus in a group decision-making context. *The Journal of Applied Behavioral Science*, *37*(4), 408–425.

Mohammed, S., and Ringseis, E. (2001). Cognitive diversity and consensus in group decision making: The role of inputs, processes, and outcomes. *Organizational behavior and human decision processes*, *85*(2), 310–335.

Mucherino, A., Papajorgji, P. J., Pardalos, P. M., Mucherino, A., Papajorgji, P. J., and Pardalos, P. M. (2009). K-nearest neighbor classification. *Data mining in agriculture*, 83–106.

Needleman, S. B., and Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, *48*(3), 443–453.

Neisser, U. (2014). *Cognitive psychology: Classic edition*. Psychology press.

Ożdżyński, P., and Zakrzewska, D. (2017). Using frequent pattern mining algorithms in text analysis. *Information Systems in Management*, *6*.

Paape, D., Vasishth, S., Paape, D., and Vasishth, S. (2022). Is reanalysis selective when regressions are consciously controlled? *Glossa Psycholinguistics*, *1*(1).

Parshina, O., Sekerina, I. A., Lopukhina, A., and von Der Malsburg, T. (2022). Monolingual and bilingual reading processes in russian: An exploratory scanpath analysis. *Reading Research Quarterly*, *57*(2), 469–492.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Pinheiro, J., and Bates, D. (2006). *Mixed-effects models in s and s-plus*. Springer science and business media.

Quinlan, J. R. (2014). *C4. 5: Programs for machine learning*. Elsevier.

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria.

Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

Random, F., and Leo, B. (2001). Random forests. *Mach. Learn*, *45*(1), 5–32.

Rasamoelina, A. D., Adjailia, F., and Sinčák, P. (2020). A review of activation function for artificial neural network. *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, 281–286.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, *323*(6088), 533–536.

Sakoe, H., and Chiba, S. (1978). Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, *26*(1), 43–49.

Salton, G., Wong, A., and Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, *34*(1), 1–47.

Shmoys, D. B., Tardos, É., and Aardal, K. (1997). Approximation algorithms for facility location problems. *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 265–274.

Stevenson, C., Smal, I., Baas, M., Dahrendorf, M., Grasman, R., Tanis, C., Scheurs, E., Sleiffer, D., van der Maas, H., et al. (2020). Automated aut scoring using a big data variant of the consensual assessment technique: Final technical report.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, *27*.

Tagaris, G., Richter, W., Kim, S.-G., and Georgopoulos, A. P. (1997). Box-jenkins intervention analysis of functional magnetic resonance imaging data. *Neuroscience research*, *27*(3), 289–294.

Ulitzsch, E., He, Q., and Pohl, S. (2022). Using sequence mining techniques for understanding incorrect behavioral patterns on interactive tasks. *Journal of Educational and Behavioral Statistics*, *47*(1), 3–35.

Von der Malsburg, T., and Vasishth, S. (2011). What is the scanpath signature of syntactic reanalysis? *Journal of Memory and Language*, *65*(2), 109–127.

Von der Malsburg, T., and Vasishth, S. (2013). Scanpaths reveal syntactic underspecification and reanalysis strategies. *Language and Cognitive Processes*, *28*(10), 1545–1578.

Wang, J., and Dong, Y. (2020). Measurement of text similarity: A survey. *Information*, *11*(9), 421.

Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, *78*(10), 1550–1560.

Xu, J., Jiang, M., Wang, S., Kankanhalli, M. S., and Zhao, Q. (2014). Predicting human gaze beyond pixels. *Journal of vision*, *14*(1), 28–28.

Yu, W., Zhao, F., Ren, Z., Jin, D., Yang, X., and Zhang, X. (2023). Mining attention distribution paradigm: Discover gaze patterns and their association rules behind the visual image. *Computer Methods and Programs in Biomedicine*, *230*, 107330.

Part II

Publications

JOURNAL OF
Creative Behavior
Publication of the Creative Education Foundation™

PHILIP BUCZAK
HE HUANG
BORIS FORTHMANN
PHILIPP DOEBLER

The Machines Take Over: A Comparison of Various Supervised Learning Approaches for Automated Scoring of Divergent Thinking Tasks

ABSTRACT

Traditionally, researchers employ human raters for scoring responses to creative thinking tasks. Apart from the associated costs this approach entails two potential risks. First, human raters can be subjective in their scoring behavior (inter-rater-variance). Second, individual raters are prone to inconsistent scoring patterns (intra-rater-variance). In light of these issues, we present an approach for automated scoring of Divergent Thinking (DT) Tasks. We implemented a pipeline aiming to generate accurate rating predictions for DT responses using text mining and machine learning methods. Based on two existing data sets from two different laboratories, we constructed several prediction models incorporating features representing meta information of the response or features engineered from the response's word embeddings that were obtained using pre-trained GloVe and Word2Vec word vector spaces. Out of these features, word embeddings and features derived from them proved to be particularly effective. Overall, longer responses tended to achieve higher ratings as well as responses that were semantically distant from the stimulus object. In our comparison of three state-of-the-art machine learning algorithms, Random Forest and XGBoost tended to slightly outperform the Support Vector Regression.

Keywords: divergent thinking, creative quality, human ratings, supervised learning, Random Forest, gradient boosting, Support Vector Regression.

Scoring of creative thinking tasks is a laborious endeavor that requires human and time resources. Motivated by a potential reduction of scoring efforts automated scoring of creative thinking tests has become a current hot topic in creativity research. This is documented by several published recent attempts for automated scoring of the popular Alternate Uses Task (AUT; e.g., Beaty & Johnson, 2021; Dumas, Organisciak, & Doherty, 2020; Stevenson et al., 2020). Prior work has mostly relied on semantic distance approaches as a way to automatically quantify the originality of the responses. These approaches are examples of unsupervised machine learning approaches and they have shown remarkable success as evidenced by strong correlations at the person-level between semantic distance scores and scores provided by human raters (Beaty & Johnson, 2021; Dumas et al., 2020). However, (semi-)supervised learning approaches have also been successfully implemented (Stevenson et al., 2020). While all of these approaches displayed promising results, there is still room for further improvement, and the current work aims at extending and complementing existing work on automated scoring of the AUT. Specifically, this work focuses on supervised learning approaches which have been less extensively studied in this context, as well as on improved pre-processing of text data. The goal was to examine a supervised learning pipeline in terms of its rating performance using varying supervised learning algorithms (i.e., Random Forest, XGBoost, and Support Vector Regression [SVR]) as well as varying feature set compositions differing in size and interpretability. We further aimed at finding the most relevant features for the prediction of human ratings.

DIVERGENT THINKING AND ITS ASSESSMENT

Divergent thinking (DT) refers to the cognitive capacity to generate multiple options in response to a given task (Guilford, 1967). Hence, the response format is open-ended which naturally implies a scoring of

productivity (i.e., the number of responses). However, this productivity scoring (i.e., fluency) has often been criticized (e.g., Zeng, Proctor, & Salvendy, 2011) for its lack of conceptual relevance with respect to creativity. Creativity commonly refers to something that is perceived as original and useful (Runco & Jaeger, 2012) in the given context. Hence, DT responses should be scored for originality and usefulness beyond mere productivity indices such as fluency. Original responses in DT tasks can be identified based on three classical facets (Wilson, Guilford, & Christensen, 1953): uncommonness, cleverness, and remoteness. Rater instructions often contain explicit instructions to consider all three when scoring originality (e.g., Hofelich-Mohr, Sell, & Lindsay, 2016; Silvia et al., 2008).

Uncommonness refers to the statistical rarity of a response Cropley, 1967; Wallach & Kogan, 1965; Wilson et al., 1953) and is historically perhaps the oldest originality indicator (Hargreaves, 1927). Cleverness refers to the cunning aptness of a response and clever responses are maturely thought through. Cleverness has traditionally been scored by human judges (Forthmann et al., 2017; French et al., 1963; Wilson et al.,1953). Finally, remoteness refers to responses that are associatively more distant as compared to the most obvious responses (Silvia et al., 2008; Wilson et al., 1954). Traditionally, remoteness was scored for the Consequences Task (e.g., generating possible consequences for the scenario that people do not need to sleep anymore) with more far-reaching consequences being more remote than immediate ones (e.g., that people go to work at night).

In addition, remoteness has been conceptualized as semantic distance of responses in comparison to the task stimulus or a semantic representation of it (Dumas & Dunbar, 2014; Hass, 2017). This way originality can be automatically scored by semantic vector models of meaning such as Latent Semantic Analysis (Landauer & Dumais, 1997), GloVe (Pennington, Socher, & Manning, 2014), and Word2Vec (Mikolov, Chen, Corrado, & Dean, 2013). For example, in the AUT, the semantic distance between the item object (e.g., knife) and a response (e.g., use it as a mirror) is calculated to reflect the remoteness of the response. Early validity evidence for scoring based on semantic vector models was quite inconsistent across studies (Forster & Dunbar, 2009; Harbison & Haarman, 2014; Hass, 2017). Elaboration bias (semantic distance depends technically on the number of words; Forthmann et al., 2019) of the most often used LSA approach was identified as one potential source to explain these inconsistent findings.

Newer studies, however, have successfully addressed the issue of elaboration bias and demonstrated strong validity evidence for person-level aggregated scores (e.g., Beaty & Johnson, 2021; Dumas et al., 2020). For example, Beaty and Johnson (2021) improved automatic scoring by relying on a variety of semantic spaces (created by different approaches) as well as multiplicative compositional models instead of additive ones, whereas Dumas et al. (2020) improved automated scoring by a weighting approach that relies on inverse document frequency. Notably, beyond the AUT, semantic distance scoring displayed validity evidence for the Torrance Test of Creative Thinking (Acar et al., 2021), creative verb association (Beaty & Johnson, 2021; Heinen & Johnson, 2018; Prabhakaran, Green, & Gray, 2014), the Consequences Test (LaVoie, Parker, Legree, Ardison, & Kilcullen, 2020), the Remote Associates Test (Beisemann, Forthmann, Bürkner, & Holling, 2020), or abstract figure naming (Sung, Cheng, Tseng, Chang, & Lin, 2022).

SUPERVISED MACHINE LEARNING

Supervised machine learning refers to the construction of predictive models rather than descriptive ones (Lantz, 2013). In such models, an algorithm learns to predict values for a target variable of interest as opposed to unsupervised machine learning models in which the goal is to build a descriptive model of the data, typically with the aim to abstract from the original data and discover structure. For example, the above-mentioned vector models of semantic meaning (e.g., LSA, HAL, GloVe, and Word2Vec) are examples of unsupervised learning algorithms because vector spaces are created towards a quantitative description of how different terms relate semantically to each other and not with the goal to predict any specific target. Supervised models, however, try to approximate an unknown functional relationship between covariates (also called features) and an outcome variable by learning a model from examples. Such functional relationships can originate either from a regression context where the outcome is continuous or from a classification context where the outcome is discrete and input features are mapped to a class label. Here, we consider the prediction of (mean) creative quality ratings as a regression problem.

There is a wide variety of machine learning algorithms differing in their mathematical and conceptual approaches, interpretability, numerical demands, and context of the application. In this work, we focus on Random Forest (Breiman, 2001) and XGBoost (Chen & Guestrin, 2016), because they have shown to be highly performant in a variety of situations with structured tabular data, and on SVR, the regression analog

of the highly competitive and somewhat more well-known Support Vector Machine (Vapnik, 2000) for classification. All three, Random Forest, XGBoost, and SVR, are known to rely on custom feature engineering for peak performance, so we will combine them with different feature sets.

USING SUPERVISED MACHINE LEARNING FOR AUTOMATED SCORING OF DT TASKS

Most of the recent attempts to automatically score DT tasks for originality rely on unsupervised learning. While the current work aims at extending the existing work on automatic scoring of DT tasks it should be noted that very few supervised learning approaches exist in the literature. First, there is the pioneering work by Paulus, Renzulli, and Archambault Jr. (1970) that has been neglected until nowadays. They used a sample of $N = 100$ participants to develop prediction equations for fluency, flexibility, and originality of the responses based on features of the response such as number of question marks, number of commas, number of periods, number of words, number of sentences, average word length, the standard deviation of word length, for example (for complete feature lists for all activities see Paulus et al., 1970). Prediction equations were developed based on stepwise multiple regression analysis and cross-validated on a sample of $N = 53$ participants. Their cross-validation results were quite impressive with correlations between automatically derived scores and human rater scores in the range from .42 to .96, and most of the correlations reported exceeded .70. More than 50 years later these findings are still impressive and promising as the computational power and availability of powerful algorithms have clearly increased over the last five decades. A more recent approach for a (semi-)supervised learning algorithm to score DT tasks has been developed by Stevenson et al. (2020). Based on the word embeddings (WE) derived from Word2Vec models, they cluster AUT task responses w.r.t. their semantic distance first. For each cluster, a representative mean creativity score is obtained by averaging all cluster-specific ratings. New responses are then assigned the creativity score of the semantically nearest cluster. Thus, Stevenson et al. (2020) employ a hybrid, semi-supervised approach that combines unsupervised learning (through clustering) and supervised learning (deriving predictions from observed examples). Their reported validity evidence was clearly on par with the unsupervised approaches proposed by Beaty and Johnson (2021) as well as Dumas et al. (2020).

AIM OF THE CURRENT STUDY

The aim of our study is to develop and evaluate an ML-based approach for automated scoring of DT tasks. In contrast to Stevenson et al.'s (2020) hybrid approach, our approach stems more from the traditional perspective in ML. More precisely, we engineer custom features from original DT data sets and employ frequently used machine learning algorithms, namely Random Forest, XGBoost, and SVR, to predict mean creativity ratings. We aim to compare the predictive performance of these three algorithms, and study the impact of different feature sets (including the influence of individual features) and semantic spaces to derive conclusions and recommendations for practical use cases.

To this end, we follow the general structure of the pipeline depicted on the right-hand side of Figure 1. Whereas in the classical human-based approach (left-hand side), human raters base their ratings directly on the raw response data, our ML-based approach requires further data processing. This usually includes some form of pre-processing, for example, removing spacing, punctuation, or encoding errors. Once the responses are properly pre-processed, they are used to generate a set of informative features. The resulting data set is then used to train the machine learning model. This step can also encompass some sort of model selection, for example, comparing different learners, performing feature selection, or hyperparameter optimization. The final model can then be used to obtain predictions for new data points.

METHODS

DATA AND FEATURE ENGINEERING

For our analysis, we used two DT data sets by Silvia et al. (2008) and Hofelich-Mohr et al. (2016), both containing rated responses from AUTs. While the former data set contains 3,432 responses to alternative uses of *brick* and *knife* rated by three human judges, the latter contains 3,870 responses to alternative uses of *brick* and *paperclip* rated by four human judges. In both cases, ratings ranged from 1 (worst) to 5 (best). For Silvia et al. (2008), inter-rater reliability was .48 (based on an absolute agreement intra-class correlation coefficient for average ratings), and for Hofelich-Mohr et al. (2016) it was .66. According to Cicchetti's (2001) criteria, the obtained levels of interrater reliability were *fair* and *good*, respectively.

Data formatting and sanitization (e.g., removing errors in spacing, punctuation, encoding, etc.) was not necessary for both data sets, because they were already pre-processed in the original source. However, text

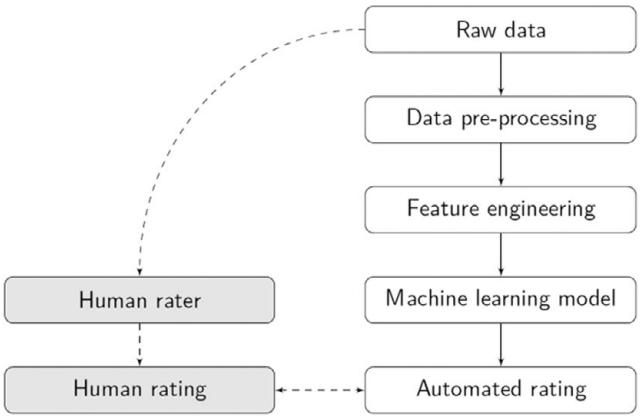

FIGURE 1. Pipeline for automated scoring of DT tasks using machine learning.

formatting and sanitization is generally an important pre-processing step. For the purpose of our analysis, we averaged the respective ratings for each observation. Thus, a baseline observation contained the participant's response, the respective stimulus item, and the response's mean rating. We generated further features which are displayed in Figure 2. The different features can be grouped based on their type and interpretability.

The feature group "meta-features" consists of the features "number of words," "average word length," and "maximum word length". These can be obtained from the respective response and are handily interpretable. The WE describe the word vector representation of a response w.r.t. to a (pre-trained) semantic space. Based on the dimensionality d of the semantic space, d features are added, each representing a word vector space component "loading". Word embeddings are not interpretable by themselves. Generally, pre-

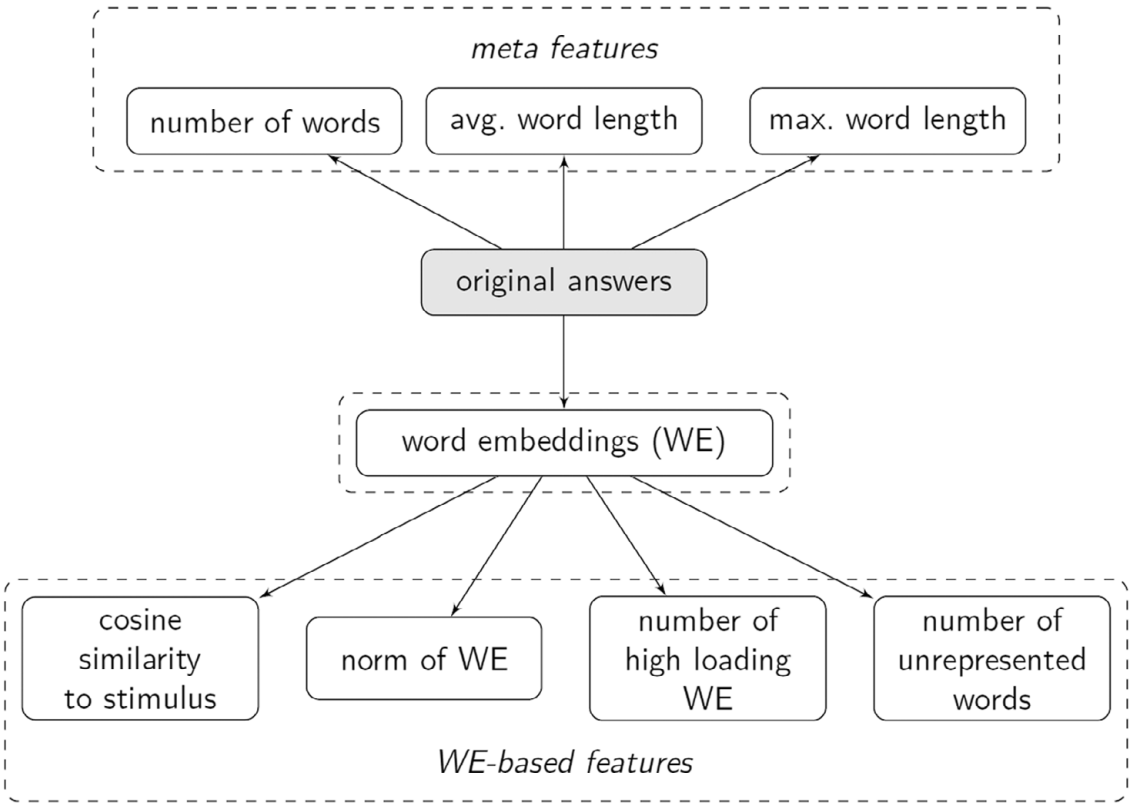

FIGURE 2. Features engineered from original answers.

trained semantic spaces provide vector representations for single words. However, responses in DT tasks are usually phrases or sentences. To obtain sentence embeddings, we sum the WE of the individual words contained in the sentence.

The feature group "WE-based features" contains the similarity of the response to the stimulus item (based on the cosine distance of the respective WE), the norm of the response's WE vector, the number of a response's "high" loading WE components (value larger than the 75%-quantile) and the number of words from the response that could not be represented using the respective semantic space (corpus missings). These features appear interpretable but one must keep in mind that they are based on uninterpretable WE.

