technische universität
dortmund

Endbericht

PG568: TabScript – Handschrifterkennung auf Android-basierten Tablets

Sulejman Begovic, Rebecca Doherty
Shinazi Faruki, Daria Filatova
Nina Hesse, Dennis Kesper
Julian Kürby, Niclas Raabe
Johann Straßburg, Christian Wieprecht

4. März 2014

Betreuer: Prof. Dr.-Ing. Gernot A. Fink
Dipl.-Inf. Leonard Rothacker
M. Sc. René Grzeszick

Fakultät für Informatik
Informatik XII, Arbeitsgruppe
Mustererkennung in eingebetteten Systemen
Prof. Dr.-Ing. Gernot A. Fink

Inhaltsverzeichnis

1. Einleitung

Mobile Rechnersysteme, etwa Smartphones und Tablet-Computer, sind ein wesentlicher Bestandteil unseres Lebens geworden. Im privaten Bereich werden sie vor allem zur Unterhaltung eingesetzt, ein Einsatz als Arbeitsgerät ist aber ebenso möglich. So bietet es sich beispielsweise an, Tablet-Computer als Notizblock zu verwenden, da diese den Benutzer, im Gegensatz zu einem Laptop, nicht verdecken, und so eine bessere Kommunikation mit Gesprächspartnern ermöglichen. Ein großer Nachteil ist aber die Möglichkeit zur Texteingabe: Tablets verfügen in der Regel nur über ein sogenanntes Soft-Keyboard, also eine Tastatur auf dem Bildschirm. Durch diese gestaltet sich die Texteingabe weniger natürlich, als mit Stift und Papier, da sie beim Drücken einer Taste nicht die gleiche haptische Rückmeldung an den Benutzer liefert, wie eine physikalische Tastatur. Die Projektgruppe befasst sich aus diesem Grund mit einer Applikation, die eine natürlichere Art der Interaktion unterstützt: Handschrift. Abbildung 1.1 zeigt eine Situation, in der eine Texteingabe ohne Bildschirmtastatur getätigt wurde. Die ursprüngliche Form des Schreibens soll nicht durch das Tippen einzelner Tasten verhindert, sondern vielmehr die typische Handhaltung und der Gebrauch eines Stiftes bzw. Fingers zum Schreiben unterstützt werden.

Abbildung 1.1. – Tabscript verkörpert Handschrift als intuitive Eingabemethode auf dem Tablet-Bildschirm statt gewöhnlicher Bedienung über eine Bildschirmtastatur.

1.1. Historie der Handschrifterkennung

Der folgende Abschnitt stellt verschiedene Hardware sowie Konzepte vor, die in der Vergangenheit maßgeblich dazu beigetragen haben, die Evolution der Handschrifterkennung voranzutreiben. Manche der nachfolgenden Mechanismen bedienen sich anderer Konzepte als der heute verbreiteten Verarbeitungsschritte, doch zeigen sie, auf welche unterschiedlichen Verfahren im Laufe der Zeit gesetzt wurden und welche sich letztendlich durchgesetzt haben.

Bevor die maschinelle Erkennung von Handschrift erforscht wurde, war der Markt des *mobile computing* bereits sehr belebt. Tragbare Rechner setzten anfangs auf Hardware-Tastaturen oder Stifteingaben auf dem Display, was sich jedoch auf simple Auswahl- und Zeichenaufgaben beschränkte [MS02]. Echte Handschrifterkennung auf einem mobilen Gerät, dass eine gewisse Aufmerksamkeit erregte und dadurch den Markt für seine Nachfolger quasi eröffnete, fand 1993 mit der Vorstellung des Apple Newton (Apple MessagePad) statt.

Einen ersten kommerziellen Erfolg konnte die Firma Palm 1996 mit dem Palm Pilot feiern. Das Gerät vermied in vielerlei Hinsicht Fehler und Unzulänglichkeiten, die dem Newton einen durchschlagenden Erfolg verwährten. Dem recht unerkundeten Feld der Handschrifterkennung begegnete Palm mit der Einführung des Graffiti-Alphabets, einem buchstabenbasierten Erkennungssystem [MS02]. Dem Nutzer fiel dabei die Aufgabe zu, seinen Text Zeichen für Zeichen anhand von Gesten einzugeben, gemäß den in Abbildung 1.2 dargestellten Stiftgesten. Die verschiedenen Gesten beginnen stets an

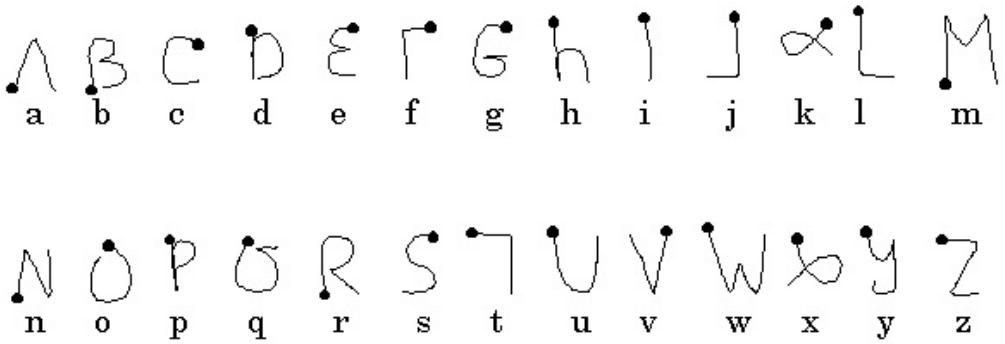


Abbildung 1.2. – Das Graffiti-Alphabet mit seinen zeichenspezifischen Gesten [MS02]. Jeder Buchstabe weist eine eindeutige Stiftbewegung auf.

dem hervorgehobenen Punkt. Bei Buchstaben mit großer Ähnlichkeit, wie zum Beispiel „u“ und „v“, steigt die Differenzierbarkeit durch jeweilige Startpunkte der Gesten auf den sich gegenüberliegenden Seiten.