MACHINE LEARNING ALGORITHMS

Decision trees

A major class of supervised machine learning algorithms is decision tree models such as Breiman's classification and regression trees (CART; Breiman, Friedman, Stone, & Olshen, 1984). Decision trees split the feature space into disjoint regions in which the target function is predicted through a constant value. For regression trees, the constant value is chosen as the mean observed outcome of all observations falling into the region. Decision trees can be thought of as a tree-like flow chart in which a series of yes/no decisions (splits) leads to a prediction for the outcome. An exemplary decision tree is depicted in Figure 3. In this example, an answer containing two words with an average length of five would lead to a predicted mean creativity rating of 2.5. In the CART algorithm, the splits are chosen such that the total variance of the observations falling into the two subsequent regions is minimized. Although highly interpretable, single tree models suffer from high variability as little perturbations in the data may already lead to distinctly different tree structures. Further, single tree models are often insufficient to capture complicated functional relationships, and thus, several tree models are often combined to form so-called ensembles.

Random Forest

Random Forests (RF; Breiman, 2001) are an ensemble method comprised of a large number of decision trees whose individual predictions are combined into an overall prediction. The RF algorithm grows its trees independently on bootstrap samples of the original data (Efron & Tibshirani, 1993). The RF prediction is then determined (in a regression context) by averaging over all individual tree predictions. It can be shown that the variance of the RF prediction depends on the pairwise correlation of the individual trees as well as the number of trees (Hastie, Tibshirani, & Friedman, 2009). The greater the number of trees and the smaller the pairwise correlation, the smaller the variance term is. Thus, the number of trees is usually chosen large (i.e., several hundred). As for the pairwise correlation, the RF algorithm aims to de-correlate its trees by restricting the number of split variables considered inside the individual tree models and instead choosing a random subset of potential split variables. In this sense, a forest of individual trees with a restricted set of variables is grown. Thus, the RF reduces the high variability of individual tree models overall, but at the same time loses the high interpretability of single trees as its prediction is based on aggregating several hundred trees.

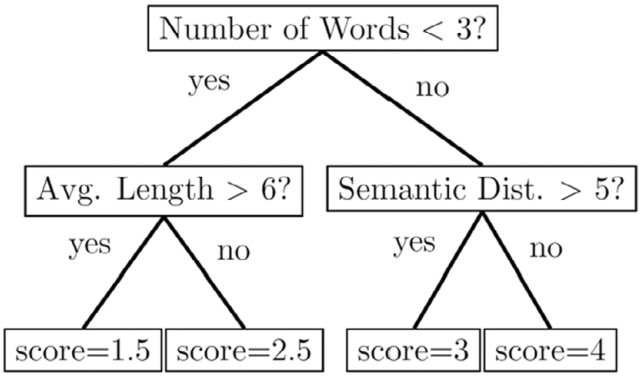

FIGURE 3. An example of a single decision tree.

Gradient Boosting Decision Trees

Gradient Boosting Decision Trees (GBDT; Friedman, 2001) are another ensemble method based on decision trees. Similar to RFs, GBDT reduce the high variability of single tree models but offers only limited interpretability. Different from RFs, however, GBDT models are generated in a stepwise fashion aimed at reducing the mismatch between prediction and data by iteratively fitting decision trees to the discrepancy between prediction and data. A large number of trees can make the discrepancy arbitrarily small but increase the risk of overfitting, that is, the model would resemble the training data too closely and fail to successfully predict unknown future data (Hastie et al., 2009). In this work, we use XGBoost (XGB; Chen & Guestrin, 2016) which is a highly performant and fast implementation of GBDT adding various regularization techniques to reduce the risk of overfitting.

Support Vector Regression

In contrast to regular least squares regression which optimizes a squared loss, the SVR (Vapnik, 2000) uses the so-called ϵ-insensitive loss (see Appendix A) which only penalizes errors of magnitude greater than a pre-specified ϵ and ignores smaller errors. The difference between these two loss functions is visualized in Figure A1 (see Appendix A). Geometrically speaking, the SVR aims to place a tube in the data space that encloses as many data points as possible (i.e., minimizing the ϵ-insensitive loss) while at the same time penalizing too wide tube diameters. SVR can also be used to model non-linear functions by transforming the observations into higher dimensional spaces using non-linear mappings and finding an optimal solution there. To avoid costly computations in high-dimensional spaces, so-called kernel functions are used to perform the required calculations in the original (lower-dimensional) space rather than the transformed space which can greatly reduce the computational complexity. We refer to Vapnik (2000) for examples of such kernel functions.

DATA ANALYSIS

We implemented our pipeline in R (R Core Team, 2020). Machine learning experiments were conducted with the mlr (Bischl et al., 2016) interface using the RF, SVR, and XGB implementations from the ranger (Wright & Ziegler, 2017), e1071 (Meyer, Dimitriadou, Hornik, Weingessel, & Leisch, 2020), and xgboost (Chen et al., 2020) package, respectively. For our analysis, we considered four different feature sets varying in size and interpretability as well as five different semantic spaces.

Table 1 provides an overview of the feature sets. Out of these, only the second feature set does not depend on the WE. Containing only the number of words, average word length, and maximum word length, it is the simplest and most interpretable feature set, in stark contrast to the first feature set which only contains the WE and is, thus, completely uninterpretable. The third feature set adds further features which, however, are only interpretable to a certain degree since they are based on the WE. Feature Set 4 combines all features, i.e. meta information, WE-based features, and the WE.

The pre-trained semantic spaces we used for our analysis differed in word count and dimensionality. The word count refers to the number of words for which the semantic space contains vector representations. The dimensionality of the semantic space refers to the length of the word vector representations. The semantic spaces we used, are further specified in Table 2. Four were originally obtained using the GloVe algorithm (Pennington et al., 2014), while the fifth was obtained through the Word2Vec method (Mikolov et al., 2013). For determining the WE of the answers, we considered additive composition and zero padding in our analysis.

TABLE 1. Feature Sets Used for Analysis. Cf. Figure 2 for the Explanation of Feature Groups

Feature set	Included feature groups
1	WE
2	Meta information
3	Meta information + WE-based features
4	Meta information + WE-based features + WE

Note. WE, Word embeddings.

TABLE 2. Semantic Spaces Used for Analysis

Algorithm	Words	Dimensionality
GloVe	400,000	50
GloVe	400,000	100
GloVe	400,000	300
GloVe	2,000,000	300
Word2Vec	3,000,000	300

Our analysis approach was 2-fold. Comparison Study 1 compared the predictive performance of RF, SVR, and XGB models for the presented data sets. The aim of Comparison Study 2 was to analyze the generalizability of the ML models using "external" data for validation. For this purpose, we used one of the data sets for training (i.e., fitting) the models and the other data set, respectively, for validation. Since the data sets for training and validation originated from different data generating processes, the performance results gave an indication of how well these models could generalize.

All three ML algorithms used in our analysis have an individual set of hyperparameters that require tuning. Thus, we combined our model training with hyperparameter tuning. Table A1 (see Appendix A) contains an overview of the hyperparameter sets and respective search spaces used for tuning. To achieve an honest comparison between our algorithms in Comparison Study 1, we employed a nested resampling approach using 5-fold cross-validation (CV) in the outer validation and a 3-fold CV in the inner hyperparameter tuning loop. This is needed because tuning and validating the same data instances would lead to overly optimistic error estimates (Cawley & Talbot, 2010). For Comparison Study 2, we performed the hyperparameter tuning via a 10-fold CV on the training data and used the optimal parameter choices for fitting the respective models on the entire training data set.

RESULTS

COMPARISON STUDY 1

Figure 4 shows the RMSE values achieved in 100 replications of nested resampling for a given combination of data set, learner, and feature set with an additive composition of WE. Because the choice of semantic space did not have noticeable impact on the performance, we are only showing the results for the semantic space GloVe 6B 50d. We refer to the Online Supplement for the results for the remaining semantic spaces. As a benchmark, we have provided the mean RMSE obtained when using the mean rating computed from the training data set as a constant prediction for all responses in the validation data set.

For both data sets, the predictive performance was worst when relying only on meta information from the answer. Including the information contained in the WE either indirectly (through WE-based features) or directly, greatly improved the performance. The best RMSE values were generally reached by using all features combined. Comparing the three ML learners, the RF models seemed to perform best in most of the scenarios tying with XGB in some cases. There is no scenario in which the SVR achieved the lowest mean RMSE.

These results are echoed when looking at the correlation between automated predictions and the respective human ratings on a response level as shown in Figure 5. The highest response-level correlations scores were achieved with the combination of all features using RF ranging from .71 to .73 on the Hofelich-Mohr et al. (2016) data set and from .55 to .57 on the Silvia et al. (2008) data set.

We obtained similar results when computing correlations on the person-level. Figure 6 shows that RF using the combined feature set performs best, achieving scores from .72 to .75 for the Hofelich-Mohr et al. (2016) data and from .58 to .65 for the Silvia et al. (2008) data.

COMPARISON STUDY 2

Figure 7 shows the RMSE values achieved in the cross-sample analysis. As in Comparison Study 1, the choice of semantic space did not have a noticeable impact on the performance results. When using the Hofelich-Mohr et al. (2016) data for model training and the Silvia et al. (2008) data for validation (A), the ML models could not outperform the mean RMSE achieved when using the mean rating from the training data as the prediction for the validation data. Thus, the ML models did not generalize well in this case. Only the SVR could improve upon the benchmark when using the WE as feature. In contrast, when using the

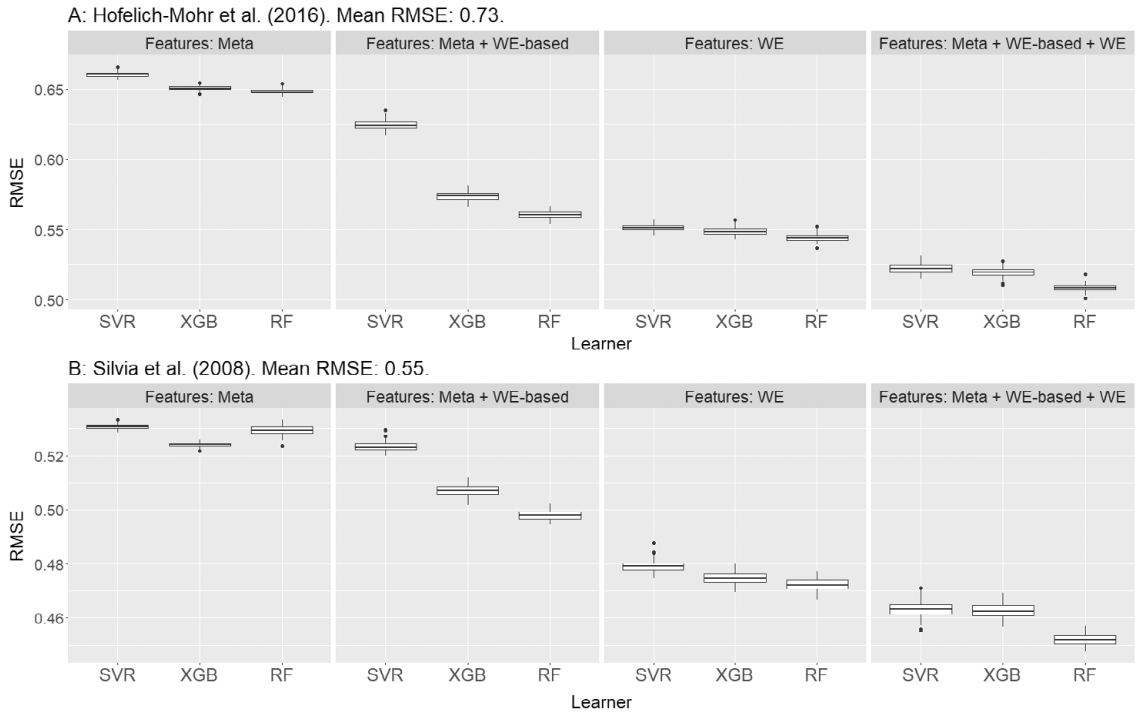

FIGURE 4. Nested resampling RMSE values for Hofelich-Mohr et al. (2016) data (a) and Silvia et al. (2008) data (b).

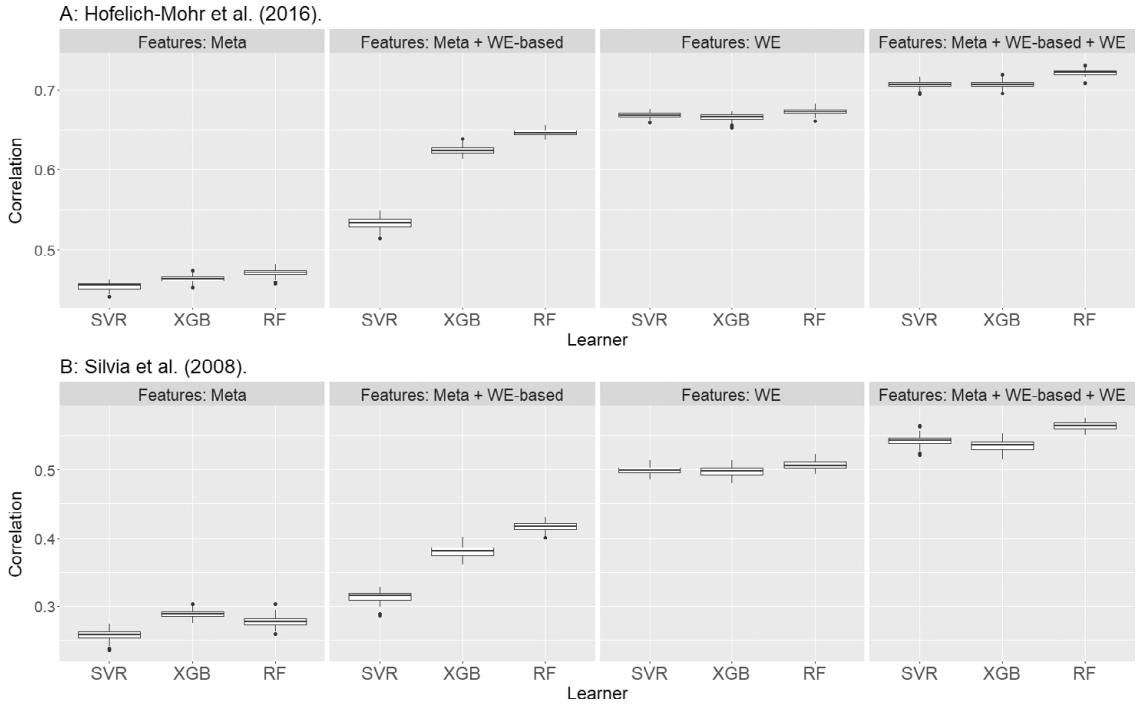

FIGURE 5. Nested resampling response-level correlations for Hofelich-Mohr et al. (2016) data (a) and Silvia et al. (2008) data (b).

Hofelich-Mohr et al. (2016) data for model training and the Silvia et al. (2008) data for validation (B), the ML models outperformed the benchmark mean RMSE in all scenarios indicating improved generalizability. The lowest RMSE values were reached by XGBoost using the combined feature set.

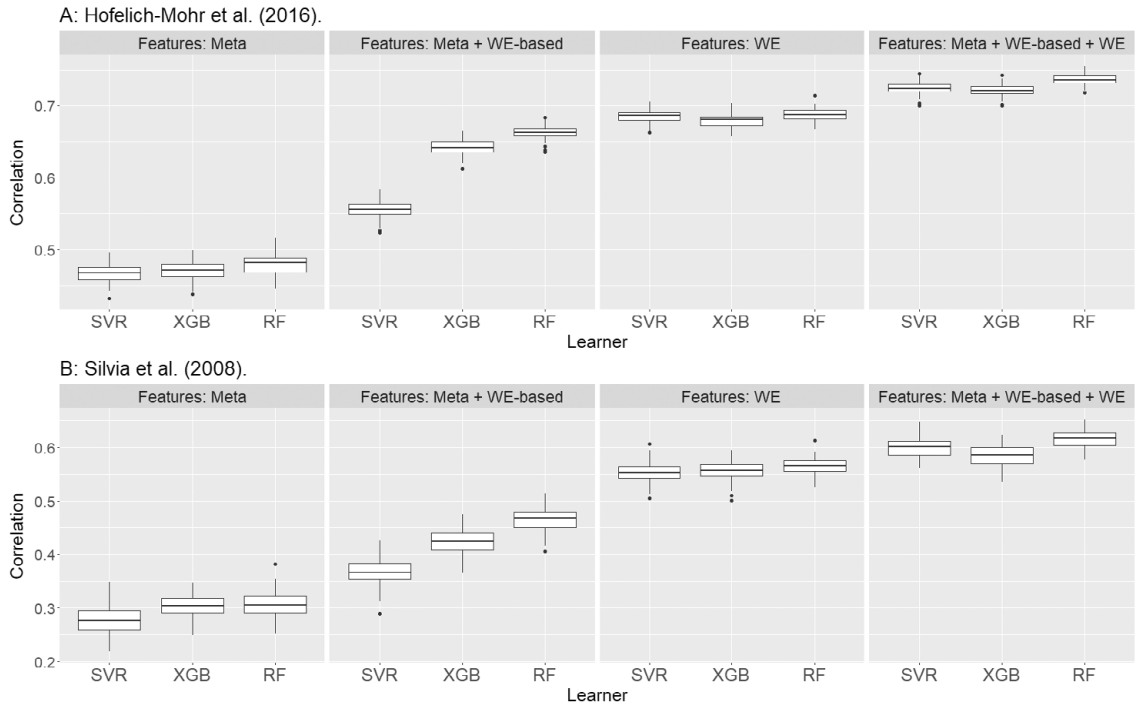

FIGURE 6. Nested resampling person-level correlations for Hofelich-Mohr et al. (2016) data (a) and Silvia et al. (2008) data (b).

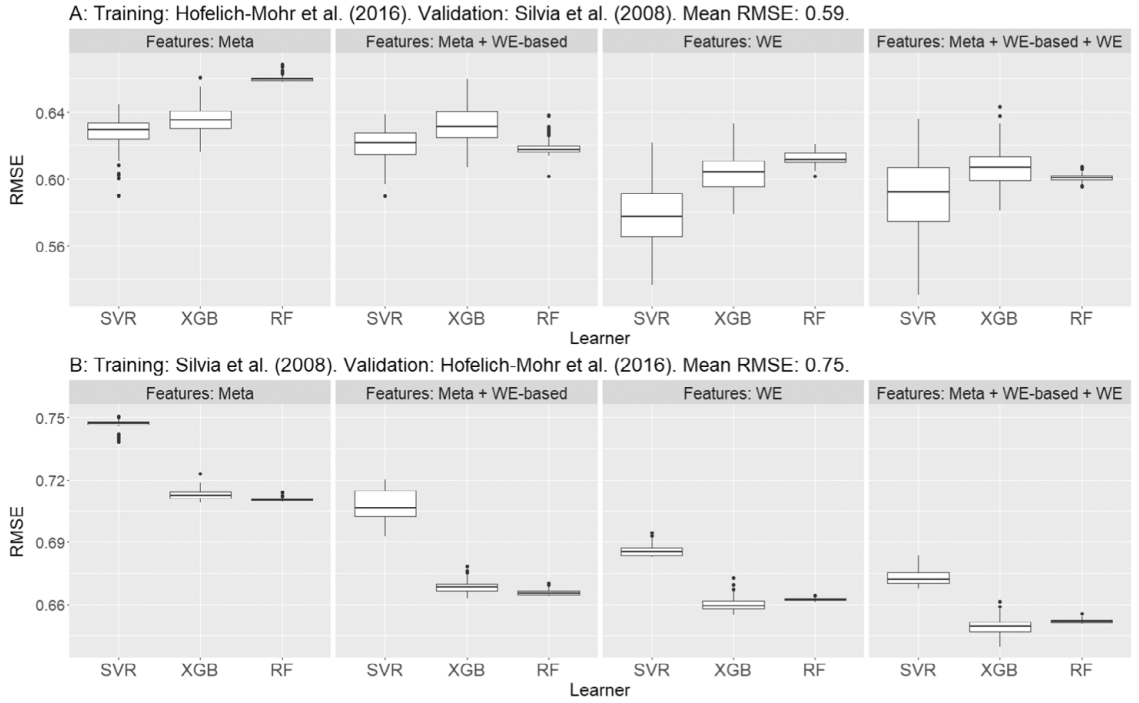

FIGURE 7. Cross-sample RMSE values.

Regarding the response-level correlations (Figure 8), the three ML performed similarly well once WE were included as features. The highest response-level correlation scores were achieved with the combined feature set. When validating the Silvia et al. (2008) data (A) RF reached correlation scores between .57 and

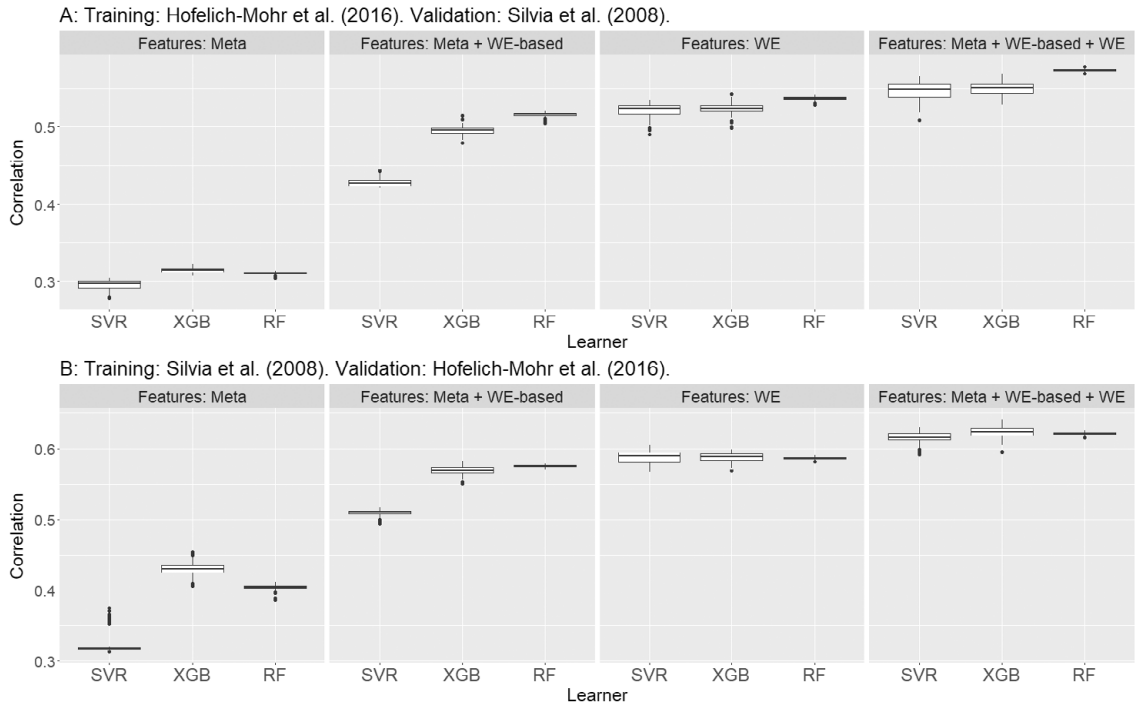

FIGURE 8. Cross-sample response-level correlations between predicted and observed mean ratings.

.58. Interestingly, despite performing best w.r.t. to the RMSE, the SVR achieved lower mean correlations than the RF and XGB models. When using the Hofelich-Mohr et al. (2016) data for validation (B), XGBoost achieved the highest correlation scores ranging between .59 and .64.

On the person-level (Figure 9), RF and SVR performed similarly when validating the Silvia et al. (2008) data (A) using the combined feature set. RF reached correlation scores of .76 to .78. When validating the data from Hofelich-Mohr et al. (2016), RF achieved correlation scores between .59 and .60.

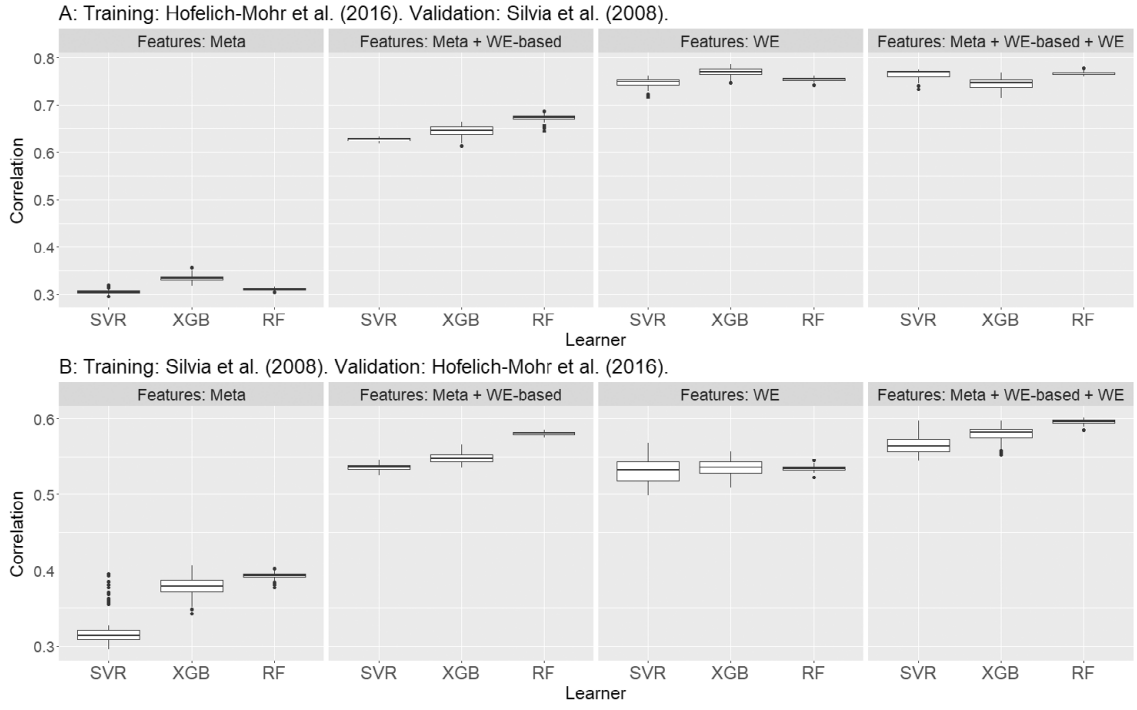

FIGURE 9. Cross-sample correlations between predicted and observed mean ratings on the person-level.

FEATURE IMPORTANCE AND PARTIAL DEPENDENCE

To analyze how predictive individual features were, we computed variable importance scores for RF and XGB models trained on the Hofelich-Mohr et al. (2016) data (we observed similar findings when using the data from Silvia et al., 2008). Figure 10 shows the feature importance of the seven meta and WE-based features as well as the seven most important features for the combined feature set.

For both, RF and XGB, the stimulus similarity was the most important feature followed by the number of words. As for the combined feature set, the first seven dimensions of the WE (d1–d7) were the most important features in RF and XGB models. This seems plausible as the WE used here are the results of a Principal Component Analysis (PCA) of the original WE (see Raunak, Gupta, & Metze, 2019). The deeper mathematical reason is that the first components of a word vector space, by construction, explain more variance in the semantic space than components with a higher index.

To explore how some of these features affected the prediction, we calculated the partial dependence which indicates how the predictions partially depend on the values of the features (see Friedman, 2001). Figure 11 shows the partial dependence of the most important two (interpretable) features, that is, the stimulus similarity and the number of words. We obtained similar results for RF and XGB. As seems intuitive, the mean rating increases with the number of words used in response and decreases the more similar the response and the stimulus object become.

DISCUSSION

We compared the predictive performance of three ML algorithms, that is, RF, XGB, and SVR for the automated prediction of mean creative quality ratings in DT tasks (Guilford, 1967). These algorithms were embedded within a pipeline which also encompassed the generation of meaningful features from the original data. The features generated ranged from interpretable meta information, for example, number of words or maximum word length, to uninterpretable features such as each response's WE into a semantic space. The semantic spaces used in this work were pre-trained using GloVe (Pennington et al., 2014) or Word2Vec (Mikolov et al., 2013). In each case, the WE added several hundred potential covariates that were either used as features themselves or as a source of generating further features such as the cosine similarity between the response and the stimulus object, the WEs' norm, or the number of high loading WE components.

Our analysis showed mostly subtle differences between the ML algorithms. In most cases, RF and XGB tied for the best performance while slightly outperforming the SVR. When working with a single data source, all three algorithms significantly outperformed the RMSE benchmark for predicting the mean rating

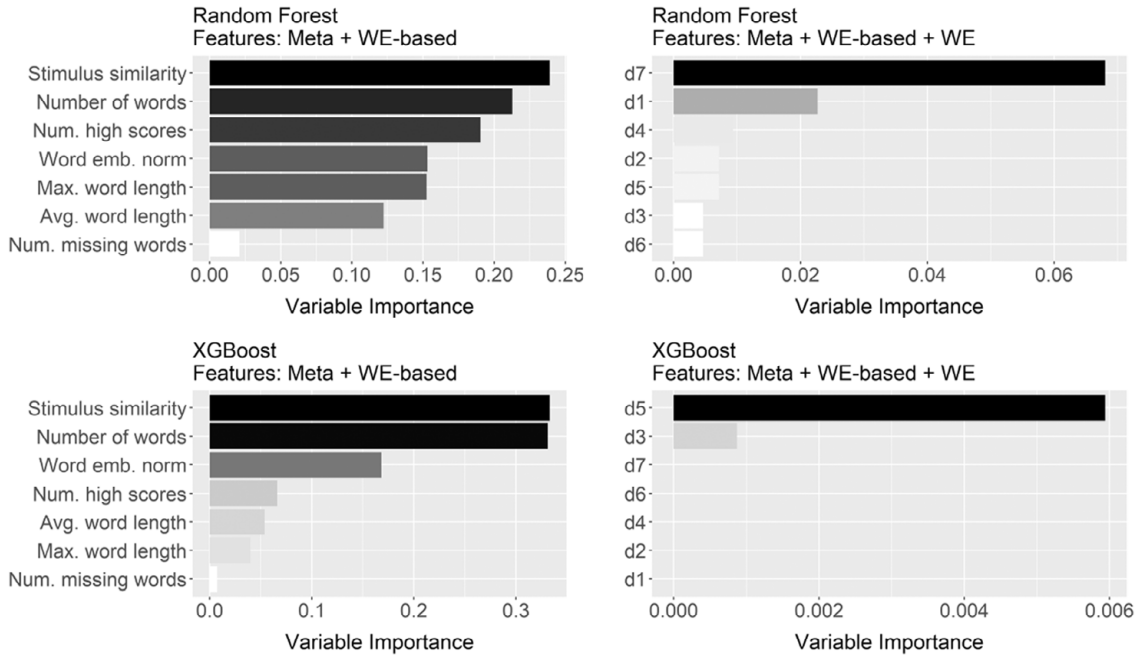

FIGURE 10. Variable importance for RF and XGB models using different feature sets.

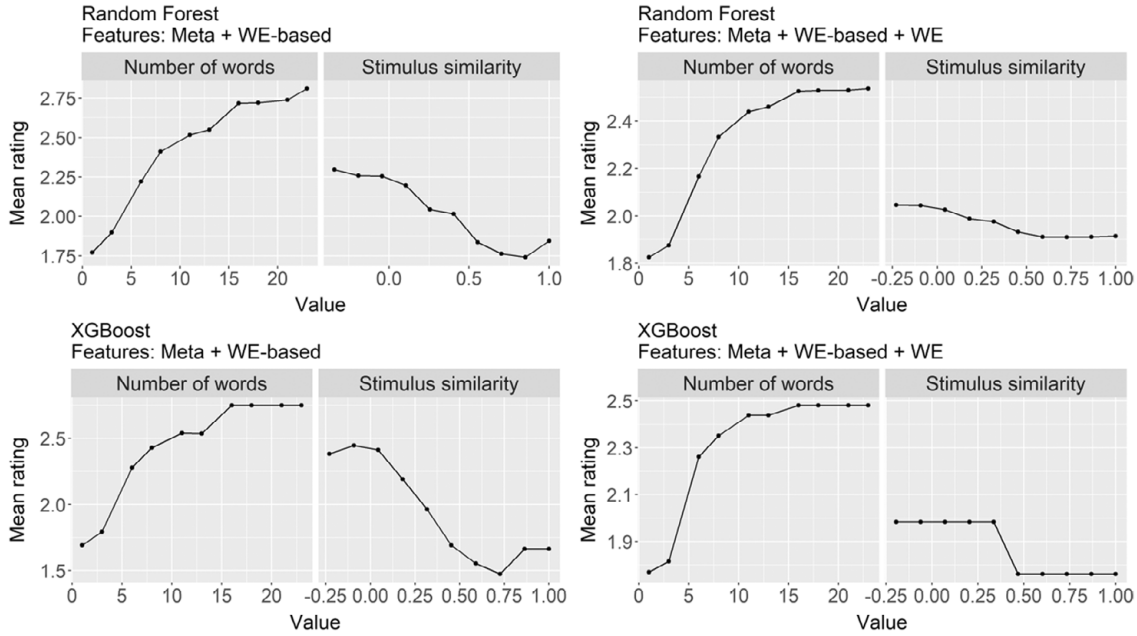

FIGURE 11. Partial dependence of stimulus similarity and number of words using RF and XGB models.

from the training set. The best model, an RF using meta information, WE as well as WE-based features, achieved person-level correlations from .72 to .75 for the Hofelich-Mohr et al. (2016) data and .58 to .65 for the Silvia et al. (2008) data. Correlations of similar size were observed cross-sample, with large linear associations of predictions from one sample to the other.

While the cross-sample correlations were satisfying, the results were not clear-cut for predicting the original rating in terms of the RMSE: When training on the Silvia et al. (2008) data and validating on the Hofelich-Mohr et al. (2016) data, all models handily outperformed the benchmark of predicting the training sample mean. Once the training/validation order is reversed, that is, when training on the Hofelich-Mohr et al. (2016) data set and validating on the Silvia et al. (2008) data set, almost all models perform worse than simply predicting the training sample mean. This indicates that the models did not generalize well to another sample in these cases, and the deeper reason behind this was mean rating differences. The qualitative differences between the RMSE and correlations are explained as follows: the means of average human ratings are hard to predict cross sample, introducing a bias from the point of view of the original ratings. Mathematically, the (square of the) bias is part of the RMSE, explaining why the simple baseline model predicting the training sample mean is competitive in terms of RMSE. In combination with the positive correlations of moderate to large size, this suggests that differences in how the 5-point Likert scales are used by raters are at the core of the cross-sample prediction challenges. Indeed, rater means for the brick stimulus vary substantially (Hofelich-Mohr et al., 2016: $M_1 = 2.29$, $M_2 = 2.58$, $M_3 = 1.64$, $M_4 = 1.61$, pooled $M = 2.03$; Silvia et al., 2008: $M_1 = 1.89$, $M_2 = 1.08$, $M_3 = 2.13$, pooled $M = 1.70$). The differing rating behaviors are further visualized in Figure A2 (see Appendix A). One way to reduce the mean shift problem is to change the prediction task: When first z-standardizing the (training) sample, the (test) sample mean is (approximately) zero. However, no faithful prediction of the original rater behavior results.

Even when assuming instruction and stimulus were completely identical in our cross-sample analysis of predictive performance, there are two inherent substantial generalizability challenges: The underlying population and the rater training might differ between studies. Both potential differences affect the rater's use of the Likert scale. While this is obvious for effects of training, population differences will for example lead to different observed maximum performance and are likely to change rater's perception of relative differences in latent ability. Relatedly, a drift of population or maturation creates a similar challenge. Any algorithmic approach, including those studied here, cannot generalize well if target population scaling differs. Mapping scales is challenging because identical benchmark responses would need to be available. Empirically, we find some overlap between responses to the brick stimulus in the two data sets, but they are mostly limited to lowly rated ideas (e.g., building a house, usage as a door stopper, or as a weapon). In sum, generalizability

challenges limit the quality of predictions in other samples, but part of the problem is not specific to algorithmic approaches, as human raters trained differently would equally distort ratings in another study. In other words, an algorithmic scoring will implicitly use the scale of the original human raters.

Another potential source of bias related to the raters may be demographic bias, for example, with regard to respondents' sex or race. Demographic variables might be known to raters, and bias ratings, or merely reflected in the choice of wording of otherwise identical ideas. This also ties into the debate of fairness in ML (for a review see e.g., Mehrabi, Morstatter, Saxena, Lerman, & Galstyan, 2022). Since the data used here did not contain any demographic information, we could not analyze this form of bias. Even if rater bias is not as prevalent, an automated rating algorithm may still inherit bias from the semantic space used for obtaining the WEs (Lauscher & Glavaš, 2019).

Regarding the features used, we found that including WE in some form (either directly or indirectly) greatly increased the predictive performance of the models. Models that solely contained the interpretable meta-information features were not competitive in comparison. Thus, there exists some form of trade-off between predictive performance and feature interpretability that needs to be addressed in respective applications. Additional preliminary results from our simulations also suggest that is possible to reduce the dimensionality of feature sets by performing a PCA on the WE matrix and replacing the WE components by a set of principal components. When including the first 50% of the principal components of the WE instead of all WE components themselves, our models achieved similar results as shown in Figure A3 (see Appendix A). Thus, it is possible to further reduce the computational complexity of our models without notably sacrificing predictive performance. For practical purposes, it would be interesting to study how much further the complexity of our models can be reduced before suffering a significant performance loss.

The choice of the semantic space used for generating the WEs did not have an impact on the model performance. However, further preliminary results suggest that the WE composition method may be a potential source of performance gain. In this work, we have used additive composition for determining the WE of sentences, that is, the WE of individual words were simply added up. The downside of this approach is that through the summation syntactic or word-order information of the phrases is lost (Landauer, 2002), and since not all phrases are of equal length, simply stacking the vector representation of all words together would result in different lengths of input vectors for the ML models. A simple solution is the so-called zero padding method, which adds zeros at the end of the short responses to match the length of long responses. The drawback of this method is that the dimensionality of the input vector can become particularly large for a few particularly long responses. Figure A4 (Appendix A) shows that padding improves predictive performance in most cases. One potential explanation is that additive composition entails information loss by aggregation and that several responses could lead to a similar composite. For practical use, however, it must be stressed that padding leads to greatly increased computational effort in time and memory. Therefore, if one is willing to sacrifice a small amount of predictive performance, using additive word composition may already suffice.

Overall, this work serves as proof of concept for automatically predicting creative quality ratings from DT tasks in the spirit of Paulus et al. (1970) leaving open many potential research avenues for future work. A research question of particular practical importance might be analyzing whether or not algorithms for automated predictions can distinguish between responses that would be deemed creative and responses that would be deemed nonsensical by the human rater. This could be seen as a specific instance of so-called "adversarial examples" (Biggio & Roli, 2018), an approach widely used in pattern recognition to gain insights about a model and to understand when predictions of a model tip in an unexpected direction.

The supervised approaches discussed here attempt to algorithmically reproduce human ratings, the current gold standard in AUT scoring. In contrast, unsupervised approaches rely on WEs and related features and are not explicitly optimized to reproduce human ratings, but are found to be correlated (Beaty & Johnson, 2021; Dumas et al., 2020). The semi-supervised approach of Stevenson et al. (2020) relies on distances to clusters learned in an unsupervised manner from a training set and predicts mean cluster ratings without employing regression or prediction models, circumventing the need for extensive parameter tuning. We see several clear methodological increments relative to the conceptually closest semi-supervised approach by Stevenson et al. Next to the cross-sample validation in two large English language samples, our preprocessing pipeline has the potential to improve upon all other existing approaches (unsupervised or semi-supervised). Further, our nested cross-validation framework allows us to easily change the learners (any supervised algorithm could replace the three approaches we study), the evaluation metrics, as well as the cross-sample comparison.

A logical next step is the direct comparison of unsupervised, supervised, and semi-supervised approaches. Toward a fair comparison, measures like the (cross-validated) RMSE potentially favoring supervised approaches would need to be complemented by validity evidence and robustness analysis, including adversarial examples and measures for the ability to handle previously unobserved responses.

The field of creative thinking research is currently on the mission to further improve the automated scoring of tasks such as the AUT. The current work extends and complements the existing body of research by examining supervised learning algorithms and a variety of features. The best-performing algorithms (i.e., RF and XGB) in our work push person-level correlations into the range of previous works using unsupervised or semi-supervised algorithms (Beaty & Johnson, 2021; Dumas et al., 2020; Stevenson et al., 2020), which renders them as promising algorithms to further improve automated scoring in future work (with the take-home message that the RMSE is hard to do well cross-sample because of mean differences). In addition, the studied feature sets look promising, and we have shown that dimensionality reduction approaches can help in reducing model complexity. Overall, we expect that a combination of unsupervised, semi-supervised, and supervised algorithms has the potential to push the correlation between ratings and predictions towards the correlation of two independent groups of raters, with latent variable correlation approaching unity.