Mit dieser Vorgehensweise entledigte Palm sich dem unzureichend gelösten Problem, ganze Wörter oder kursive Handschrift verarbeiten zu müssen. Gleichzeitig blühte auch ein reger Diskurs über die methodische Messbarkeit der Erkennungsgüte auf. Faktoren wie Schreibstil, Anwendungsabhängigkeit und Trainingseffekte wurden als Ursachen für verschiedene Messergebnisse identifiziert [MZ97]. Zweifellos war

der Palm Pilot jedoch in seiner Ära der populärste Vertreter seiner Art. Bis Ende des Jahres 1999 wurden über fünf Millionen Einheiten abgesetzt und ein weltweiter PDA-Marktanteil von 63 % erreicht [Sca]. Die augenscheinliche Erfolgsgarantie von zeichenbasierten Erkennungssystemen rief weitere Wettbewerber dazu auf, den Markt mit eigenen Produkten zu bereichern. So bot das von Microsoft 1996 veröffentlichte Betriebssystem Windows CE in einer späteren Version das JOT-Alphabet, welches das Graffiti-Grundgerüst um mehrere ähnliche Gesten pro Buchstabe erweiterte [MS02]. Damit fand die Zeichenerkennung Einzug auf einer großen Auswahl an Geräten, die mit diesem Betriebssystem ausgeliefert wurden.

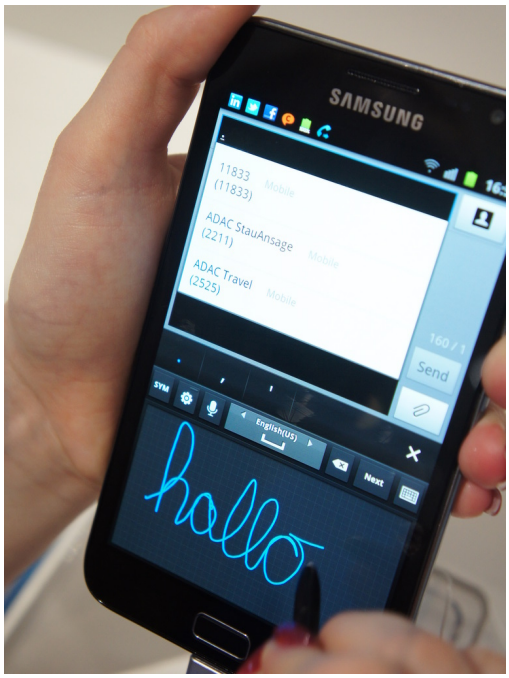


Abbildung 1.3. – Die Mixtur aus Tablet-Computer und Mobiltelefon zeichnet das Samsung Galaxy Note aus und erklärt seine recht ungewöhnliche Dimensionierung.[1]

Unterdessen wurde seitens der Forschung Fortschritte bei der Erkennung ganzer handgeschriebener Wörter und Texte erzielt. Im Jahr 2000 führten Jaeger, Manke und Waibel den NPen++ Handschrifterkenner ein, der die heute gängige Verarbeitungspipeline aufweist und damit eine kontinuierliche Stifteingabe ermöglicht [JMW00]. Ein aktueller Vertreter für mobile Geräte mit Handschrifterkennung ist das Samsung Galaxy Note, welches 2011 auf den Markt gebracht wurde. In Zusammenarbeit mit dem Grafiktablet-Hersteller Wacom entstand mit dem Galaxy Note eine Komposition aus Smartphone und Tablet Computer, die damit handschriftliche Stifteingaben von Texten in der Familie der Smartphones forcierte. Durch die Benutzung des beigefügten Stiftes

[1] http://photos.pcpro.co.uk/blogs/wp-content/uploads/2011/09/DSC02184.JPG, zuletzt abgerufen am 21.10.2013

wurde das Display größer als bei üblichen Mobiltelefonen gestaltet, wie Abbildung 1.3 zeigt.

1.2. Projektziel

Ziel der Projektgruppe ist es, eine Software zu entwickeln, mit der natürliche menschliche Handschrift zur Eingabe auf einem Android-basierten Tablet-Computer genutzt werden kann. Dabei haben wir uns dazu entschieden, eine Notiz-Applikation zu schreiben, die es dem Benutzer ermöglicht, mehrere Listen mit Aufgaben zu verwalten. Die einzelnen Aufgaben können dabei handschriftlich vom Benutzer eingegeben werden, um ihm das Eintragen zu erleichtern. Diese handschriftlichen Eingaben werden dann automatisch in Maschinenschrift übersetzt. Für die Handschrifterkennung wurde das Toolkit ESMERALDA[1] verwendet. Dieses bietet verschiedene Funktionalitäten zur Entwicklung einer Erkennungssystem für sequentielle Daten wie Handschrift.

1.3. Inhaltsübersicht

Dieser Endbericht gliedert sich im Wesentlichen in drei Bereiche. In Kapitel 2 wird zunächst die Applikation mit ihren Funktionalitäten im Detail vorgestellt. Der zweite Teil widmet sich der Handschrifterkennung. Dort werden in Kapitel 3 einige Grundlagen vorgestellt. Dann werden die einzelnen Schritte der Handschrifterkennung im Detail vorgestellt; von der Vorverarbeitung in Kapitel 4 über die Segmentierung und die Merkmalsberechnung in Kapitel 5 bis zur Erkennung in Kapitel 6. Außerdem wird das Training des Modells besprochen. Dabei wird darauf eingegangen, welche Methoden sich als besonders geeignet herausgestellt haben, und welche weniger geeignet waren. In Kapitel 7 wird eine Evaluation der einzelnen Bereiche vorgestellt, sowie eine ausführliche Bewertung der Applikation und der Erkennung.