CONFLICT OF INTEREST

We have no known conflict of interest to disclose.

AUTHOR CONTRIBUTIONS

The authors made the following contributions. Philip Buczak: conceptualization, formal analysis, methodology, software, visualization, original draft preparation, review, and editing; He Huang: conceptualization, formal analysis, methodology, software, visualization, original draft preparation, review, and editing; Boris Forthmann: conceptualization, funding acquisition, original draft preparation, review, and editing; and Philipp Doebler: conceptualization, funding acquisition, project administration, original draft preparation, review, and editing.

DATA AVAILABILITY STATEMENT

All R scripts used in this work are openly available in a repository of the Open Science Framework (https://osf.io/4sbhn/). Both datasets are also openly available. Data from Hofelich-Mohr et al. (2016) are available at https://conservancy.umn.edu/handle/11299/172116. Data from Silvia et al. (2008) are available at https://osf.io/8vrck/.

REFERENCES

Acar, S., Berthiaume, K., Grajzel, K., Dumas, D., Flemister, C., & Organisciak, P. (2021). Applying automated originality scoring to the verbal form of Torrance tests of creative thinking. *Gifted Child Quarterly*. doi: 10.1177/00169862211061874

Beaty, R.E., & Johnson, D.R. (2021). Automating creativity assessment with SemDis: An open platform for computing semantic distance. *Behavior Research Methods*, *53*(2), 757–780.

Beisemann, M., Forthmann, B., Bürkner, P.-C., & Holling, H. (2020). Psychometric evaluation of an alternate scoring for the remote associates test. *The Journal of Creative Behavior*, *54*(4), 751–766.

Biggio, B., & Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, *84*, 317–331.

Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., ... & Jones, Z.M. (2016). mlr: Machine learning in R. *Journal of Machine Learning Research*, *17*(170), 1–5.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*, 123–140.

Breiman, L., Friedman, J., Stone, C.J., & Olshen, R.A. (1984). *Classification and regression trees*. Boca Raton, FL: CRC Press.

Cawley, G.C., & Talbot, N.L. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, *11*, 2079–2107.

Chen, T., & Guestrin, C. (2016). XGboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). New York, NY: Association for Computing Machinery.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., Cho, H., ... & Li, Y. (2020). *xgboost: Extreme gradient boosting*. R package version 1.2.0.1. Available from: https://CRAN.R-project.org/package=xgboost

Cicchetti, D.V. (2001). Methodological commentary the precision of reliability and validity estimates re-visited: Distinguishing between clinical and statistical significance of sample size requirements. *Journal of Clinical and Experimental Neuropsychology*, *23*(5), 695–700.

Cropley, A.J. (1967). *Creativity*. London, UK: Longmans.

Dumas, D., Organisciak, P., & Doherty, M. (2020). Measuring divergent thinking originality with human raters and text-mining models: A psychometric comparison of methods. *Psychology of Aesthetics, Creativity, and the Arts*, *15*(4), 645–663.

Dumas, D., & Dunbar, K.N. (2014). Understanding fluency and originality: A latent variable perspective. *Thinking Skills and Creativity*, *14*, 56–67; doi: 10.1016/j.tsc.2014.09.003

Efron, B., & Tibshirani, R. (1993). *An introduction to the bootstrap*. Boca Raton, FL: Chapman & Hall/CRC.

Forster, E.A., & Dunbar, K.N. (2009). Creativity evaluation through latent semantic analysis. In N.A. Taatgen & H. van Rijn (Eds.), *Proceedings of the 31th Annual Conference of the Cognitive Science Society* (pp. 602–607). Stroudsburg, PA: Cognitive Science Society.

Forthmann, B., Holling, H., Çelik, P., Storme, M., & Lubart, T. (2017). Typing speed as a confounding variable and the measurement of quality in divergent thinking. *Creativity Research Journal*, *29*, 257–269; doi: 10.1080/10400419.2017.1360059

Forthmann, B., Oyebade, O., Ojo, A., Günther, F., & Holling, H. (2019). Application of latent semantic analysis to divergent thinking is biased by elaboration. *Journal of Creative Behavior*, *53*(4), 559–575.

French, J.W., Ekstrom, R.B., & Price, L.A. (1963). *Manual for kit of reference tests for cognitive factors*. Princeton, NJ: Educational Testing Service.

Friedman, J.H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*(5), 1189–1232.

Guilford, J.P. (1967). *The nature of human intelligence*. New York: McGraw-Hill.

Harbison, H.J.I., & Haarman, H. (2014). Automated scoring of originality using semantic representations. In P. Bello, M. Guarini, M. McShane, & B. Scassellati (Eds.), *Proceedings of COGSCI 2014* (pp. 2327–2332). Austin, TX: Cognitive Science Society.

Hargreaves, H.L. (1927). *The 'factulty' of imagination*. London, UK: Cambridge University Press.

Hass, R.W. (2017). Tracking the dynamics of divergent thinking via semantic distance: Analytic methods and theoretical implications. *Memory & Cognition*, *45*(2), 233–244.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference and prediction* (2nd edn). New York, NY: Springer.

Heinen, D.J.P., & Johnson, D.R. (2018). Semantic distance: An automated measure of creativity that is novel and appropriate. *Psychology of Aesthetics, Creativity, and the Arts*, *12*(2), 144–156.

Hofelich-Mohr, A., Sell, A., & Lindsay, T. (2016). Thinking inside the box: Visual design of the response box affects creative divergent thinking in an online survey. *Social Science Computer Review*, *34*(3), 347–359.

Landauer, T.K., & Dumais, S.T. (1997). A solution to Plato's problem: The Latent Semantic Analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, *104*, 211–240; doi: 10.1037/0033-295X.104.2.211

Landauer, T.K. (2002). On the computational basis of learning and cognition: Arguments from LSA. *Psychology of Learning and Motivation*, *41*, 43–84.

Lantz, B. (2013). *Machine learning with R*. Birmingham, UK: Packt Publishing.

Lauscher, A., & Glavaš, G. (2019). Are we consistently biased? Multidimensional analysis of biases in distributional word vectors. In R.F. Mihalcea (Ed.), *Lexical and Computational Semantics (*SEM) – Proceedings of the Eighth Conference: June 6–7, 2019, Minneapolis: NAACL HLT 2019* (pp. 85–91). Stroudsburg, PA: Association for Computational Linguistics.

LaVoie, N., Parker, J., Legree, P.J., Ardison, S., & Kilcullen, R.N. (2020). Using latent semantic analysis to score short answer constructed responses: Automated scoring of the consequences test. *Educational and Psychological Measurement*, *80*(2), 399–414.

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., & Galstyan, A. (2022). A survey on bias and fairness in machine learning. *ACM Computing Surveys*, *54*(6), 1–35.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2020). *e1071: Misc functions of the department of statistics, probability theory group (formerly: E1071)*. R package version 1.2.0.1. Available from: https://CRAN.R-project.org/pace1071

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv Preprint arXiv:1301.3781*.

Paulus, D.H., Renzulli, J.S., & Archambault, F.X., Jr. (1970). *Computer simulation of human ratings of creativity. Final report*. Storrs, CT: School of Education, University of Connecticut.

Pennington, J., Socher, R., & Manning, C.D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Stroudsburg, PA: Association for Computational Linguistics.

Prabhakaran, R., Green, A.E., & Gray, J.R. (2014). Thin slices of creativity: Using single-word utterances to assess creative cognition. *Behavior Research Methods*, *46*(3), 641–659.

R Core Team. (2020). *R: A language and environment for statistical computing*. Vienna: R Foundation for Statistical Computing.

Raunak, V., Gupta, V., & Metze, F. (2019). Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)* (pp. 235–243). Stroudsburg, PA: Association for Computational Linguistics.

Runco, M.A., & Jaeger, G.J. (2012). The standard definition of creativity. *Creativity Research Journal*, *24*(1), 92–96.

Silvia, P., Winterstein, B., Willse, J., Barona, C., Cram, J.T., Hess, K.I., ... & Richard, C.A. (2008). Assessing creativity with divergent thinking tasks: Exploring the reliability and validity of new subjective scoring methods. *Psychology of Aesthetics, Creativity, and the Arts*, *2*, 68–85.

Stevenson, C., Smal, I., Baas, M., Dahrendorf, M., Grasman, R., Tanis, C., ... & van der Maas, H. (2020). *Automated AUT scoring using a big data variant of the consensual assessment technique: Final technical report*. Psychology Research Institute, University of Amsterdam.

Sung, Y.-T., Cheng, H.-H., Tseng, H.-C., Chang, K.-E., & Lin, S.-Y. (2022). Construction and validation of a computerized creativity assessment tool with automated scoring based on deep-learning techniques. *Psychology of Aesthetics, Creativity, and the Arts*. https://doi.org/10.1037/aca0000450

Vapnik, V.N. (2000). *The nature of statistical learning theory* (2nd edn). New York: Springer.

Wallach, M.A., & Kogan, N. (1965). *Modes of thinking in young children: A study of the creativity-intelligence distinction*. New York, NY: Holt, Rinehart & Winston.

Wilson, R.C., Guilford, J.P., Christensen, P.R. (1953). The measurement of individual differences in originality. *Psychological Bulletin*, *50*(5), 362–370. doi: 10.1037/h0060857

Wilson, R.C., Guilford, J.P., & Christensen, P.R., & Lewis, D.J. (1954). A factor-analytic study of creative-thinking abilities. *Psychometrika*, *19*(4), 297–311.

Wright, M.N., & Ziegler, A. (2017). ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, *77*(1), 1–17.

Zeng, L., Proctor, R.W., & Salvendy, G. (2011). Can traditional divergent thinking tests be trusted in measuring and predicting real-world creativity? *Creativity Research Journal*, *23*(1), 24–37.

Philip Buczak, He Huang, TU Dortmund University

Boris Forthmann, University of Münster

Philipp Doebler, TU Dortmund University

Correspondence concerning this article should be addressed to Philipp Doebler, Department of Statistics, TU Dortmund University, Vogelpothsweg 87, 44221 Dortmund, Germany. E-mail: doebler@statistik.tu-dortmund.de

ACKNOWLEDGMENT

Open Access funding enabled and organized by Projekt DEAL.

AUTHOR NOTE

P.B. and H.H. have contributed equally to this work and are listed here in alphabetical order. The work of P.B. and H.H. is supported by a grant from The Eric & Wendy Schmidt Fund for Strategic Innovation. The authors gratefully acknowledge the computing time provided on the Linux HPC cluster at TU Dortmund University (LiDO3), partially funded in the course of the Large-Scale Equipment Initiative by the German Research Foundation (DFG) as project 271512359.

APPENDIX A

ϵ-INSENSITIVE LOSS

For a pre-specified ϵ, the ϵ-insensitive loss $L\left(|y-\widehat{f}(\mathbf{x})|_\epsilon\right)$ is given by

$$L\left(|y-\widehat{f}(\mathbf{x})|_\epsilon\right) = \begin{cases} 0, & \text{if } |y-\widehat{f}(\mathbf{x})| \leq \epsilon, \\ |y-\widehat{f}(\mathbf{x})|-\epsilon, & \text{otherwise} \end{cases}$$

where y is the observed target value, $\mathbf{x}$ is the input feature vector and $\widehat{f}(\mathbf{x}) = \widehat{y}$ is the predicted target value (Vapnik, 2000).

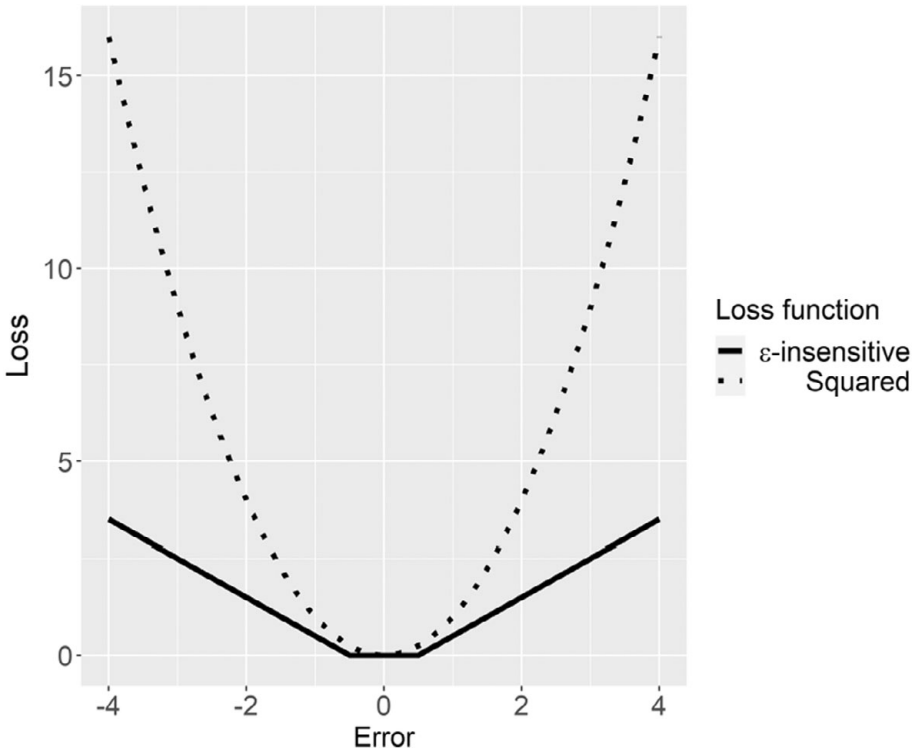

FIGURE A1. Comparison of ϵ-insensitive loss and quadratic loss.

LINEAR SVR OPTIMIZATION PROBLEM

The linear SVR optimization problem is given by finding parameters $\mathbf{w}$ and b such that

$$\Phi\left(\mathbf{w}, \boldsymbol{\xi}_{\mathbf{1}}, \ldots, \boldsymbol{\xi}_{\mathbf{n}}, \boldsymbol{\xi}_{\mathbf{1}}^{*}, \ldots, \boldsymbol{\xi}_{\mathbf{n}}^{*}\right) = \frac{1}{2}(\mathbf{w} \cdot \mathbf{w}) + C \sum_{i=1}^{n}\left(\xi_i + \xi_i^{*}\right)$$

is minimized subject to

$$\begin{cases} y_i - (\mathbf{w} \cdot \mathbf{x}_i) - b & \leq \epsilon + \xi_i, \quad i = 1, \ldots, n \\ (\mathbf{w} \cdot \mathbf{x}_i) + b - y_i & \leq \epsilon + \xi_i^{*}, \quad i = 1, \ldots, n \\ \xi_i, \xi_i^{*} & \geq 0, \quad i = 1, \ldots, n \end{cases}$$

where C is a pre-specified penalization/cost factor and ξ_i, ξ_i^{*} are so-called slack variables for each unit i introduced to ease the optimization process, that is, to allow for observations to lie outside the ϵ-tube (Vapnik, 2000).

TABLE A1. Hyperparameters and Respective Search Spaces

Learner	Hyperparameter	Search space
Random Forest	mtry	{2, . . ., #Features}
	min.node.size	{1, . . ., 10}
	splitrule	{variance, extratrees}
Support Vector Regression	cost	2^x with $x \in [-5, 5]$
	gamma	2^x with $x \in [-5, 5]$
XGBoost	nrounds	{10, . . ., 200}
	max_depth	{1, . . ., 20}
	eta	[0.05, 0.3]
	alpha	[0, 1]
	lambda	[0, 1]
	gamma	[0, 5]

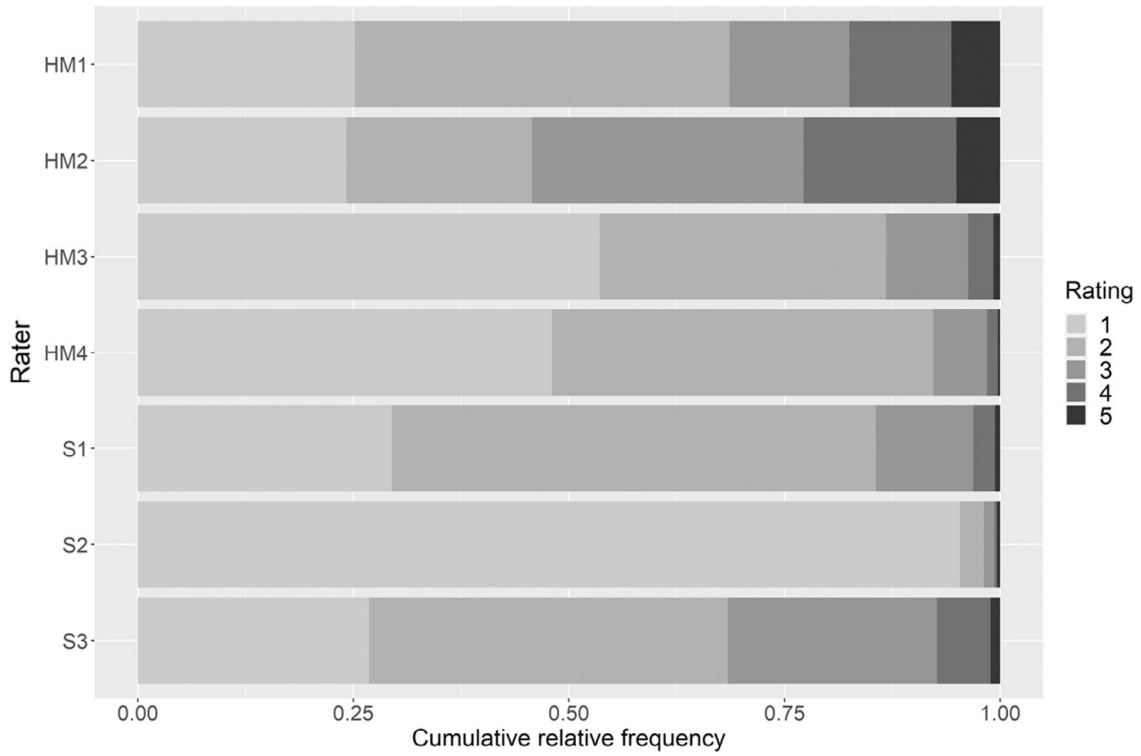

FIGURE A2. Rating behavior for the four raters (denoted by HM1–HM4) from Hofelich-Mohr et al. (2016) and the three raters (denoted by S1–S3) from Silvia et al. (2008).

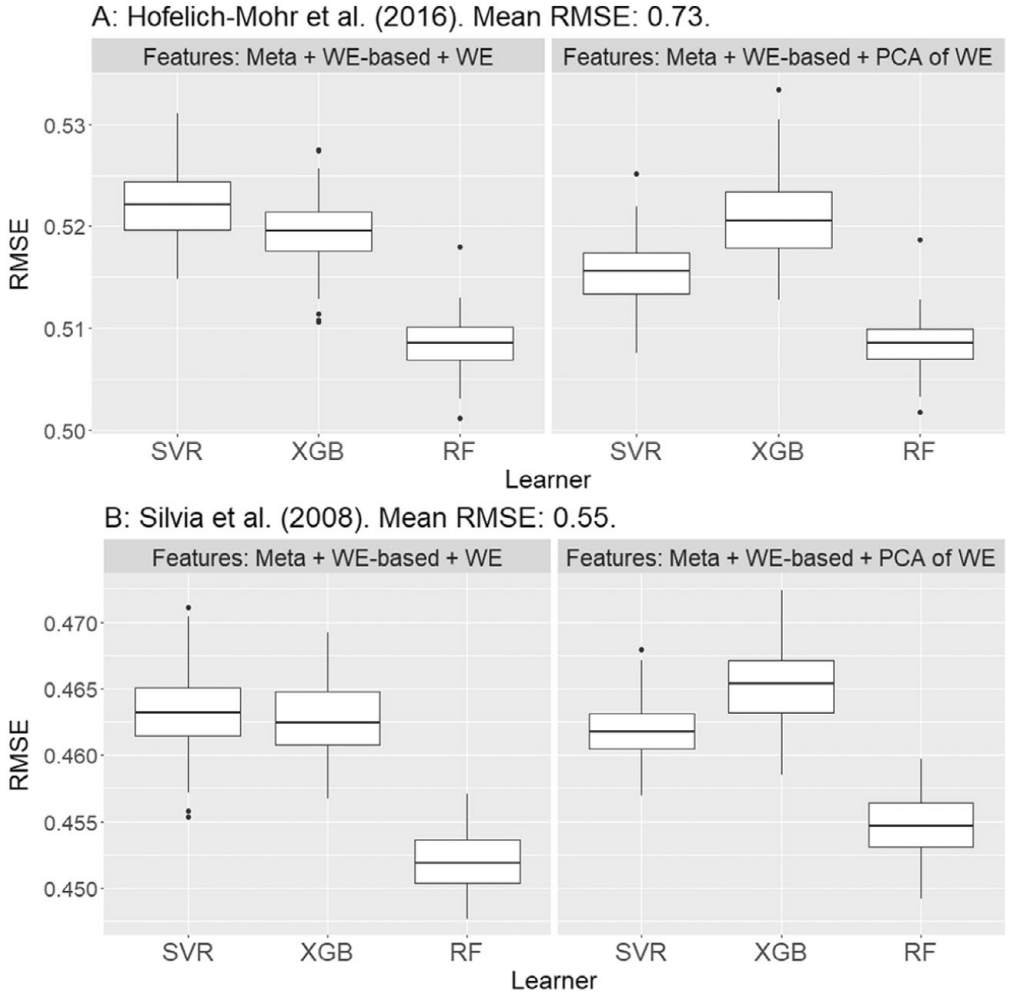

FIGURE A3. Nested resampling RMSE values when including WE or the first 50% principal components of a PCA performed on the WE.

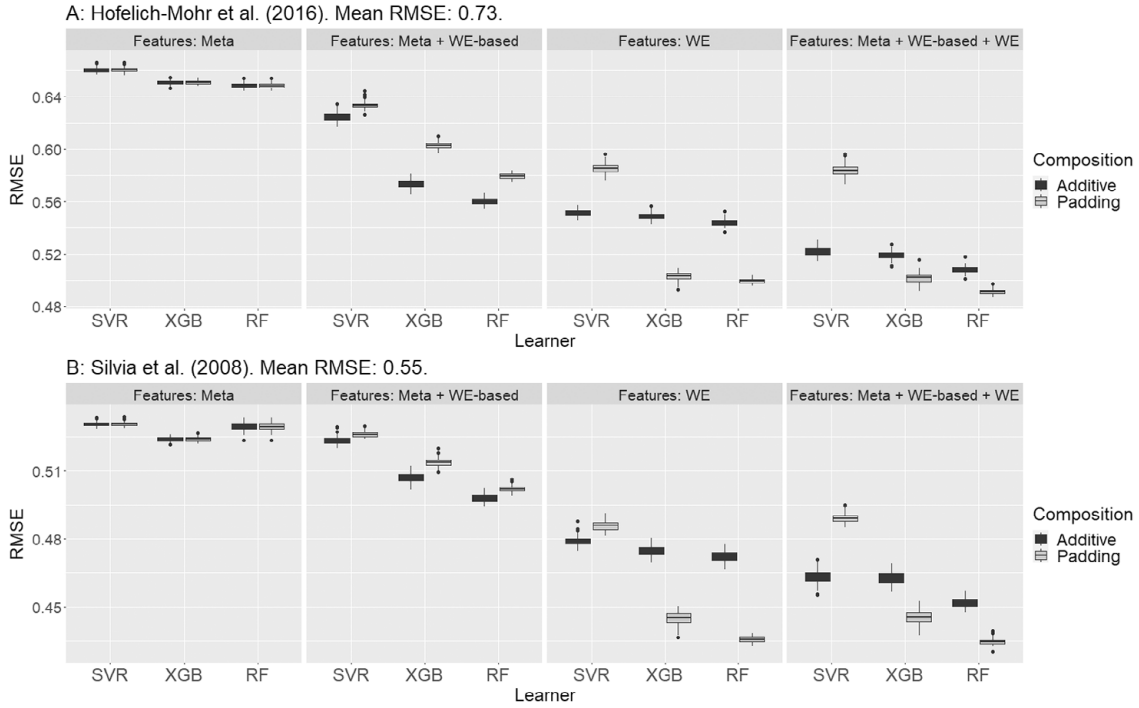

FIGURE A4. Nested resampling RMSE values for additive composition and padding.

Behavior Research Methods
https://doi.org/10.3758/s13428-023-02332-w

ORIGINAL MANUSCRIPT

Short-time AOIs-based representative scanpath identification and scanpath aggregation

He Huang[1] · Philipp Doebler[1] · Barbara Mertins[1,2]

Accepted: 26 December 2023

Abstract
A new algorithm to identify a representative scanpath in a sample is presented and evaluated with eye-tracking data. According to Gestalt theory, each fixation of the scanpath should be on an area of interest (AOI) of the stimuli. As with existing methods, we first identify the AOIs and then extract the fixations of the representative scanpath from the AOIs. In contrast to existing methods, we propose a new concept of short-time AOI and extract the fixations of representative scanpath from the short-time AOIs. Our method outperforms the existing methods on two publicly available datasets. Our method can be applied to arbitrary visual stimuli, including static stimuli without natural segmentation, as well as dynamic stimuli. Our method also provides a solution for issues caused by the selection of scanpath similarity.

Keywords Eye-tracking · Scanpaths · Aggregation · Representative scanpath · Scanpath clustering

Introduction

Eye-tracking has been widely used in applied and scientific research over the past decades. It allows researchers to study the movements of a participant's focus of visual attention during a variety of activities and thus provides insight into the cognitive processes underlying human behaviors (Eckstein et al., 2017; Hartmann & Fischer, 2016; Koć-Januchta et al., 2017; Peterson et al., 2015). Due to the high degree of viewing freedom, one challenge for the researcher is to understand the scanning strategies in a sample. The present paper proposes a novel method to identify a sample-level aggregated scanning strategy using a representative scanpath.

Scanpaths

In many applications, the recordings of the point of regard (POR) provided by the eye-tracking equipment are reduced to scanpaths. A scanpath (see Fig. 1) is the temporal sequence of spatial positions of fixations and saccades. Fixations occur when the gaze pauses over informative regions, and saccades are rapid movements between fixations (Salvucci & Goldberg, 2000). The micromovements within the fixation are classified as noisy low-level oculomotor phenomena and are subdivided into microsaccades, tremor and drift, which are involuntarily produced during fixation periods. This reduction is convenient to minimize the complexity of eye-tracking data while retaining its most essential characteristics in higher-level analyses. Scientists have proposed various ways of identifying fixations. Most of them are based on thresholds of velocity, distance or dispersion, duration, and angle of the POR. The choice of method may depend on the specific research question and the nature of the eye-tracking data being analyzed. In this paper, we focus on the resulting scanpaths.

Areas of interest

As stated in Gestalt theory (Kanizsa, 1979), the elements in bounded areas are perceived as belonging together, so that a single fixation can be located at anywhere of the bounded areas. Therefore, the location of the fixations should be understood in the context of the larger visual scene, namely area of interest (AOIs). As a result, a study of scanpaths should be based on the AOIs that they pass

✉ He Huang
he.huang@tu-dortmund.de

[1] Department of Statistics, TU Dortmund University, 44227 Dortmund, Germany

[2] Departments of Cultural Studies, TU Dortmund University, 44227 Dortmund, Germany

Published online: 09 January 2024

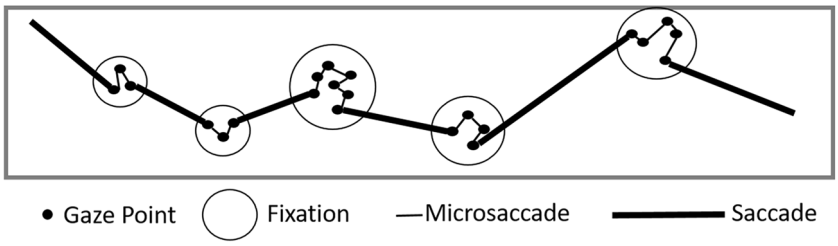

Fig. 1 An example of a scanpath

through, or the AOIs estimated from them. It is important to distinguish between a priori AOIs, which can be obtained before viewing and post hoc AOIs, which are derived from the fixations on scanpaths. A priori AOIs can be divided into gridded AOIs and semantic AOIs. Gridded AOIs, as shown in Fig. 2a, are created by placing grids with equal size over the stimulus, which ignores the semantic aspects of the stimulus. Semantic AOIs, as shown in Fig. 2b, are the natural segmentation of the stimulus. Semantic AOIs are used, among other applications, for webpages, since webpages can be automatically segmented into rectangular visual elements (= AOIs) using the underlying source code (Akpınar & Yeşilada, 2013). Post hoc AOIs, as shown in Fig. 2c, are the areas that contain relatively many or long time fixations. The post hoc AOIs formed by the fixations on scanpaths from different observers can be detected by cluster algorithms like *k*-means, DBSCAN, and OPTICS, which group fixations with similar positions (He et al., 2017; Latimer, 1988; Naqshbandi et al., 2016). The advantage of post hoc AOIs are that they can be obtained automatically. Empirically, post hoc AOIs take into account the semantics of the stimuli to some extent. The disadvantages are that fixations are treated as isolated points on the two-dimensional plane. The temporal order and the duration attribute of the fixations are ignored in existing scanpath aggregation approaches. The consequence of this disadvantage can be well explained with the simple example shown in Fig. 3. Twenty scanpaths are randomly generated, so that the second fixations are uniformly distributed in the blue box and the third fixations are uniformly distributed in the red box. However, since the red and blue boxes are so close to each other, the post hoc AOI obtained by a clustering algorithm will most likely be the black box. Such an AOI covers two spatially close but semantically different AOIs, namely the blue box and red box. In particular, when the stimulus is a dynamic video, spatially identical or close but temporally different fixations may come from completely different visual elements. Therefore, the post hoc AOIs obtained from all the fixations fail to describe the dynamic movement pattern of scanpaths accurately. In this paper, we distinguish between short-time AOIs in a limited time segment (e.g., the blue and red boxes) and global AOIs across all time points (e.g., the black box).

Scanpath aggregation by representative scanpaths

The dynamic movement pattern in a group of scanpaths can be extracted by a representative scanpath, which can be seen as an aggregated scanning strategy for a sample of viewers. A representative scanpath s^* can be an existing scanpath in the sample, which minimizes the average distance from other sample members. Formally, given a sample of scanpaths $S = \{s^1, \cdots, s^M\}$,

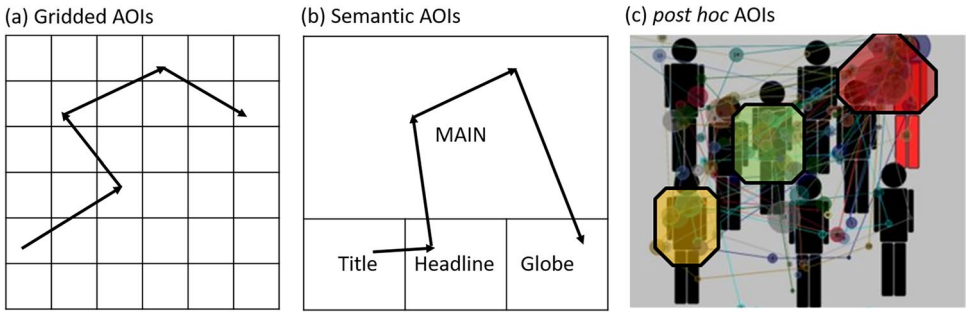

Fig. 2 An example of a priori AOIs and post hoc AOIs. Gridded AOIs (**a**), semantic AOIs of a web page (**b**), and post hoc AOIs (**c**)

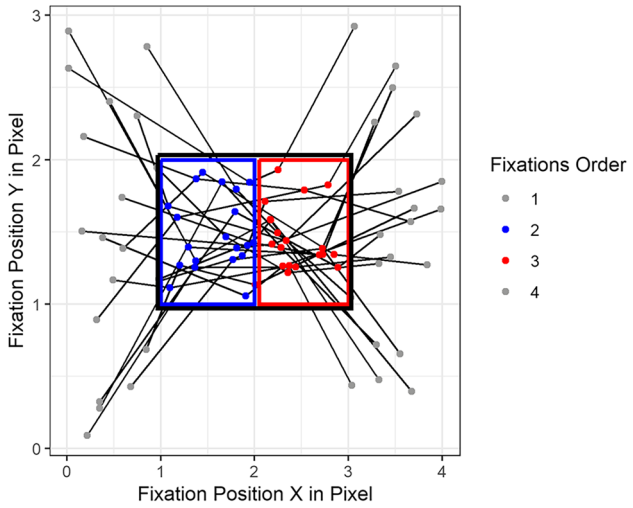

Fig. 3 An example of short-time AOIs (*red* and *blue boxes*) and global AOIs (*black box*) obtained from the fixations on a group of random generated scanpaths

$$s^* = \arg\ \min_{s \in S} \frac{1}{M} \sum_{i=1}^{M} d(s, s^m), \tag{1}$$

where $d(\cdot, \cdot)$ is the scanpath dissimilarity. For example, von der Malsburg and Vasishth (2013), Parshina et al. (2022), and Paape et al. (2022) cluster the scanpaths into several groups and find for each group a scanpath that minimizes the average Scasim (von der Malsburg & Vasishth, 2011) dissimilarity as the prototypical view pattern for that group.

Instead of identifying an existing representative scanpath from the sample, some methods have been proposed to generate an artificial scanpath to represent the group. Their common approach is to first define or find AOIs and then derive a representative scanpath from the AOIs (see Table 1). Eraslan et al. (2014) represent the scanpath with a string by replacing the fixations on it with a shorthand name of the semantic AOI. For example, the scanpath on the middle panel in Fig. 2 will be represented T - H - M - M - G (where T stands for 'Title' etc.). Then, a common subsequence shared by all subjects is identified as the representative scanpath. This method is limited to the cases where no common subsequence shared by individuals exists. It can be improved by the method of Hejmady and Narayanan (2012), who instead find the frequent subsequence supported by a specified number of subjects with the sequential pattern mining algorithm (Ayres et al., 2002). Similarly, Wang et al. (2022) detect frequent sub sequence using the k-mer analysis, which counts the frequency of subsequence of neighbouring elements of length k (Manekar & Sathe, 2018). The common limitation of these semantic-AOIs-based methods is that they are only suitable for visual stimuli where natural segmentation exists or that can be easily segmented by a computer, such as webpages. However, for an arbitrary image, it is usually difficult to make a decision on how to segment it. Such segmentation is usually subjective, and different ways of segmentation lead to different strings for scanpaths. The method of scanpath aggregation based on post hoc AOIs can well avoid the problems caused by segmentation and is applicable to stimuli without existing segmentation. The heuristic method and the Candidate-constrained Dynamic Time Waring Barycenter Averaging (CDBA) method proposed by Li and Chen (2018) obtained the segmentation of stimuli through post hoc AOIs by clustering the fixations on scanpaths. Their methods try to search a scanpath s^* that minimizes the average distance to all the scanpaths in sample, as shown in Eq. (1). However, the search space is no longer limited to the original sample S, but a set S of reconstructed scanpaths. Scanpaths in S are constructed by connecting the centers of the post hoc AOIs. Since the AOI centers can be used repeatedly in the same scanpath, the set S can contain a huge number of reconstructed scanpaths. To reduce the computational effort, the length of the scanpaths in S (number of fixations) is limited to the maximum length of the original scanpaths. Even so, the set S is still large and traversing all the scanpaths in it, namely the heuristic method, is very time consuming. A compromise is to initialize a random scanpath and then update the scanpath according to certain rules iteratively, namely the CDBA method, which achieves a local optimum after a smaller number of updates. Heuristic and

Table 1 Methods of AOI-based scanpath aggregation

Method	AOI	Description
Eraslan et al. (2014)	A priori (semantic)	(i) Represents scanpaths as string; (ii) find common string shared by all scanpaths
Hejmady & Narayanan (2012)	A priori (semantic)	(i) Represents scanpaths as string; (ii) find frequent strings shared by a majority of scanpaths
Wang et al. (2022)	A priori (semantic)	(i) Represents scanpaths as string; (ii) detect frequent sub sequence using the k-mer analysis
CDBA & Heuristik (Li & Chen, 2018)	Post hoc (global)	(i) Construct a candidate set of representative scanpaths; (ii) find an optimal scanpath in the candidate set that minimizes the average DTW distance to the original scanpaths

CDBA methods can be used to automatically find a representative scanpath in a sample of scanpaths for any static visual stimuli. However, the fixations on the detected representative scanpath are the centers of the post hoc AOIs obtained from all the fixations without considering their order and time period, namely the global AOIs; cf. the black box shown in Fig. 3. Such fixations fail to represent the location of the fixations in a particular time period. A better way is to use the short-time AOIs; cf. the blue and red boxes in Fig. 3.

In addition to the above methods, Burch et al. (2019) developed a visualization tool to aggregate the scanpaths. They first identify the post hoc AOIs based on the density of the fixations on the stimulus, and then present a hierarchical flow chart between the AOIs, in which the thickness of the lines between AOIs is determined by the transfer probability between them. This tool does not actually generate a representative scanpath, but visually aggregates the scanpaths hierarchically, which can also help us to understand the view patterns of a group. However, due to using of the post hoc AOIs, this method has the same limitation as Li and Chen (2018).

In this paper, we propose a new algorithm to identify short-time AOIs from a group of scanpaths and get the representative scanpath by linking the centers of the short-time AOIs. The basic idea to obtain short-time AOIs is to cluster the fixations from different time periods separately. The definition of short-time AOIs and how to detect them automatically will be explained formally and in detail in next section. Then, we evaluate our method on two publicly available eye-tracking data sets. Results show that the generated scanpath can be more representative than any existing scanpath from the sample.

Methodology

In this section, we first present an overview of our design, including a formal problem statement and the skeleton of our algorithm, which we call Short-Time-AOI Scanpath Aggregation (STASA). After that, we describe each step of our method in detail.

Problem statement

A fixation on the two-dimensional (2D) plane can be denoted by a quadruple (x, y, b, e), in which (x, y) is the space location of the fixation in pixel coordinates and (b, e) with $b < e$ are the start time point and end time point of the fixation. Thus, a scanpath s with n fixations is a sequence of fixations with $s = \{(x_1, y_1, b_1, e_1), \cdots, (x_n, y_n, b_n, e_n)\}$ and $b_1 < e_1 < b_2 < e_2 < \cdots < b_n < e_n$. Figure 4 shows an example of a scanpath with $n = 4$ in a three-dimensional (3D) coordinate system resulting from adding a time axis. Given a set of scanpaths $\{s^1, \cdots, s^M\}$, with $s^m = \left\{ \left(x_1^m, y_1^m, b_1^m, e_1^m\right), \cdots, \left(x_{n_m}^m, y_{n_m}^m, b_{n_m}^m, e_{n_m}^m\right) \right\}$, for $m = 1, \cdots, M$, we develop an efficient algorithm to generate a representative scanpath s^*. Following Li and Chen (2018), the representativeness of s^* can be evaluated by its average distance to all scanpaths in the sample

$$\frac{1}{M} \sum_{i=1}^{M} d(s^*, s^m). \tag{2}$$

In contrast to Li and Chen (2018), we do not fix $d(\cdot, \cdot)$ as the Dynamic Time Warp Distance (Vintsyuk, 1968) because it can only measure the distance of two scanpaths by considering them as trajectories, which ignores the difference of duration of the fixations on them. Anderson et al. (2015) provide an overview of common scanpath similarity measures. As dissimilarity measures can always be transformed to similarity measures, we use this two concepts interchangeably in this paper. These similarity measures fall into two categories. One focuses on only one or two aspects of scanpath similarity, such as fixation overlap and gaze shift of two scanpaths (Shepherd et al., 2010), linear distance of fixations (Mannan et al., 1995; Mathôt et al., 2012), and cross-recurrence (Anderson et al., 2013). Therefore, we do not use them in this paper. Another category of similarity measures covers as many aspects of scanpath similarity as possible,

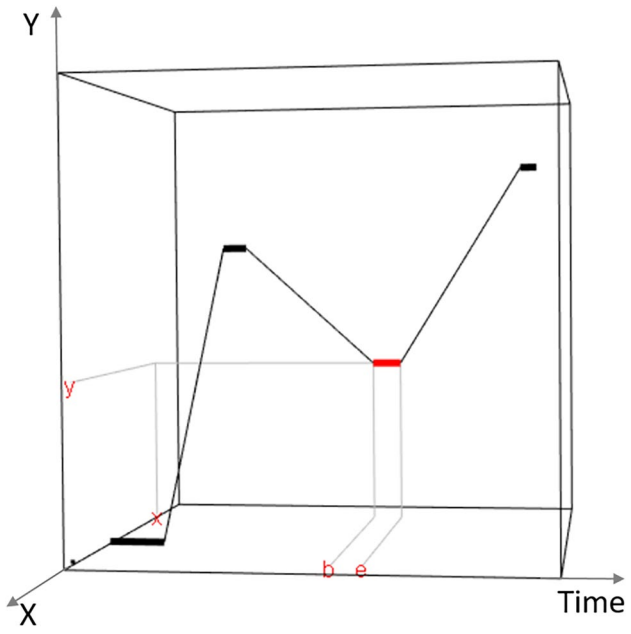

Fig. 4 An example of scanpath with four fixations in 3D. The third fixation (in *red*) locates on (x, y) and last from b to e

i.e., spatial and temporal similarity of fixations and order of fixations. They are the Levenshtein distance (Levenshtein et al., 1966), the Scasim (Von der Malsburg & Vasishth, 2011), ScanMatch (Cristino et al., 2010), and MultiMatch (Jarodzka et al., 2010). A brief description of these methods follows, as they are used in this paper to compute the average similarity in Eq. (1) and evaluate the representativeness of the obtained scanpath.