[1] `http://sourceforge.net/projects/esmeralda/`, zuletzt abgerufen am 12.03.2013

2. Applikation

In diesem Kapitel wird die graphische Oberfläche und die Funktionalität der Applikation vorgestellt. Dabei wird vor allem auf die für den Nutzer relevanten Funktionalitäten Bezug genommen. Der technischen Umsetzung der Handschrifterkennung widmen sich die nächsten Kapitel.

2.1. Programmstruktur

Wie bereits in Abschnitt 1.2 erwähnt, wurde im Rahmen der Projektgruppe TabScript eine Applikation entwickelt, die das Erstellen und Verwalten von Aufgaben mit Hilfe von Listen ermöglicht.

Der Benutzer kann zu diesem Zweck in der graphischen Oberfläche neue Aufgabenlisten, auch *Taskliste* genannt, sowie neue Aufgaben (*Tasks*) innerhalb einer konkreten Liste erstellen. Abbildung 2.1 gibt einen Überblick über die Anwendung. Aufgaben lassen sich per Klick als „erledigt“ markieren. In einer speziellen Ansicht kann der Benutzer außerdem sogenannte *Tags* erzeugen und diese anschließend beliebigen Aufgabenlisten zuweisen. Auf diese Weise lässt sich eine Kategorisierung der vorhandenen Listen erreichen.

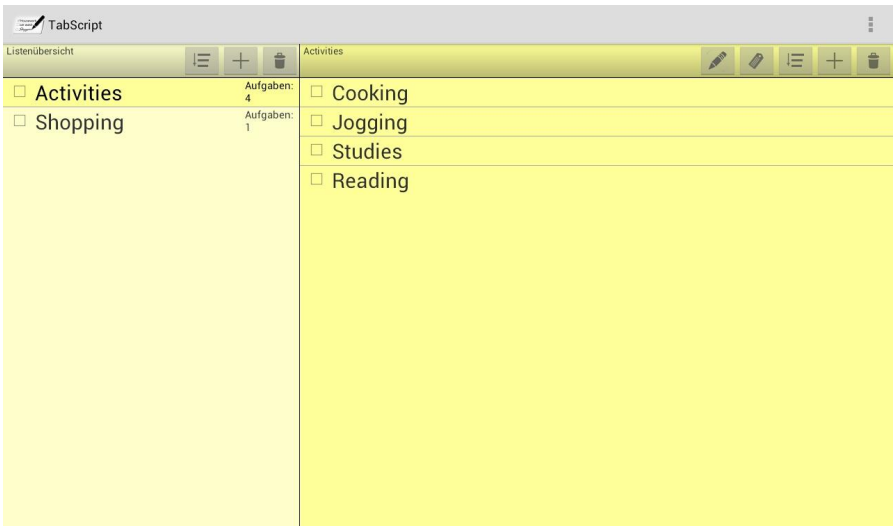


Abbildung 2.1. – Übersicht über die graphische Oberfläche der Anwendung TabScript.

Benutzereingaben erfolgen stets handschriftlich, werden durch die Anwendung erkannt sowie in Maschinenschrift angezeigt und gespeichert. Für Listen, Aufgaben und

Tags stehen neben den oben erwähnten Möglichkeiten noch eine Reihe von weiteren Verwaltungs-Optionen zur Verfügung, über die in den folgenden Abschnitten ein Überblick gegeben wird. Ein detailliertes Handbuch findet sich in Anhang A.

2.2. Startbildschirm

Die Anwendung zeigt beim Start auf der linken Bildschirmseite eine Übersicht über die bereits vorhandenen Listen an, die im folgenden als *Listenübersicht* bezeichnet wird. Sollte noch keine Liste erstellt worden sein, so ist diese Übersicht zunächst leer. Die rechte Bildschirmhälfte zeigt (durch Anklicken) für eine ausgewählte Liste der Listenübersicht alle enthaltenen Aufgaben. Abbildung 2.1 zeigt eine Listenübersicht mit zwei Listen. Die Aufgabenliste „Activities“ wurde ausgewählt, und die enthaltenen Aufgaben sind auf der rechten Bildschirmhälfte dargestellt.

Das standardmäßig integrierte Android-Menü der Anwendung (hier: obere Menüleiste, ganz rechts) bietet Optionen, mit denen es möglich ist, Tags zu verwalten oder Einstellungen an der App vorzunehmen. Weiterhin kann der Benutzer sich dort zusätzliche Informationen über die Anwendung, sowie die verwendeten externen Bibliotheken unter dem Eintrag „Über TabScript“ anzeigen zu lassen. Es ist zu beachten, dass sich das Android-Menü je nach Gerätemodell an unterschiedlichen Stellen befinden kann.

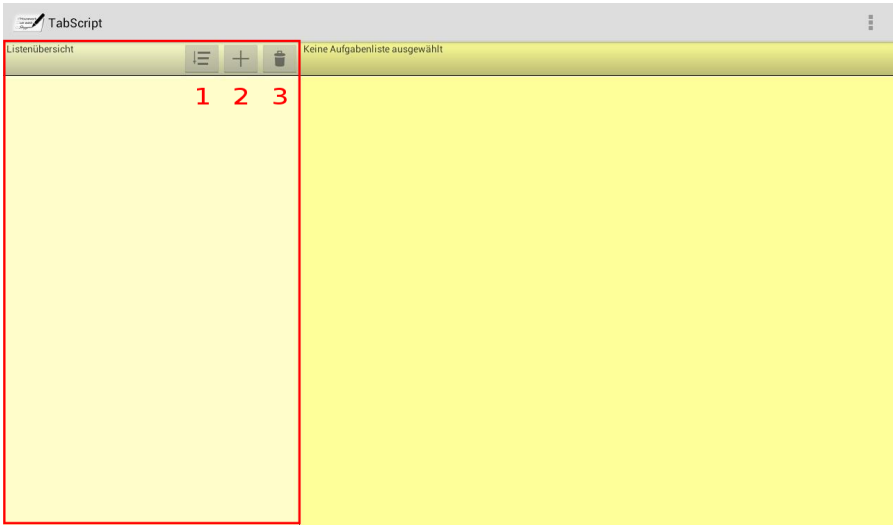


Abbildung 2.2. – Die Listenübersicht (hier rot markiert) mit ihren Menü-Buttons.