Levenshtein distance measures the difference between two strings, so that the scanpath has to be transformed into strings of AOIs first. The distance between two strings is the minimum number of single character edits (insertions, deletions, or substitutions) required to transform one string into the other.

Scasim is a dissimilarity measure based on the Levenshtein distance. Instead of converting the scanpath into a string, it generalizes the character editing operations to penalize the distance between fixations taking into account the high acuity in the fovea and the drop in resolution towards the periphery.

ScanMatch first converts scanpaths into strings after placing grids with equal size on the stimulus (see Fig. 2a) and names the grids with letters. A letter is repeated a certain number of times depending on the duration of the fixation on the corresponding grid cells, e.g., each individual letter represents a 50-ms bin. The resulting string sequence incorporates spatial location, sequential information, and temporal durations of the fixations on the scanpaths. Therefore, comparing them provides an overall similarity of the scanpaths. The strings are aligned and compared using Needleman and Wunsch (1970) algorithm. The Needleman–Wunsch algorithm creates a matrix with all scoring possibilities based on a substitution matrix, which provides a score for aligning two AOI letters. The substitution matrix encode information about the relation between each AOI (grid). The relationship can be based on the semantic segmentation of an image, or basically the Euclidean distance between the grids. Therefore, the similarity calculated by ScanMatch depends highly on the number of grids and the substitution matrix used in Needleman–Wunsch algorithm.

MultiMatch represents the two scanpaths to be compared as sequences of saccade vectors (the vectors between each two temporally consecutive fixations), say $(\vec{u}_1, \cdots, \vec{u}_m)$ for one scanpath and $(\vec{v}_1, \cdots, \vec{v}_n)$ for another. The saccade vectors on two scanpaths are aligned and compared from five perspectives, i.e., vector similarity, direction similarity, length similarity, position similarity, and duration similarity. The vector similarity is the average difference of the aligned saccade vector pairs, namely $||\vec{u} - \vec{v}||$, which actually incorporates the direction similarity (angle between $\vec{u}$ and $\vec{v}$) and length similarity ($||\vec{u}||$ - $||\vec{v}||$). The position similarity and duration similarity are for the fixations on the two ends of the saccade vectors. Position similarity is linked to their spatial distance, and duration similarity is their difference in duration.

In summary, Scasim calculates scanpath dissimilarity directly based on the pixel coordinates of the fixation, whereas ScanMatch replaces the fixations with the letters of grids they are located and calculates the similarity based on the grid's distance. The saccade vector used by MultiMatch is also based on the pixel coordinates of the fixation, but MultiMatch simplifies scanpaths firstly based on angle and amplitude of the saccades before calculating their similarity. After simplification, spatially closed fixations are merged. In this way, MultiMatch is similar to ScanMatch, where spatially closed fixations are also most likely be merged by the grids, depending on the size of the grids. In addition, ScanMatch and Scasim provide a single score for the overall similarity of two scanpaths, whereas MultiMatch provides five scores for each of the five different aspects.

Overall procedure of STASA

The principal idea of STASA is to get short-time AOIs first and then obtain "synthetic" fixations of the representative scanpath from short-time AOIs. In Fig. 5a we display the scanpaths in a three-dimensional (3D) coordinate system after adding a time axis. The four scanpaths in this figure concentrate more or less in the upper part of the stimulus from time point t_1 to t_2, and concentrate in the middle part of the stimulus from time point t_3 to t_4, and so on. These areas of concentration are short-time AOIs. A short-time AOI lasts from one time point to another, and has a bounded range in space, which can be denoted as (P, b, e), where P the set of vertices of the bounding polygon, and (b, e) are the start and end time point. After we obtain the short-time AOIs, we link them (and the fixations derived from them) in time order to get a representative scan path for the scanpaths. As shown in Fig. 5b, we display the representative scanpath together with its short-time AOIs back on the two-dimensional (2D) stimulus, which is a natural way of visualization for static stimuli. The polygons indicate the shape and location of the short-time AOIs and the area of the cycle is proportional to the duration of the short-time AOIs.

As shown in Fig. 6, our method consists of five steps: i) Represent scanpaths in a 3D coordinate system and take cross-section slices along the time axis; ii) Apply DBSCAN (Ester et al., 1996) clustering algorithm on the fixations on each slice; iii) Surround each cluster of fixations on each slice by a polygon; iv) Determine short-time AOIs based on the similarity of polygons on adjacent slices; v) Determine the fixations of the representative scanpath from short-time AOIs. Each of these steps is explained in detail in the following subsections.

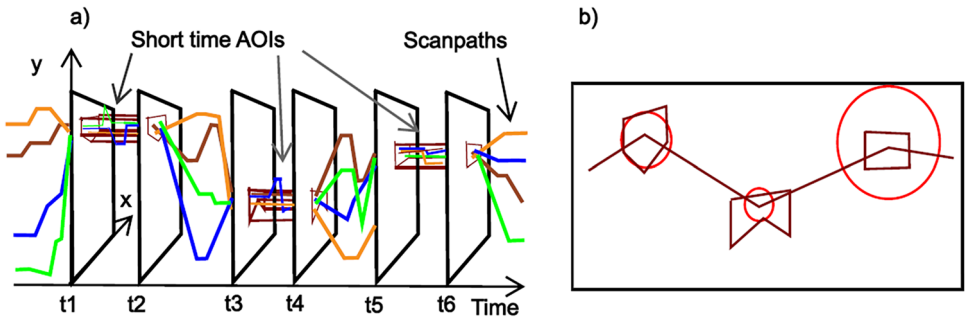

Fig. 5 From short-time AOIs to representative scanpath. **a** Scanpaths and short-time AOIs in 3D. **b** Short-time AOIs and representative scanpath in 2D

Take cross section slices along the time axis

All scanpaths are start-time aligned and represented in the 3D coordinate system. We take cross-section slices along the time axis with a small equal time distance δ, and get the space locations (x, y) of all fixations from all scanpaths on each slice. A smaller δ makes the calculation more accurate but increases the computational effort. For smaller data sets (usually a few dozen scanpaths with a few seconds of experimentation time), we can use the sampling interval of

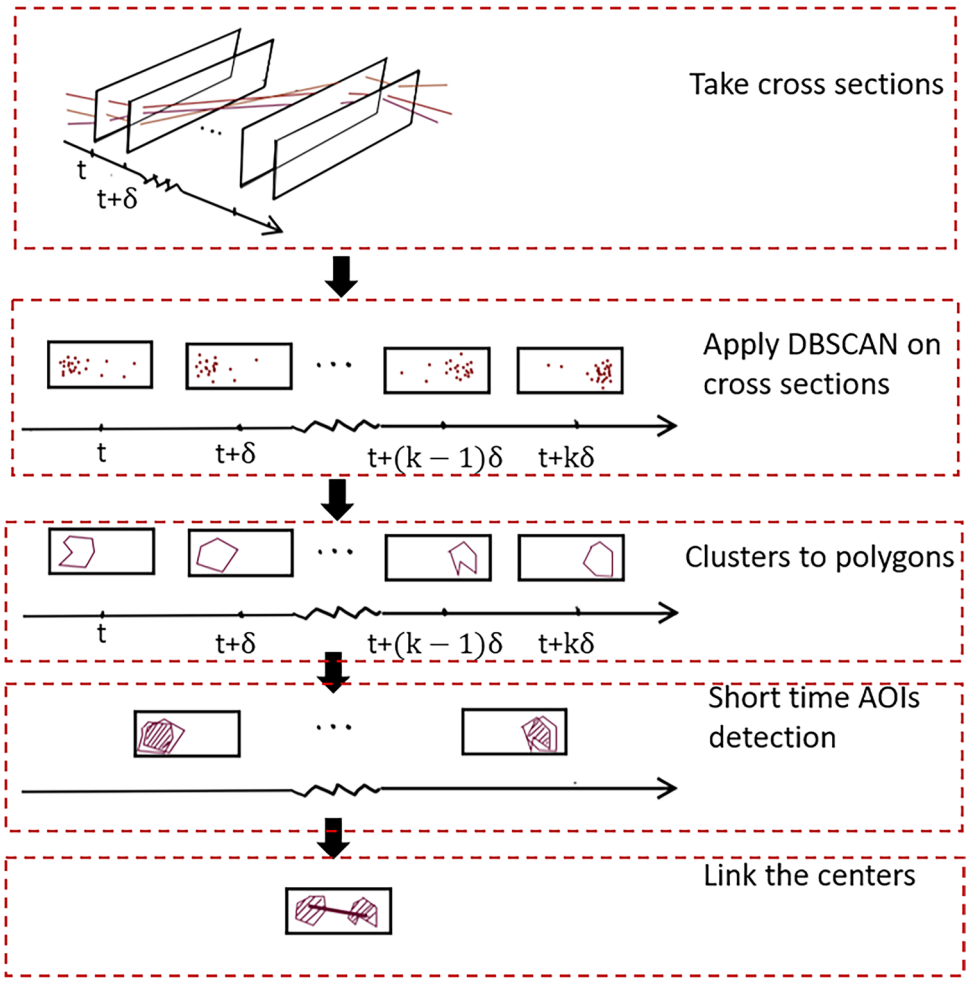

Fig. 6 Overall procedure of STASA

the original eye-tracking data, which is the inverse of the sampling frequency of the eye-tracker used. For dynamic stimuli, a slice can be a single frame of the video.

Apply DBSCAN on slices

For fixation points on each cross-section slice, we use the DBSCAN (Ester et al., 1996) clustering algorithm to remove outliers and find relatively dense areas on that slice. The principal idea of DBSCAN is to find dense areas and form clusters by expanding the dense areas recursively. The number of clusters is not prespecified and is determined by DBSCAN. The dense area is formed by those points that have at least *minPt* neighboring points within a given search radius ϵ. Compared with other traditional clustering algorithms like *k*-means, DBSCAN is well suited for detecting AOIs on visual stimuli, since it detects arbitrarily shaped clusters without apriori knowledge of the stimuli, whereas *k*-means and other clustering algorithm often prefer roughly circular AOIs. The choice of parameters ($minPt$, ϵ) is crucial to our algorithm. Different parameter values will result in different numbers and sizes of clusters. Because we will later compare the similarity of clusters on two adjacent slices, having multiple clusters on the same slice will make the later steps difficult. Since our goal is to find a common-view pattern for a group of scanpaths, we identify only one major cluster for each slice. This cluster should i) include as much points as possible; ii) include points as dense as possible. The density of the cluster can be controlled by the parameter ϵ; the smaller the ϵ, the higher the density, and the minimum number of points in the cluster is restricted by *minPt*. To achieve our goal, we get the optimal (ϵ^*, $minPt^*$) as follows:

- For each slice by applying DBSCAN, we start with a large ϵ (width of the visual stimulus) and a small *minPt* (half of the number of scanpaths in set *L*), which will result in one cluster containing all the points as shown in Fig. 7a.
- Then we keep *minPt* fixed and decrease ϵ by one pixel gradually. The reduction of ϵ requires denser points within the cluster, which causes the original cluster formed under larger ϵ to split into two or more denser clusters and some isolated points (the grey points in Fig. 7b) are no longer contained within any cluster.
- Then we continue to reduce ϵ, so that more isolated points are excluded from the clusters and the relatively sparse clusters disappear. We stop decreasing ϵ when only the last densest cluster is left (see Fig. 7c). This ϵ value is the searched optimal value ϵ^*.

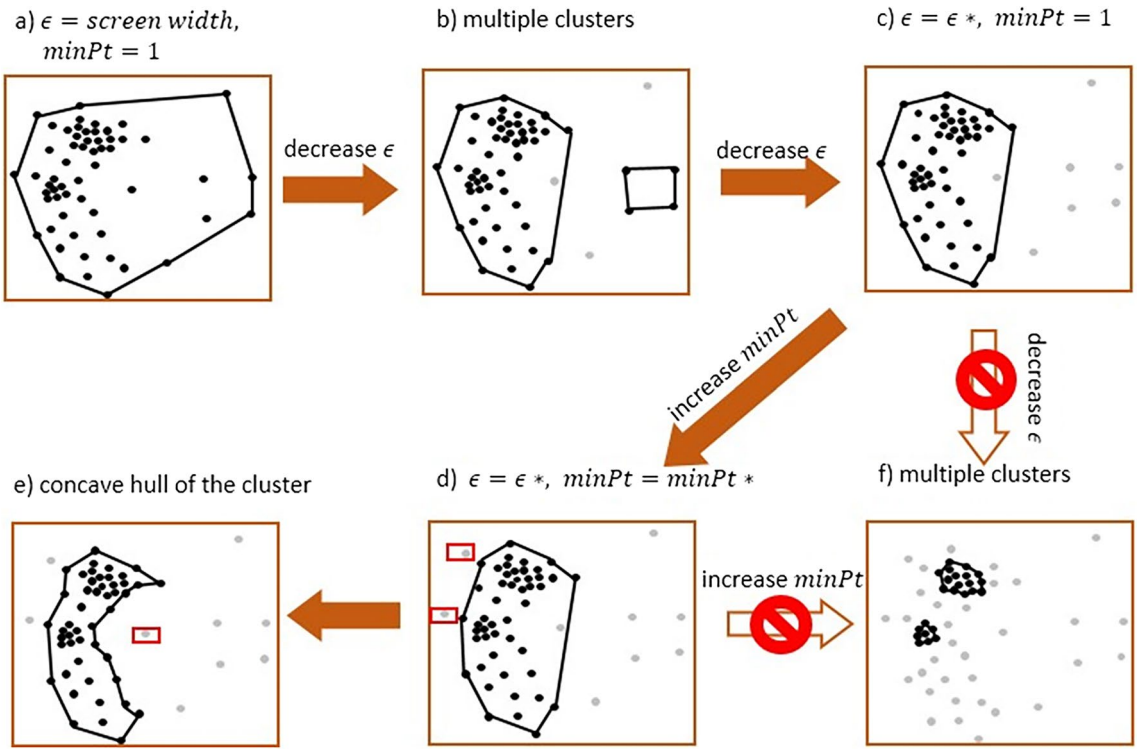

Fig. 7 The process of determining the relatively large and dense cluster for each cross-section slice and obtaining the concave boundary of the cluster

- Next, we keep the value of ϵ fixed by ϵ^*, and increasing the value of *minPt* by one gradually. An increase in *minPt* also requires denser points within the cluster, which excludes some distant points at the margin of the cluster, as shown in the red borders in Fig. 7d. However, it may also lead the cluster to split into multiple clusters. We choose the maximal value of *minPt* so that the number of clusters remains at one as the searched value $minPt^*$.

Surround clusters with polygons

The cluster obtained by DBSCAN on each slice is a collection of points. To estimate a boundary of the cluster, we obtain the concave hull, which is a polygon using the approach of Park and Oh (2012). A concave hull is a shrink-wrapping of a set of points, which has a smaller area compared to the convex hull and represents a natural boundary of the points. From the simple example shown in Fig. 7d, e, we can see that the concave hull is better than the convex hull to reflect the geometrical characteristics of fixations. The outlier points in the red box are now no longer in the cluster.

Short-time AOIs determination

After we obtain the concave polygons on each slice, we determine the short-time AOIs. First, we determine the time interval (b, e) of the short-time AOIs as follows.

We calculate the similarity $S_{(i,i+1)}$ of the polygons on i-th and $(i+1)$-th slice for all i with

$$S_{(i,i+1)} = \frac{I_{(i,i+1)}}{U_{(i,i+1)}}, \tag{3}$$

where $U_{(i,i+1)}$ is the combined area of the polygons on slice i and $i+1$, and $I_{(i,i+1)}$ the overlapped area of the polygons on slice i and $i+1$. The similarities $S_{(i,i+1)}$ against the time point $i * \delta$ can be displayed in a scatter plot as shown in the right panel of Fig. 8. The time intervals (b_j, e_j) of the short-time AOIs are those in which all the calculated similarities are greater than a predefined threshold θ. Extremely short intervals, which contain less than three slices, are excluded, making sure that the duration of the resulting fixation is no shorter than 100 ms. We can see that under different θ we obtained different representative scanpaths. How to choose the threshold θ will be discussed latter.

After we obtained the time intervals of the short-time AOIs, we compute the spatial scope of them as the union of areas surrounded by the polygons on all the slices within the corresponding time interval. Denote by p_i the obtained polygon on i-th slice, then the polygon P_j of j-th short-time AOIs is defined as

$$P_j = \bigcup_{\delta \cdot i \in [b_j, e_j]} p_i \tag{4}$$

Suppose we get J short-time AOIs in the previous step. Then the representative scanpath s^* would consist of J fixations. The spatial location of j-th fixation (x_j, y_j) is the geometric center of the polygon P_j, and the start time and end time of j-th fixation is the start time and end time of the j-th short-time AOI, namely b_j and e_j, for all $j \in \{1, \cdots, J\}$. The advantage of using the geometric center is that it has the smallest distance to the vertices of the AOI. However, if we get a short-time AOI that is very close to the edge of a circle, then its geometric center will be surrounded by this AOI, but it will not be on part of this AOI. It is also difficult to use any point on the circle to represent the position of a circle. However, this is not due to the use of the concave hull, since the geometric center of the concave hull and convex hull in this case are the same or similar. The reason we chose concave hull is to make the resulting AOI as much as possible in the region of dense fixations. Therefore, when visualizing the scanpath, we recommend displaying the scanpath together with the short-time AOIs in which each fixation is located, as shown in the simplified diagram in Fig. 5b. The visualization function of our package also provides the option for users to show the AOIs together with the scanpath.

Choice of the threshold parameter

The threshold θ we mentioned above is a tuning parameter of our method. A smaller θ will combine polygons on two adjacent slices, say i-th and $(i+1)$-th slices, with smaller overlapping area to form a larger short-time AOIs. There are two possible reasons for the small overlap of polygons on adjacent slices. i) The scanpaths are temporally highly unsynchronized. For example, a group of scanpaths (say each with n fixations) are spatially almost identical (focusing on the same visual elements), but the duration of each fixation is different, so that they are not synchronous. Then our task is to generate a scanpath that is also spatially almost identical to these scanpaths, but where the duration of each fixation is the overlap time of the n fixations on all the scanpaths at the similar location. The parameter θ can be used deal with this situation. θ is the minimum requirement for the similarity of AOIs on two adjacent cross sections within the same short-time AOI. If the scanpaths are perfectly synchronized, then the resulting AOIs on each cross section will be identical within the same short-time AOI, requiring the value of θ to be set to one in order to distinguish between different short-time AOIs (fixations at different locations).