In der Listenübersicht gibt es drei Menü-Buttons, mit denen die Listen bearbeitet werden können. Die Buttons werden hier der Verständlichkeit halber mit roten Nummern versehen (siehe Abbildung 2.2). Der „Sort-Button“ (1) dient dazu, die vorhandenen Listen zu sortieren. Mit dem „Hinzufügen-Button“ (2) lässt sich eine neue Aufga-

benliste erstellen und mit Hilfe des Entfernen-Buttons (3) können eine oder mehrere Listen gelöscht werden.

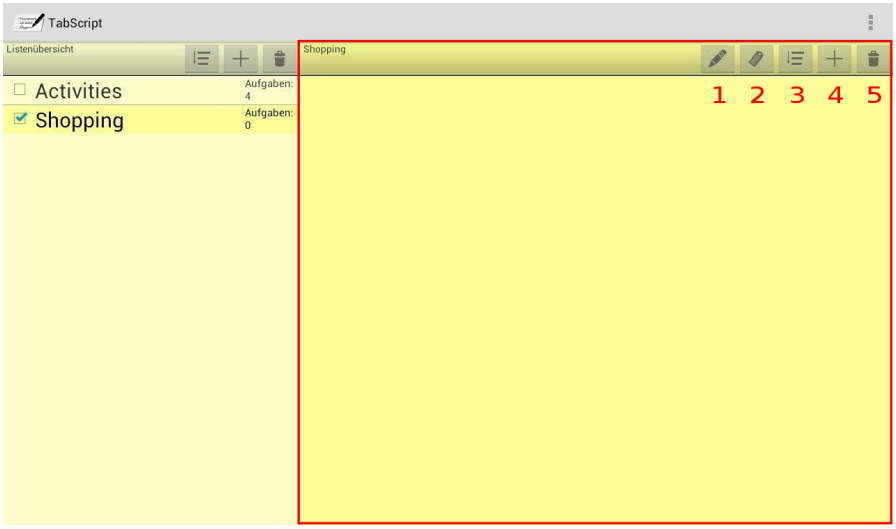


Abbildung 2.3. – Die Aufgabenübersicht (hier rot markiert) und ihre Menü-Buttons.

Die Aufgabenübersicht besitzt ebenfalls eine Menüleiste, welche dieselben Funktionalitäten für einzelne Aufgaben zur Verfügung stellt wie die Menüleiste der Listenübersicht für Listen (siehe Abbildung 2.3). Darüber hinaus enthält diese Ansicht noch zwei weitere Buttons (1 und 2). Mit Hilfe von Button 1 kann der Name der aktuell ausgewählten Liste bearbeitet werden. Button 2 ermöglicht es dem Benutzer, die ausgewählte Liste mit einem oder mehreren Tags zu versehen. Die jeweiligen Tags werden dann in der Listenübersicht direkt unter dem Listennamen angezeigt. Alle Menü-Buttons dieser Ansicht werden erst nach einem Klick auf eine Liste aus der Listenübersicht sichtbar.

2.3. Listenübersicht

In der Listenübersicht wird rechts neben dem Listennamen jeweils angezeigt, wie viele Aufgaben in einer Liste aktuell enthalten sind. Links neben dem Listennamen wird dem Benutzer ein Kästchen angezeigt, welches gegebenenfalls abgehakt ist. Dies ist der Fall, wenn sämtliche Aufgaben dieser Liste als „erledigt“ markiert wurden (siehe Abschnitt 2.4).

Möchte der Benutzer nun die Listen der Listenübersicht sortieren, so wählt er dazu den Button 1 (siehe Abbildung 2.2) aus. Es erscheint ein Pop-Up-Menü, in dem er aussuchen kann, ob die Listen nach ihrem Namen, dem Erstellungsdatum oder der Anzahl ihrer Aufgaben sortiert werden sollen. Durch einen sogenannten *Longklick*, also dem längeren Klicken auf einen Listennamen, und anschließendem Ziehen an eine andere Position in der Übersicht, kann der Benutzer einzelne Listen auch manuell verschieben.

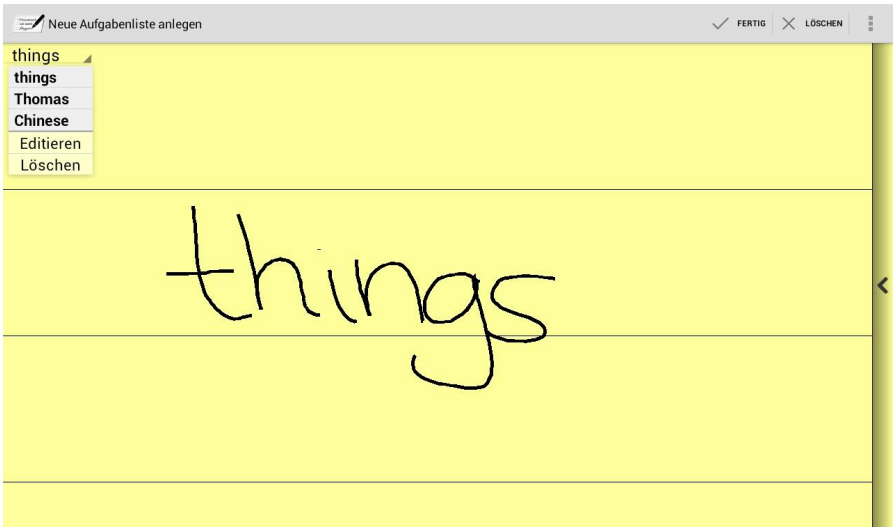


Abbildung 2.4. – Die Schreib-Ansicht der Anwendung.