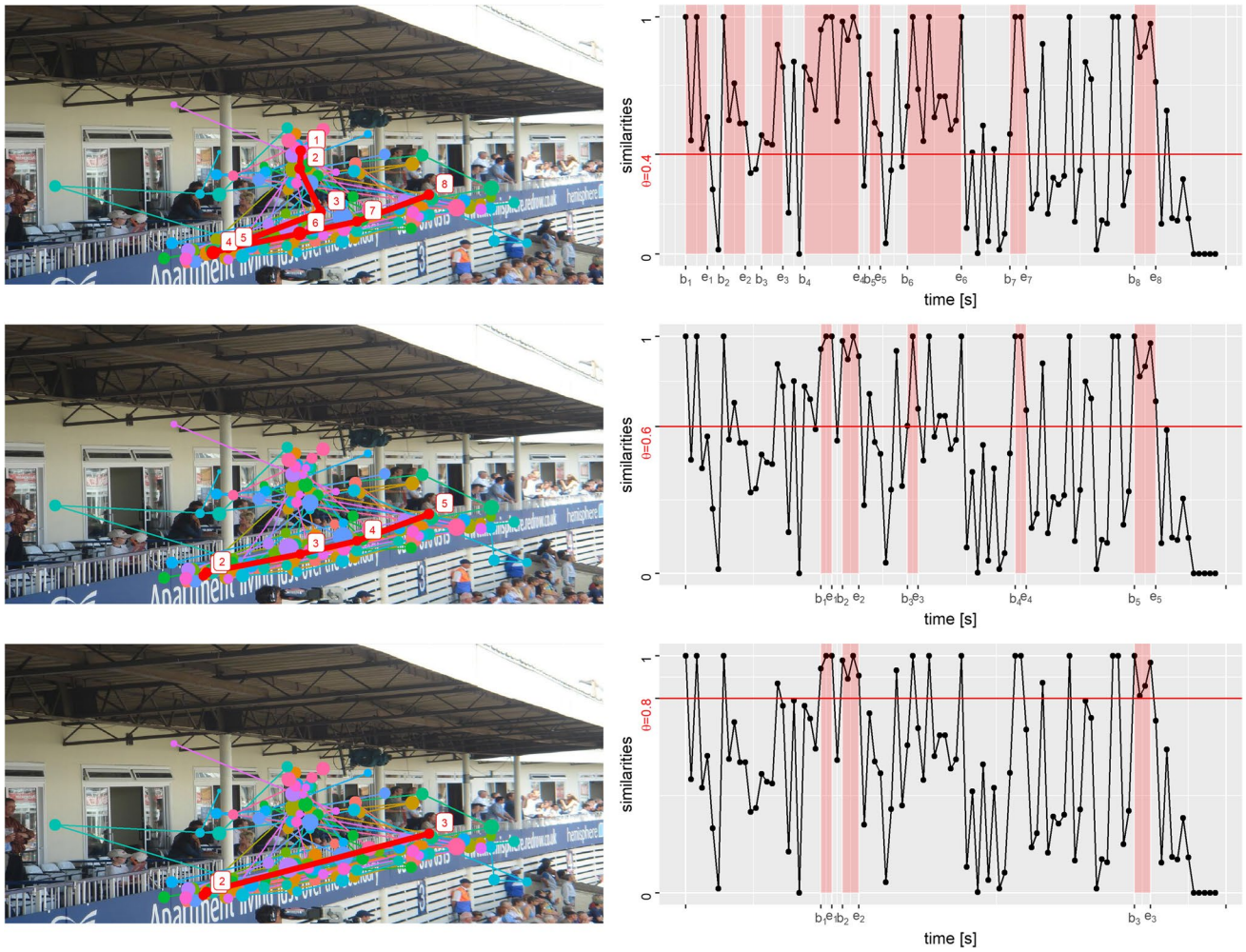

Fig. 8 An example of representative scanpaths with different values of θ. The scanpaths are in the left panels. The *thick red line* is the representative scanpath, the *other colored lines* are scanpaths of individual observers. The fixations on representative scanpaths are numbered by time order. The *right panels* show how the time intervals of the short tine AOIs are obtained under corresponding θ

However, in most cases, observers are not synchronized due to the different durations of each fixation, even though their scanpaths are spatially similar. We assume that their fixations on similar locations overlap in time. Then different AOIs will be formed on different cross sections, as different cross sections contain different points from these n fixations. In such a case, different values of θ tolerate different degrees of degree of desynchronization. ii) The two polygons are from two different visual elements. Most observers' fixations were on one of the visual elements at the time point corresponding to the i-th slice and on the other visual element at the time point of $(i+1)$-th slice. However, again due to the high degree of view freedom, a few observers gazed earlier on another visual element at the time point of the i-th slice or stayed on the previous visual element at the time point of $(i+1)$-th slice. This results in two polygons both covering two different visual elements. In this case, a larger θ can result in two spatially different AOIs, corresponding to two different visual elements of the stimulus, instead of their combination.

Figure 8 shows three different representative scanpaths obtained by our method with three different values of θ. The left panels contain representative scanpaths and right panels illustrate how the time interval of the short-time AOIs are obtained under the corresponding value of θ. The image is one of the 1003 stimuli in the MIT1003 data set (Judd et al., 2009), which was observed by 15 individuals. For convenience, we only show three cases with θ at 0.4, 0.6, and 0.8 from top to bottom. All the three representative scanpaths contain the horizontal path at the bottom of the stimulus, which is the most common view pattern of the eye movement in this sample. As θ increases, the representative scanpath tends to be shorter and contains no longer the vertical eye movement at the beginning of the viewing. This is because at the beginning of the experiment, the gaze of the viewer is attracted to the center of the screen due to a

centered fixation cross before the stimulus is displayed. After the stimulus is displayed, the viewers move their gaze at different speeds. Some viewers were quicker to target the texts on the wall and then moved their gaze to the right horizontally. Hence, fixations at the very beginning of the viewing cannot form short-time AOIs under larger θ because they are actually far apart in time.

The choice of θ also depends on how the representativeness of the scanpath is defined. A suitable definition is usually multidimensional, such as the five dimensions of MultiMatch. In practical applications, we recommend tuning θ between 0.1 and 0.9 spaced by 0.1, and then select the θ that minimizes the average dissimilarity to all the scanpaths in sample. The dissimlarity is recommended to be Scasim. Scasim is chosen for the following reasons. As previously described, most scanpath similarities focus on only one aspect of the scanpath similarity, while Levenstein distance, Scasim, ScanMatch, and MultiMatch are able to cover both spatial and temporal aspects of the scanpath similarity. MultiMatch, however, gives five scores for five different aspects of scanpath. Making one score larger would reduce some other scores. Levenstein distance and ScanMatch are also excluded as the default because they have to divide the stimuli in advance into different regions (or gids). Different divisions will lead to different results. In addition, we only tried nine different theta values between 0.1 and 0.9. The user could also try more values, but based on experiments on two large datasets, we do not think it is necessary, since the difference of the scanpaths obtained by the neighboring θ's values is not significant.

Furthermore, if the scanpaths within the samples are very different from each other, we may obtain an unrealistic representative scanpath with extremely short fixation durations or extremely long saccade durations, as shown in the right panel of Fig. 8 by $\theta = 0.8$. In this case, we should choose a smaller θ to avoid such an unrealistic scanpath. However, the user does not have to choose the value of θ manually. Different θ values will generate scanpaths with different representativeness (i.e., average similarity with all the scanpaths of the sample). By default, our method will automatically select the θ that generates a scanpath with the highest representativeness, and an extremely unrealistic scanpath is certainly not highly representative (because it is so different from the real scanpaths). Therefore, the final scanpath will not be the case where $\theta = 0.8$, rather more likely the case where $\theta = 0.4$. It is possible that the obtained scanpath at the optimal θ is still unrealistic, or partially unrealistic. For example, in Fig. 8, at θ of 0.4, the time intervals between the fixations (i.e., the saccade duration) of the second half of the scanpath are still very long. If 0.4 is the optimal θ value (it may not be, we just show only three values of θ for the sake of convenience), then it suggests that these observers have a common observation pattern in the first half of the scanpath, but the second half of the scanpath does not.

Empirical evaluation

To evaluate our method, we compared the performance of our method with CDBA and heuristic methods on two large public benchmark eye-tracking data sets, namely MIT1003 data set (Judd et al., 2009) and OSIE data set (Xu et al., 2014). The MIT1003 data set contains 1003 natural indoor and outdoor scenes freely viewed by 15 observers for 3 s. The longest dimension of each stimulus is 1024 pixels, and the other dimension ranges from 405 to 1024 pixels.

The OSIE data set contains 700 images freely viewed by 15 observers for 3 s. All the images are of the size 800×600 pixels. One difficulty with this data set for us is that it contains only the duration information for the fixations and their exact start and end time points are unknown. That is, the data structure of each fixation in this data set is (x, y, d) instead of (x, y, b, e), where d is the duration of the fixation. However, our method requires the exact start and end time points to align the scanpaths and to take slices along the timeline. To solve this problem, we constructed artificial start and end time points for each fixation as follows. For each scanpath $s = \{(x_1, y_1, d_1), \cdots, (x_n, y_n, d_n)\}$ in the OSIE data set, we calculated its total saccade time (ts) with

$$ts = 3 - \sum_{i=1}^{n} d_i, \tag{5}$$

which is the difference of the total observation time and the total fixation time. We then divide ts evenly between each of the two fixations. That is, the saccade time between each two fixations will be set as ts/n. Thus the start time point of i-th fixation can be computed with

$$b_i = \sum_{h=1}^{i-1} d_h + (i-1) \cdot ts/n, \tag{6}$$

and the end time point of i-th fixation can be computed with

$$e_i = \sum_{h=1}^{i} d_h + (i-1) \cdot ts/n. \tag{7}$$

This operation may introduce an error. A more accurate way would be to assign the duration of the saccade based on its amplitudes. However, since the time of saccades only takes up a small percentage of the overall scanpath (in the OSIE data set it is 17% in average), the difference between the artificial start and end time points and their unknown true values will also be small. To simplify the calculations, we divide the saccade time equally as shown in Eqs. (6) and (7). Experiments show that our method still outperforms other methods when this error is included.

Since there exist no ground truth representative scanpaths, for each stimulus the average similarity (or distance) of the

obtained representative scanpath to all the 15 scanpaths is used to evaluate the representativeness for that stimulus. As mentioned above, Scasim, MultiMatch, and ScanMatch are used to measure the similarity of two scanpaths. MultiMatch provides five scores for the five dimensions, i.e., vector similarity, direction similarity, length similarity, position similarity, and duration similarity. ScanMatch and Scasim provide a score for the overall scanpath similarity. MuitiMatch and ScanMatch normalize the obtained scores into the interval [0, 1], and the higher the score, the higher the similarity, while Scasim provides score for the dissimilarity in milliseconds, thus the lower the score, the higher the similarity.

For a fair comparison with the CDBA and the heuristic methods, we set the involved parameters of the ScanMatch method to the same as Li and Chen (2018), that is Xbin = 24, Ybin = 18, Threshold = 3.5, GapValue = 0, TempBin = 100. The parameter δ of our method are set to 30 ms. θ is tuned from 0.1 to 0.9 with distance 0.1.

In Table 2, we present the average representativeness scores for both the MIT1003 and OSIE datasets, each comprising 1003 and 700 stimuli, respectively. The scores are provided for our method, the heuristic method, and the CDBA method. Since the authors of the latter two methods did not publish their source code, the scores of these two methods in this table are from Li and Chen (2018). As shown in Table 2, the average overall representativeness of our method (the scores by ScanMatch) clearly outperform the CDBA and heuristic methods on both data sets. With the five scores of MultiMatch we can identify exactly where this difference comes from. First, the representative scanpaths obtained by our method are on average more representative in terms of the overall vector similarity in both data set, which mainly comes from the high similarity in vector length. The position of the fixations on our representative scanpath is also more representative on both data sets. The position of the fixations in our method are the centers of the short-time AOIs, while in the other two methods, they are the centers of the global AOIs. This is empirical evidence for our key assumptions, i.e., using the centers of the short-time AOIs as the position of fixations can be more representative for the fixations in a particular time period. For duration similarity, our method shows to be advantageous on the OSIE data set, but performs worse on the MIT1003 data set. The CDBA and heuristic methods use the average duration of all fixations within each global AOI as the duration of the fixations for representative scanpath. If we calculate the durations of the fixations in the same way, the score of duration similarity would also be increased (from 0.53 to 0.60 by dataset MIT1003). The disadvantage of using average duration, however, is that it is not possible to give the exact start and end time of each fixation for the representative scanpath, and it may make the total duration of all fixations greater than the actual duration of the viewing, when the number of fixations is large. Therefore, we prefer to calculate the duration of fixations as the difference of the end and start time points of short-time AOIs.

Then we compare our method (STASA) with a simple method of finding a scanpath in the raw sample. That is, given the scanpath dissimilarity (or similarity), find a scanpath in the sample that has the smallest average dissimilarity (or largest average similarity) with other sample members. As scanpath dissimilarity and similarity, we choose Scasim and ScanMatch. We denote these two methods as minScasim and maxScanMatch, respectively. For ScanMatch, we divide all stimuli into 12 x 8 grids, the same as the authors of ScanMatch did in their original paper. For STASA, δ of our method are set to 30 ms as default, θ is tuned from 0.1 to 0.9 with distance 0.1.

Figure 9a shows that when using Scasim as the dissimilarity measure, the representativeness of the scanpath obtained by STASA is significantly higher than that of the original scanpath in the sample for both data sets (p value < 0.01 by t test and Cohen's $d = 1.21$ by MIT1003; p value < 0.01 by t test and Cohen's $d = 1.71$ by OSIE). However, when using ScanMatch as the similarity measure, as shown in Fig. 9b, the representativeness of the scanpath obtained by STASA is only slightly better than the scanpath in the sample (p value < 0.01 from t test and Cohen's $d = 0.35$ by MIT1003; p value $= 0.01$ from t test and Cohen's $d = 0.14$ by OSIE). One reason for this discrepancy is that both Scasim and ScanMatch calculate the similarity of scanpath by first matching the fixations or grids and then comparing the differences between the paired fixations or grids.

Table 2 Average representativeness of different methods on MIT1003 and OSIE data set. The scores are normalized to range from 0 to 1

Dataset	Algorithm	Vector	Direction	Length	Position	Duration	ScanMatch
MIT1003	STASA	0.938	0.736	0.916	0.898	0.553	0.686
	Heuristic	0.882	0.749	0.905	0.874	0.674	0.474
	CDBA	0.882	0.749	0.905	0.875	0.674	0.476
OSIE	STASA	0.948	0.710	0.934	0.885	0.644	0.632
	Heuristic	0.843	0.702	0.850	0.821	0.620	0.415
	CDBA	0.843	0.849	0.820	0.701	0.617	0.416

Note. The scores of heuristic and CDBA are from Li and Chen (2018)

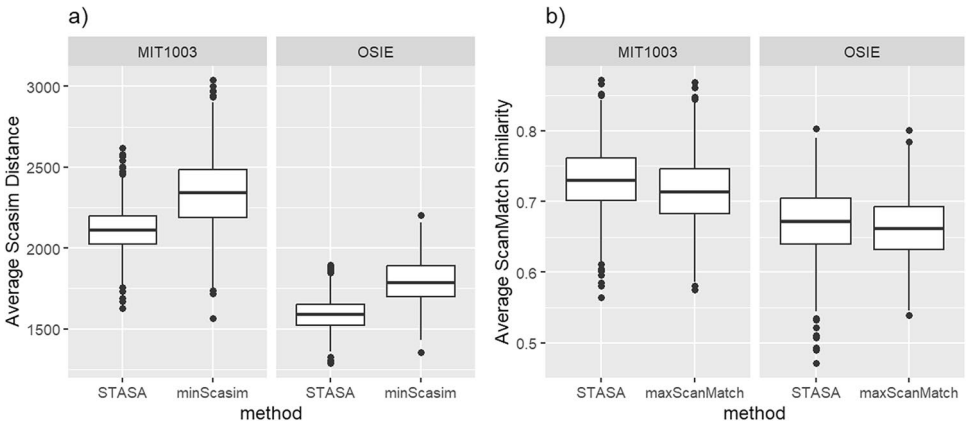

Fig. 9 Comparison of the performance of STASA and minScasim, as well as STASA and maxScanMatch on data sets MIT1003 and OSIE

However, Scasim directly compares the exact position (pixel coordinates) of the fixations, while ScanMatch compares the grids where the fixations are located. Consequently, ScanMatch cannot discern significant differences between the representative scanpath generated by STASA and the representative scanpath identified by maxScanMatch, when the ScanMatch's spatial grid is too coarse-grained.

In addition, we only tried nine different theta values equally spaced between 0.1 and 0.9. Theoretically, if more values of θ are tried, it should be possible to generate a scanpath with higher representativeness. However, as shown in Fig. 10, we do not think it is necessary because the difference of the scanpaths obtained by the neighboring θ's values on this two data sets is not significant.

Summary and discussion

This paper introduces a new method to identify a representative scanpath in a sample by following the principle of Gestalt theory that fixations are from whole AOIs instead of single gaze points. We introduced the concept of short-time AOIs, which are the AOIs formed by the fixations on scanpaths in a short period of time and obtain the fixations of the representative scanpath from the short-time AOIs. Older methods employ global AOIs, which are formed by all the fixations on the scanpaths without considering their order and duration information.

We compared our method with the existing global AOIs-based method on two large public accessible eye-tracking data sets. The results show that the representative scanpaths found by our method have higher average similarity (representativeness) to the original scanpaths as measured by ScanMatch (Cristino et al., 2010). We also use MultiMatch (Jarodzka et al., 2010) to measure the representativeness in different aspects. Results show that the scanpaths found by our method have higher representativeness in the overall saccade vector. The representativeness of the saccade vectors is essentially determined by the position of the fixations at their two ends. This is empirical evidence of the advantages of short-time AOIs. However, the limitation of the above comparison is that since Li and Chen (2018) did not publish the code and data for the global AOI-based methods, we can

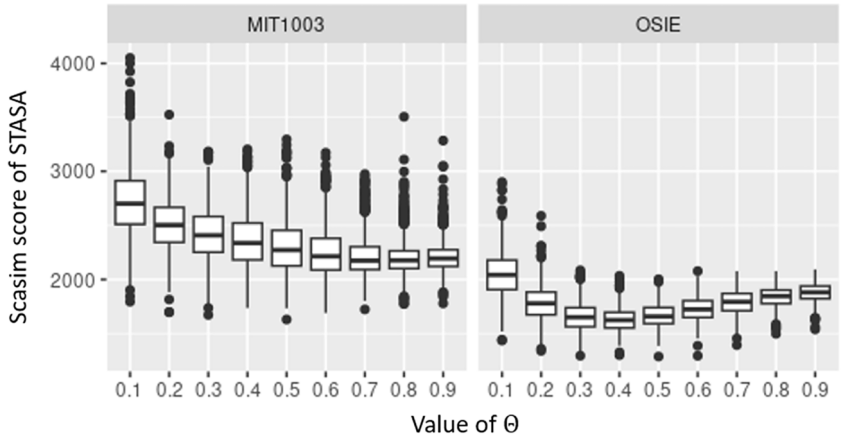

Fig. 10 Comparison of the performance of STASA and minScasim, as well as STASA and maxScanMatch on data sets MIT1003 and OSIE

only compare the average performance but without statistical tests to support the above conclusions.

Instead of generating a non-existent scanpath, an existing scanpath from the original sample can be found as a representative scanpath. This kind of method greatly reduces the computation time. We also compared our method with such simple methods. We performed the computations on a personal computer. For the two datasets used in this paper, the former can complete the computation for a single stimulus in less than a second, and STASA takes about 5 s on average. Although computation time is a disadvantage for STASA, times in the range of seconds are acceptable for general application. Regarding the representativeness, the scanpaths generated by STASA have higher representativeness than that of any existing scanpath. The above comparison depends on which scanpath similarity measure is used. STASA has a larger advantage when using Scasim as the similarity measure, which is sensitive to the location of the fixations. When using ScanMatch as the similarity measure, the advantage of STASA is no longer obvious. This is because ScanMatch represents fixations with similar locations in the same grid. If the grids are large, the differences between scanpaths will tend to be reduced. In the above comparison by ScanMatch, we partition all the stimuli into 12 x 8 grids, the same as the authors of ScanMatch did in their original paper. We expect that on finer grids, the advantage of STASA will be more obvious.

The aim of this paper is to develop a general methodology for generating representative scanpaths for a sample of scanpaths. We use Scasim to evaluate the generated scanpath and we propose a tuning strategy for the parameter of STASA in cases where the user does not specify a scanpath similarity in advance. However, potential subjectivity could arise from the user by specifying the similarity measure. This is hard to avoid because the similarity of a scanpath covers many aspects, including temporal, spatial, or semantic. Therefore, there exists no universally unbiased similarity for scanpaths, so that it is hard to generate a universally unbiased representative scanpath. However, if the user knows exactly which scanpath similarity is best suited for their specific task, our method can be adapted to that scanpath similarity by adjusting the parameter θ.