Zur Erstellung einer neuen Liste aktiviert der Benutzer den mittleren Button dieser Ansicht (2) und wird in die Schreib-Ansicht weitergeleitet. In dieser Ansicht kann der Name, den die neue Liste erhalten soll, festgelegt werden. Die Schreib-Ansicht ist in Abbildung 2.4 dargestellt. Weitere Informationen zu ihrem Aufbau finden sich in Abschnitt 2.8.

Um eine oder auch mehreren Listen auf einmal zu löschen, muss der Benutzer Button 3 drücken. Anschließend wählt er durch Anklicken der Reihe nach alle Listen aus, die gelöscht werden sollen. Angewählte Listen werden bei diesem Vorgang rot hinterlegt. Um den geplanten Löschvorgang auszuführen, muss der Nutzer zur Bestätigung auf den Button mit dem Haken und der Aufschrift „Löschen“ klicken, der in der oberen Menüleiste erscheint. Es ist zu beachten, dass mit dem Löschen einer Liste auch immer alle ihre Aufgaben gelöscht werden.

2.4. Aufgabenübersicht

Ist in der Listenübersicht eine konkrete Liste ausgewählt, so erscheint der zugehörige Listen-Titel in der Kopfleiste der Aufgabenübersicht. Die enthaltenen Aufgaben sind darunter aufgelistet und können durch einen Klick auf die Box links neben ihrem Namen als „erledigt“ markiert werden.

Die Menüleiste der Aufgabenübersicht bietet, ähnlich wie die Listenübersicht im Bezug auf komplette Listen, Optionen zum Sortieren, Hinzufügen und Löschen von Aufgaben einer Liste. All diese Optionen funktionieren auf analoge Weise zu den entsprechenden Möglichkeiten der Listenübersicht. Auch hier kann der Benutzer einzelne Aufgaben durch einen *Longklick* mit anschließendem Verschieben in der Ansicht manuell an eine andere Position versetzen.

In der Aufgabenübersicht gibt es zudem zwei weitere Optionen. Zum einen kann der Benutzer über Button 1 den Namen der aktuell aktivierten Aufgabenliste in der Schreib-Ansicht bearbeiten. Des Weiteren bietet die Funktionalität hinter Button 2 eine Möglichkeit, die ausgewählte Liste mit einem Tag zu versehen.

Bei Betätigung des Buttons werden dem Nutzer alle bereits erstellten Tags zur Auswahl angeboten und die Möglichkeit, seine Wahl zu bestätigen. Anschließend können Listen nach zugewiesenen Tags gefiltert angezeigt werden. Wie diese Funktionalität aufgerufen werden kann, wird in Abschnitt 2.6 erläutert.

2.5. Aufgaben-Informationen

Durch einen Klick auf eine Aufgabe der aktuell ausgewählten Liste öffnet sich in der Aufgabenübersicht ein Fenster, das Zusatzinformationen zu der entsprechenden Aufgabe anzeigt.

An dieser Stelle wird noch einmal der Titel der Aufgabenliste, zu der die Aufgabe gehört, angezeigt. Außerdem kann der Benutzer hier den Erstellungszeitpunkt und den Status der Aufgabe einsehen. Die Aufgabe kann auch in dieser Ansicht als „erledigt“ markiert werden. Weiterhin lässt sich unter „Fälligkeitsdatum“ ein Zeitpunkt auswählen, zu dem die Aufgabe erledigt werden soll. In den Einstellungen (siehe Unterabschnitt 2.7.2) lässt sich festlegen, wie der Benutzer über das Eintreten des Fälligkeitszeitpunktes informiert werden soll.

2.6. Tag-Übersicht

Das Android-Menü der Anwendung (hier: obere Menüleiste, ganz rechts) bietet die Möglichkeit, Tags zu verwalten. Durch einen Klick auf diese Option gelangt der Benutzer in die Tag-Ansicht (siehe Abbildung 2.5), wo alle bereits erstellten Tags aufgelistet werden. In der Menüleiste dieser Ansicht gibt es die Möglichkeiten, einen neuen Tag hinzuzufügen sowie einen oder mehrere Tags gleichzeitig zu löschen. Außerdem kann an dieser Stelle eine Sortierung nach „Name“ oder „Erstellungsdatum“ durchgeführt werden. Manuell verschoben werden einzelne Tags auch hier wieder durch einen *Longklick* mit anschließendem Ziehen an eine andere Position in der Ansicht.

Um nach einem Tag zu filtern, also ausschließlich Listen in der Übersicht anzuzeigen, die mit einem bestimmten Tag versehen worden sind, genügt ein Klick auf den jeweiligen Tag, nach dem gefiltert werden soll. Diese Auswahl kann anschließend durch Klicken auf die Schaltfläche „Filter löschen“ wieder rückgängig gemacht werden.

2.7. Einstellungen

Die Einstellungen der Anwendung sind mittels des Menü-Eintrages „Einstellungen“ im Android-Menü der Anwendung zugänglich. Dort gibt es verschiedene Optionen zum Verändern des äußeren Erscheinungsbildes der Anwendung, zur Anpassung von

Abbildung 2.5. – Die Tag-Übersicht der Anwendung und ihre Menü-Felder.

Benachrichtigungen für Aufgaben und außerdem die Möglichkeit, einen Expertenmodus zu aktivieren. Dort ist es auch möglich einzustellen, welches Wörterbuch bei der Erkennung genutzt werden soll. Zur Verfügung stehen drei verschiedene Wörterbücher: Jeweils eins mit häufig genutzten deutschen und englischen Wörtern, sowie ein Wörterbuch, das die Namen von Lebensmitteln enthält.

2.7.1. Erscheinungsbild

Unter dem Eintrag „Theme“ lassen sich verschiedene Farbschemen für die Anwendung auswählen. Für die einklappbare Seitenansicht der Schreib-Ansicht, welche in Unterabschnitt 2.8.1 beschrieben wird, kann zwischen zwei verschiedenen Übergängen gewählt werden.

Weiterhin kann der Benutzer unter den gleichnamigen Menüeinträgen die Schreibfarbe, Stiftdicke und den Schärfegrad des Schriftbildes verändern. Unter „Abtastpunkte“ lässt sich einstellen, ob die Abtastpunkte der Trajektorie in der Schreib-Ansicht angezeigt werden sollen.

2.7.2. Benachrichtigungen

Der Menüpunkt „Benachrichtigungen“ kann vom Benutzer aktiviert werden, falls für Aufgaben mit Fälligkeitsdatum eine Mitteilung gewünscht wird. Ist diese Funktionalität aktiviert, so erscheinen weitere Einträge zur genaueren Konfiguration.

In den Einstellungen lässt sich der Benachrichtigungszeitpunkt vor dem Fälligkeitsdatum auswählen. Die Zeitspanne mit den Wahlmöglichkeiten reicht von 0 bis 60 Minuten. Auch für den Benachrichtigungston gibt es eine Vielzahl von Optionen. Aktiviert der Benutzer die Einstellung „Vibration bei Benachrichtigung“, so wird zum Be-

nachrichtigungszeitpunkt einer Aufgabe zusätzlich der Vibrationsalarm des mobilen Endgerätes aktiv. Ist weiterhin „LED benutzen“ ausgewählt, so leuchtet außerdem die Benachrichtigungs-LED des Gerätes (falls vorhanden) in der jeweiligen Farbe, die unter „LED-Farbe“ ausgewählt wurde.

2.7.3. Expertenmodus

Der Expertenmodus kann unter dem gleichnamigen Eintrag aktiviert bzw. deaktiviert werden. Ist diese Einstellung aktiv, so lässt sich unter „Concept“ auswählen, ob die Erkennung auf Wort- oder Zeichenebene durchgeführt werden soll. Zudem kann der Bentutzer hier die gesamte Datenbank der Anwendung zurücksetzen.

2.8. Schreib-Ansicht

Um die Benutzerfreundlichkeit zu erhöhen, wird für jede Benutzereingabe [ob sie nun zur Eingabe einer Aufgabe oder zur Eingabe eines Listen- oder Tag-Namens dient] die gleiche Schreib-Ansicht verwendet, die in Abbildung 2.4 dargestellt ist. Die Ansicht ist einer Seite aus einem Notizblock nachempfunden, und bietet Linien zur besseren Orientierung. Die interne Bezeichnung dieser Benutzerschnittstelle ist `WriteActivity`.

Für jedes Wort, das der Benutzer in der Schreib-Ansicht in handschriftlicher Form eingibt, um beispielsweise eine neue Liste zu erstellen, erscheint am oberen Rand dieser Ansicht bei Klick auf die erkannte Maschinendarstellung des Wortes ein Drop-Down-Menü. Dort kann zwischen den besten drei von der Erkennung gelieferten Wortvorschlägen gewählt werden (vergleiche Abbildung 2.4). Weiterhin ist es möglich, das jeweilige Wort über den Eintrag „Editieren“ in der Schreib-Ansicht zu bearbeiten oder über „Löschen“ komplett zu entfernen. Das Zurücksetzen der gesamten Ansicht kann der Benutzer in der Android-Menüleiste über „Löschen“ durchführen.

Am rechten Rand der Schreib-Ansicht befindet sich ein Pfeil, über den durch das Ziehen an dieser Stelle nach links, die ausklappbare Seitenansicht ausgefahren werden kann. Diese wird im nächsten Abschnitt genauer vorgestellt.

2.8.1. Ausklappbare Seitenansicht

Befindet sich der Benutzer in der Schreib-Ansicht, so kann er sich je nach Kontext (Listen-, Aufgaben- oder Tag-Erstellung) alle anderen bereits erstellten Listen, Aufgaben oder Tags anzeigen lassen. In Abbildung 2.6 sind beispielsweise die übrigen Aufgabenlisten dargestellt. Während die Seitenansicht angezeigt wird, kann auf der Schreibfläche nicht geschrieben werden. Um die Seitenansicht wieder auszublenden, wird sie an der Stelle des angezeigten Pfeiles wieder in Pfeilrichtung (nach rechts) gezogen. Es erscheint wieder die gesamte Schreib-Ansicht.

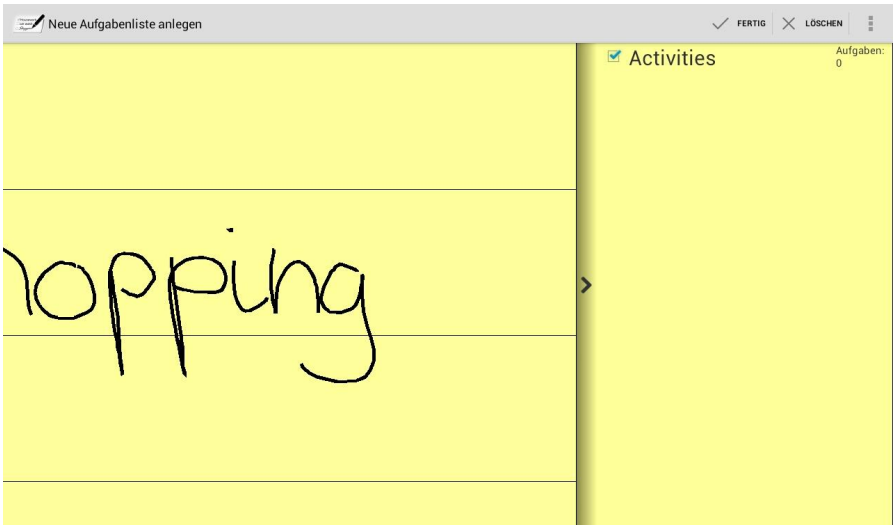


Abbildung 2.6. – Die ausklappbare Seitenansicht am rechten Bildschirmrand zeigt die bereits vorhandenen Listen, Aufgaben oder Tags an.