The introduced method in this paper is neither a scanpath clustering nor a scanpath classification method. Scanpath classification models can be used to characterize visual strategies of groups such as experts and novices, or to assign class labels, such as expertise level, to viewers depending on their scanpaths (e.g., Fuhl et al. 2019; Coutrot et al. 2018). Scanpath clustering models divide the sample into groups based on the similarity of the scanpaths with each other (e.g., Burch et al. (2018) and Haass et al. (2016)). The common task of all these methods including ours is to find view patterns for a sample of viewers. The difference is that our method is only for one group of viewers, whereas scanpath classification looks for view patterns for each of multiple groups, and scanpath clustering is to group viewers of scanpaths. Theoretically, our approach can also be used in the task of scanpath classification. For example, we can compute a representative scanpath for each class as the view pattern. A viewer can be assigned to the class who's representative scanpath has the highest similarity to the viewer's scanpath. Our approach can also be a building block for scanpath clustering. A representative scanpath serves as a cluster average. For example, in the studies of von der Malsburg and Vasishth (2013), Parshina et al. (2022), and Paape et al. (2022), scanpaths are clustered into groups firstly according to their similarities with each other. Then a prototypical view pattern for each cluster is identified using an existing scanpath in the cluster, which maximizes the similarities to all other cluster members. If the scanpath generated by our method has greater average similarity to all cluster members, then it is theoretically more suitable to be the searched prototypical view pattern. However, after clustering, the scanpaths within each cluster may be highly similar to each other. It depends on which cluster method is used; some cluster methods remove outlier scanpaths while clustering, such as DBSCAN, while others do not, such as k-means. If there are no outliers, the scanpaths within a group are highly similar to each other, so that an existing scanpath can already be highly representative. Therefore, it is to be expected that the scanpaths generated by our method will not improve the representativeness that much. Nevertheless, in some application scenarios, scanpaths are not grouped by their similarity, but based on the subject groups (e.g., novices and experts), or experimental settings, such as the task assigned to a subject. In such situations, since our method removes the outlier fixations when constructing the short times AOIs, it can be expected to generate more representative scanpath than any original scanpath. Also, if scanpaths are noisy for short periods (e.g., realistic driving simulation), the representative scanpaths obtained by our method might have the added benefit to smooth out noise.

The noise level of a set of scanpaths also depends on the number of objects on the stimulus. One reviewer of our paper suggested a potential relationship between the stimuli and the representativeness score of the representative scanpath. That is, one could expect a higher representative scanpath for stimuli containing salient objects driving observers' gaze and lower for stimuli of landscape scenes. We used a pre-trained deep learning salient object detection model (see Hou et al. 2017) to identify the salient objects on each stimulus of the MIT1003 data set and compared the representativeness of the scanpaths found by our method (STASA) with the scanpath minimizing the average Scasim distance (minScasim).

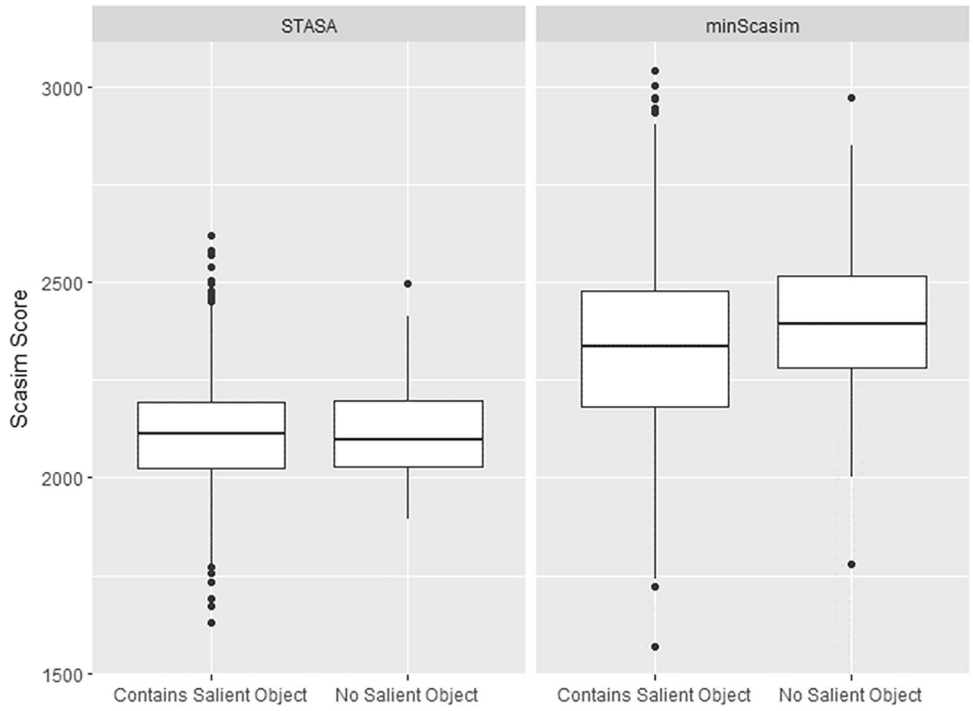

Fig. 11 Comparison of the representativeness of the scanpaths for stimuli with and without salient objects

However, as shown in Fig. 11, we did not find a noticeable difference in either of the methods. A possible reason for this is that there are too many objects in some stimuli. If only a few salient objects exist in the image, the scores of the representative scanpath should also tend to be higher on average, because if there are more salient objects in the stimuli, the observation strategies of different observers may vary widely, which leads to a lower degree of representation of the obtained representative scanpaths. We therefore used the deep learning tool YOLO3 (see Redmon and Farhadi, 2018) to automatically detect the number of objects contained in each image. As shown in Fig. 12, it is true that the score of representative scanpaths is highest (the average Scacim score is lowest) when the number of objects is only two. We can

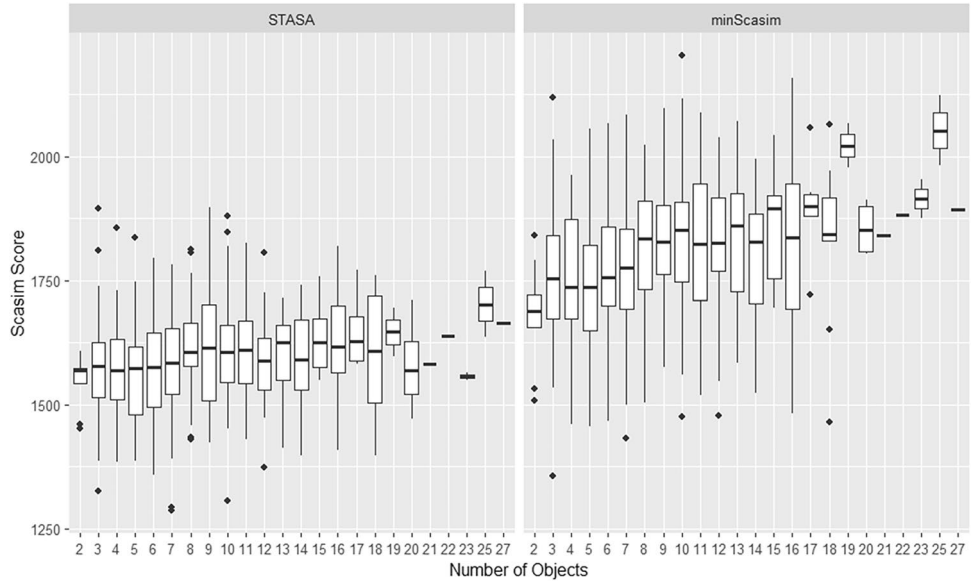

Fig. 12 The representativeness of the representative scanpaths for stimuli with different number of objects

also see a slight upward trend when the number of objects increases.

Authors contributions The authors made the following contributions. He Huang: Conceptualization, Writing - Original Draft Preparation, Writing - Review & Editing; Philipp Doebler: Conceptualization, Writing - Review & Editing; Barbara Mertins: Writing - Review & Editing.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability Two publicly available data sets (MIT1003 and OSIE) are used in this paper. They can be found for example under http://saliency.mit.edu/datasets.html.

Code availability The code for this paper is publicly accessible.

Our algorithm is implemented in R (R Core Team, 2021) and developed as an R package, which can be found under https://github.com/HeHuangDortmund/scanpathAggr

Code to produce the results in this paper can be found under https://github.com/HeHuangDortmund/STASA_ms

Declarations

Conflicts of interest/competing interests Not applicable.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Akpınar, M. E., & Yeşilada, Y. (2013). *Heuristic role detection of visual elements of web* (pp. 123–131). Springer.

Anderson, N. C., Anderson, F., Kingstone, A., & Bischof, W. F. (2015). A comparison of scanpath comparison methods. *Behavior Research Methods, 47*, 1377–1392.

Anderson, N. C., Bischof, W. F., Laidlaw, K. E., Risko, E. F., & Kingstone, A. (2013). Recurrence quantification analysis of eye movements. *Behavior Research Methods, 45*, 842–856.

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential pattern mining using a bitmap representation. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 429–435.

Burch, M., Kumar, A., & Timmermans, N. (2019). An interactive web-based visual analytics tool for detecting strategic eye movement patterns. *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, 1–5.

Burch, M., Kurzhals, K., Kleinhans, N., & Weiskopf, D. (2018). EyeMSA: Exploring eye movement data with pairwise and multiple sequence alignment. *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, 1–5.

Coutrot, A., Hsiao, J. H., & Chan, A. B. (2018). Scanpath modeling and classification with hidden Markov models. *Behavior Research Methods, 50*(1), 362–379.

Cristino, F., Mathôt, S., Theeuwes, J., & Gilchrist, I. D. (2010). ScanMatch: A novel method for comparing fixation sequences. *Behavior Research Methods, 42*(3), 692–700.

Eckstein, M. K., Guerra-Carrillo, B., Singley, A. T. M., & Bunge, S. A. (2017). Beyond eye gaze: What else can eyetracking reveal about cognition and cognitive development? *Developmental Cognitive Neuroscience, 25*, 69–91.

Eraslan, S., Yesilada, Y., & Harper, S. (2014). Identifying patterns in eyetracking scanpaths in terms of visual elements of web pages. *International Conference on Web Engineering*, 163–180.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). *A density-based algorithm for discovering clusters in large spatial databases with noise., No. 34; Vol. 96*, 226–231.

Fuhl, W., Castner, N., Kübler, T., Lotz, A., Rosenstiel, W., & Kasneci, E. (2019). Ferns for area of interest free scanpath classification. *Proceedings of the 11th ACM symposium on eye Tracking Research & Applications*, 1–5.

Haass, M. J., Matzen, L. E., Butler, K. M., & Armenta, M. (2016). A new method for categorizing scanpaths from eye tracking data. *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, 35–38.

Hartmann, M., & Fischer, M. H. (2016). Exploring the numerical mind by eye-tracking: A special issue. In *Psychological research* (Vol. No. 3; Vol. 80, pp. 325–333). Springer.

He, K., Yang, C., Stankovic, V., & Stankovic, L. (2017). *Graph-based clustering for identifying region of interest in eye tracker data analysis* (pp. 1–6). IEEE.

Hejmady, P., & Narayanan, N. H. (2012). *Visual attention patterns during program debugging with an IDE* (pp. 197–200).

Hou, Q., Cheng, M.-M., Hu, X., Borji, A., Tu, Z., & Torr, P. H. (2017). Deeply supervised salient object detection with short connections. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3203–3212.

Jarodzka, H., Holmqvist, K., & Nyström, M. (2010). A vector-based, multidimensional scanpath similarity measure. *Proceedings of the 2010 symposium on eye-Tracking Research & Applications*, 211–218.

Judd, T., Ehinger, K., Durand, F., & Torralba, A. (2009). *Learning to predict where humans look*. 2106–2113.

Kanizsa, G. (1979). *Organization in vision: Essays on gestalt perception*. Praeger Publishers.

Koć-Januchta, M., Höffler, T., Thoma, G.-B., Prechtl, H., & Leutner, D. (2017). Visualizers versus verbalizers: Effects of cognitive style on learning with texts and pictures–an eye-tracking study. In *Computers in human behavior* (Vol. 68, pp. 170–179). Elsevier.

Latimer, C. (1988). Eye-movement data: Cumulative fixation time and cluster analysis. In *Behavior research methods, instruments, & computers* (Vol. No. 5; Vol. 20, pp. 437–470). Springer.

Levenshtein, V. I., et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady, 10*, 707–710.

Li, A., & Chen, Z. (2018). Representative scanpath identification for group viewing pattern analysis. In *journal of eye movement*

research (no. 6; Vol. 11). European Group for eye Movement Research.

Manekar, S. C., & Sathe, S. R. (2018). A benchmark study of *k*-mer counting methods for high-throughput sequencing. *GigaScience, 7*(12), giy125.

Mannan, S., Ruddock, K. H., & Wooding, D. S. (1995). Automatic control of saccadic eye movements made in visual inspection of briefly presented 2-D images. *Spatial Vision*, *9*(3), 363–386.

Mathôt, S., Cristino, F., Gilchrist, I. D., & Theeuwes, J. (2012). A simple way to estimate similarity between pairs of eye movement sequences. *Journal of Eye Movement Research, 5*(1), 1–15.

Naqshbandi, K., Gedeon, T., & Abdulla, U. A. (2016). *Automatic clustering of eye gaze data for machine learning* (pp. 001239–001244). IEEE.

Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology, 48*(3), 443–453.

Paape, D., Vasishth, S., Paape, D., & Vasishth, S. (2022). Is reanalysis selective when regressions are consciously controlled? *Glossa Psycholinguistics, 1*(1).

Park, J.-S., & Oh, S.-J. (2012). A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information Science and Engineering, 28*(3), 587–600.

Parshina, O., Sekerina, I. A., Lopukhina, A., & Der Malsburg, T. von. (2022). Monolingual and bilingual reading processes in Russian: An exploratory scanpath analysis. *Reading Research Quarterly*, *57*(2), 469–492.

Peterson, J., Pardos, Z., Rau, M., Swigart, A., Gerber, C., & McKinsey, J. (2015). *Understanding student success in chemistry using gaze tracking and pupillometry* (pp. 358–366). Springer.

R Core Team. (2021). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv Preprint arXiv:1804.02767*.

Salvucci, D. D., & Goldberg, J. H. (2000). Identifying fixations and saccades in eye-tracking protocols. In Proceedings of the 2000 symposium on Eye tracking research & applications (pp. 71–78).

Shepherd, S. V., Steckenfinger, S. A., Hasson, U., & Ghazanfar, A. A. (2010). Human–monkey gaze correlations reveal convergent and divergent patterns of movie viewing. *Current Biology, 20*(7), 649–656.

Vintsyuk, T. K. (1968). Speech discrimination by dynamic programming. *Cybernetics, 4*(1), 52–57.

Von der Malsburg, T., & Vasishth, S. (2011). What is the scanpath signature of syntactic reanalysis? *Journal of Memory and Language, 65*(2), 109–127.

Von der Malsburg, T., & Vasishth, S. (2013). Scanpaths reveal syntactic underspecification and reanalysis strategies. *Language and Cognitive Processes, 28*(10), 1545–1578.

Wang, F. S., Gianduzzo, C., Meboldt, M., & Lohmeyer, Q. (2022). An algorithmic approach to determine expertise development using object-related gaze pattern sequences. *Behavior Research Methods, 54*(1), 493–507.

Xu, J., Jiang, M., Wang, S., Kankanhalli, M. S., & Zhao, Q. (2014). Predicting human gaze beyond pixels. *Journal of Vision, 14*(1), 28–28.

Huang, H. and Doebler, P. (conditionally accepted). Trajectory-based Handwriting Recognition via Spatiotemporal Convolution on Distance Matrix.
This paper has been conditionally accepted by the conference *In 2024 7th International Conference on Artificial Intelligence and Pattern Recognition*, which was held from September 20th to 22nd in Xiamen, China.

Part III

Appendix

HE HUANG

Curriculum Vitae

Department of Statistics
TU Dortmund University
(+49) 1626473829, (+86) 17830778305
huang.he2014@hotmail.com

Education

2020–present **PhD, Statistics**, *TU Dortmund University*, Dortmund, Germany.
Machine Learning in Cognitive Science

2017–2020 : **Master of Science**, *TU Dortmund University*, Dortmund, Germany.
Statistics

2015–2017 : **Bachelor of Science**, *TU Dortmund University*, Dortmund, Germany.
Statistics

2006–2010 : **Bachelor of Economics**, *Huazhong University of Science and Technology*, Wuhan, China.
Statistics (Economics)

Publications

Journal Articles

2024 He Huang, Philipp Doebler, and Barbara Mertins. Short-time aois-based representative scanpath identification and scanpath aggregation. *Behavior Research Methods*, pages 1–16. Springer, 2024.

2023 Philip Buczak, He Huang, Boris Forthmann, and Philipp Doebler. The machines take over: A comparison of various supervised learning approaches for automated scoring of divergent thinking tasks. *The Journal of Creative Behavior*, volume 57, pages 17–36. Wiley Online Library, 2023.

Under Review

Accepted on 17.07.2024 **He Huang, Philipp Doebler**, Trajectory-based Handwriting Recognition via Spatiotemporal Convolution on Distance Matrix, In *Proceedings of In 2024 7th International Conference on Artificial Intelligence and Pattern Recognition (AIPR 2024)*.

In Conference Proceedings

2020 He Huang, Martin Pouls, Anne Meyer, and Markus Pauly. Travel time prediction using tree-based ensembles. In *Computational Logistics: 11th International Conference, ICCL 2020, Enschede, The Netherlands, September 28–30, 2020, Proceedings 11*, pages 412–427. Springer, 2020.

Teaching

Summer 2023 **Case Studies I**, *Fallstudien I*.
Summer 2022 **Case Studies I**, *Fallstudien I*.
Summer 2022 **Descriptive Multivariate Statistics**, *Deskriptive Multivariate Statistik*.
Winter 2021 **Descriptive Methods**, *Deskriptive Verfahren*.
Winter 2021 **Statistics for linguists**, *Statistik für SprachwissenschafterInnen*.
Summer 2021 **Programming Course with R**.
Summer 2021 **Sampling Procedures**, *Stichprobenverfahren*.
Summer 2021 **Statistics for linguists**, *Statistik für SprachwissenschafterInnen*.
Winter 2020 **Multivariate Methods**, *Multivariate Verfahren*.
Summer 2020 **Case Studies I**, *Fallstudien I*.

Employment

2020-present **Research Assistant, Chair of Statistical Methods in the Social Sciences**, *Department of Statistics*, TU Dortmund University, Superior: Prof. Dr. Philipp Doebler.

2020-present **Research Assistant, Psycholinguistics Laboratories**, *Department of Cultural Studies*, TU Dortmund University, Superior: Prof. Dr. Barbara Mertins.

2023-present **Research Data Manager**, *Institue of From Prediction to Agile Interventions in the Social Sciences*, TU Dortmund University.

2010-2014 **Statistician**, *Department of Markets*, CSG HOLDING CO., LTD, Shenzhen. China

Computer skills

Programming Languages	Python, R, Tensorflow, Keras, Git, Shiny app, Latex
Database	SQL, MySQL
Artificial Intelligence (AI)	Deep Learning, Text Mining, Sequence Mining, Image Recognition

Languages

Chinese	Native speaker
German	10 years of using experience for living and studying in Germany
English	Skilled in listening, speaking, reading and writing

Awards

Data Mining Cup 2018 Team 5th Place

Name: He Huang Matrikelnummer: 189436

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Dissertation mit dem Titel

“Sequence Data Mining in Cognitive Science”

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht habe und die Satzung der Technischen Universität Dortmund zur Sicherung guter wissenschaftlicher Praxis in der jeweils gültigen Fassung beachtet habe. Ich versichere außerdem, dass ich die beigefügte Dissertation nur in diesem und keinem anderen Promotionsverfahren eingereicht habe und dass diesem Promotionsverfahren keine endgültig gescheiterten Promotionsverfahren vorausgegangen sind. Ferner erkläre ich, dass keine Aberkennung eines bereits erworbenen Doktorgrades vorliegt. Ich versichere an Eides statt, dass ich nach bestem Wissen die reine Wahrheit erklärt und nichts verschwiegen habe.

Dortmund, den . . . . . .

He Huang