2.9. Abhängigkeit der GUI vom mobilen Endgerät

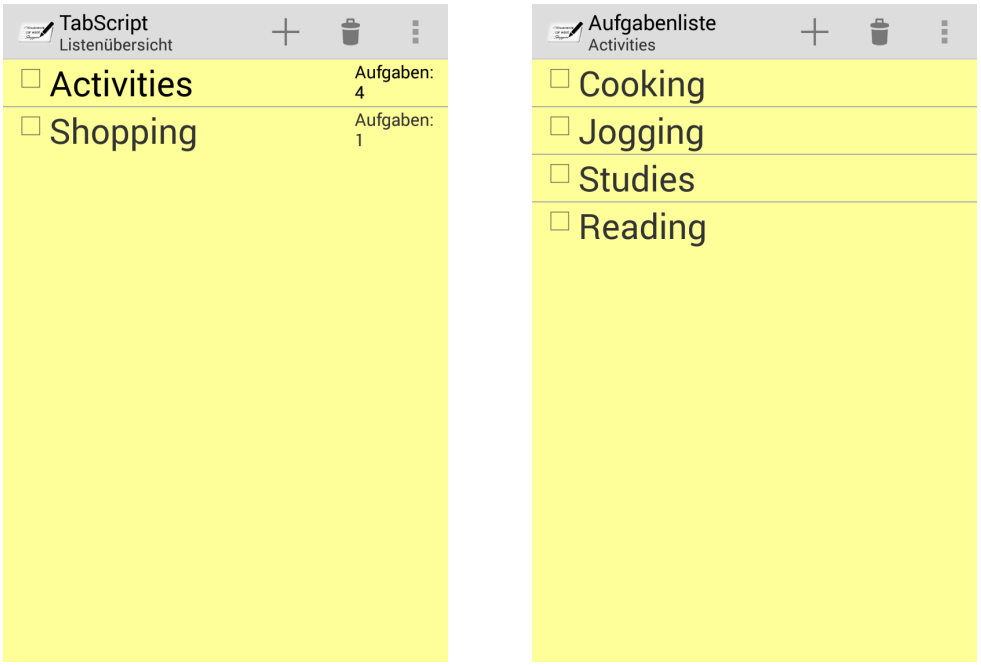
Die Anwendung soll auf verschiedenen mobilen Endgeräten, die in ihrer Größe variieren, verwendet werden können. Zu diesem Zweck unterscheidet sich die Version für große Bildschirme, wie etwa von Tablet-Computern, von der für Geräte mit kleineren Bildschirmen. Bei der Version für Tablets, welche in Abbildung 2.1 dargestellt ist, wird die Liste aller Aufgabenlisten im linken Bereich des Bildschirms angezeigt, wenn eine Aufgabenliste angewählt wurde. Da auf kleineren Bildschirmen weniger Platz vorhanden ist, wird in der Version für Smartphones die Liste von Aufgabenlisten nicht angezeigt, wenn eine Aufgabenliste ausgewählt wurde (Abbildung 2.7). Stattdessen kann zwischen den Ansichten von Aufgabenliste und der Liste aller Inhalte hierarchisch navigiert werden.

2.10. Technische Realisierung

In diesem Abschnitt gehen wir auf die technische Realisierung der Applikation ein, soweit es die Darstellung und Verwaltung der Aufgaben(-listen) betrifft. Bei der Umsetzung wurde nach dem *Model-View-Controller*-Entwurfsmuster [GHJV95] vorgegangen.

In Abbildung 2.8 sind die Model- und Controllerklassen dargestellt. Zu den Modelklassen gehören die Klassen `Task` und `TaskList`, sowie die Klasse `Tag`. Mit den `Tags` sollen die Listen kategorisiert werden. `Task` und `TaskList` beinhalten die Daten der Aufgaben resp. Aufgabenlisten. Alle Modelklassen erben von der Oberklasse `AbstractElement`, die für alle Klassen den Namen, die ID und die Position bereitstellt.

Gespeichert werden die Daten in einer SQLite-Datenbank. Die Verwaltung der SQLite-Datenbank übernimmt die Klasse `Database`, die von der Android-Klasse



(a) Aufgabenlisten (b) Aufgaben

Abbildung 2.7. – Bei kleinen Displays werden Aufgaben und Aufgabenlisten in einzelnen Activities dargestellt.

`SQLiteOpenHelper` erbt. Die Klasse ist dafür zuständig, die Modelklassen in Tabellen zu speichern, zu aktualisieren und zu löschen. Für jede Modelklasse existiert jeweils eine Tabelle. Alle Tabellen besitzen Spalten für eine ID, einen Namen und ein Position, die angibt, an welcher Stelle der Objekte das Objekt einsortiert ist. Dies spielt besonders dann eine Rolle, wenn der Nutzer manuell die Objekte über die App sortiert. Die Tabelle für die Tasklisten enthält zusätzlich noch eine Spalte, die für jede Reihe angibt, wann sie das letzte Mal aktualisiert worden ist. Tasks benötigen über die allgemeinen Informationen hinaus Spalten für die zugehörige Tasklisten-ID, ein Fälligkeitsdatum, das Datum der letzten Aktualisierung sowie die Information, ob der Task erledigt wurde, oder nicht. Die Tabelle für die Tags benötigt nur die allgemeinen Informationen Name, ID und Position. Des Weiteren existiert eine Tabelle, die die Zuordnung zwischen den Tags und ihren Tasklisten vornimmt. Daher hat diese Tabelle Spalten für die Id des jeweiligen Tags und der jeweiligen Taskliste.

Die Verwaltung der Modeldaten sowie die Verbindung zu den Viewklassen übernehmen verschiedene Controllerklassen (siehe Abbildung 2.8). Die Klasse `Controller` ist dabei der eigentliche Controller für die gesamte App. Mit dem `TaskListController` werden die Tasklisten gesondert verwaltet. Die gleiche Funktion übernimmt der `TagController` für die Tags. Außerdem existiert ein

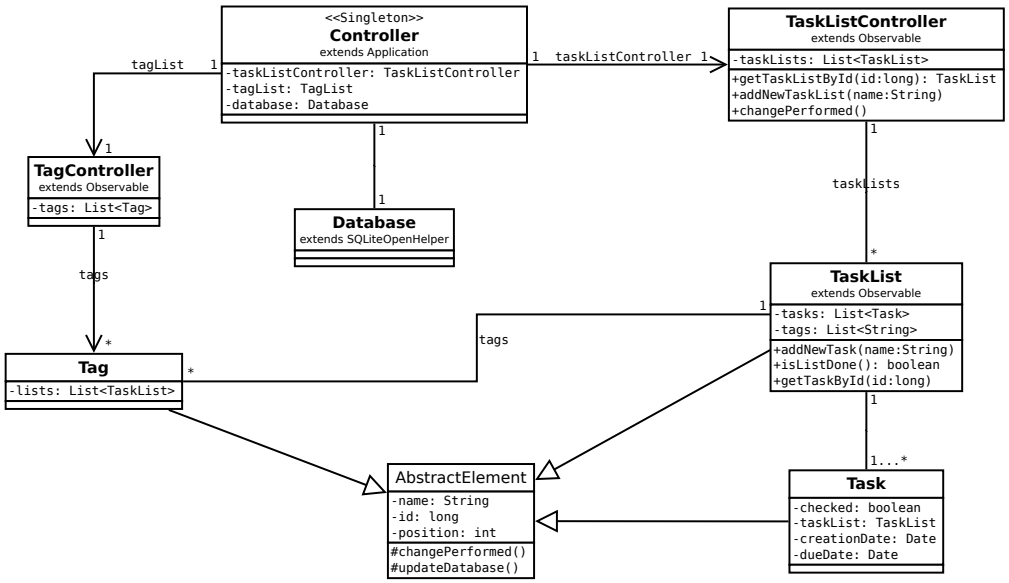


Abbildung 2.8. – Klassendiagramm für Model und Controller der App.

`TaskListTagAssociationController`, der der Tags zu den jeweiligen Tasklisten zuordnet.

Abbildung 2.9 zeigt die Klassen, die zur View gehören. Zum Anzeigen der verschiedenen Daten gibt es pro Datentyp ein eigenes `Fragment`, in dem die Daten in einer Liste dargestellt werden. Ein `Fragment` beschreibt Aussehen und Verhalten eines Teils der Benutzeroberfläche. Diese Fragments können in mehreren Activities genutzt werden. Dadurch wird zusätzlich das Anzeigen der Listen in der WriteActivity vereinfacht. Außerdem können dadurch die Listenübersicht und die ausgewählte Aufgabenliste auf großen Displays nebeneinander dargestellt werden, aber auf kleinen Displays auf zwei verschiedenen Ansichten verteilt werden.

Zum Anzeigen der Daten im Fragment verwenden wir die Klasse `DragSortListView`. Diese Klasse wird von der gleichnamigen Bibliotheken [Bau] bereitgestellt. Die Klasse `DragSortListView` erbt von der Android-eigenen Klasse `ListView` und bietet zusätzlich noch die Möglichkeit Elemente per Drag-and-Drop zu verschieben. Um die ListViews mit den Daten zu füllen, werden verschiedene Adapterklassen genutzt. Diese erzeugen aus den vorhandenen Daten Views für die einzelnen Elemente, die in den Listen angezeigt werden. Für die drei Modelklassen `Task`, `TaskList` und `Tag` existiert jeweils ein eigener Adapter. Die Adapterklassen erben von der abstrakten Klasse `AbstractAdapter`, die Methoden bereitstellt, die in allen Adaptern benötigt werden. Die Klasse `AbstractAdapter` erweitert die von Android bereitgestellte Klasse `BaseAdapter`.

Für die handschriftliche Eingabe des Benutzers ist die Klasse `WriteActivity` zuständig. Sie bietet Buttons für die Nutzerinteraktion wie beispielsweise der Fertig-Button. Die in Unterabschnitt 2.8.1 beschriebene ausklappbare Seitenansicht wird

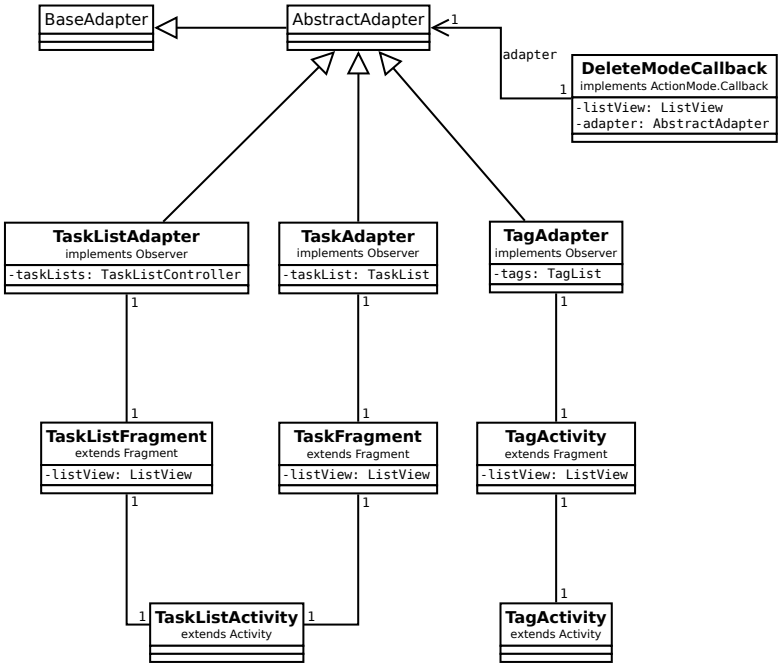


Abbildung 2.9. – Klassendiagramm für die View der App.

durch die Klasse `SlidingMenu` aus der gleichnamigen Bibliothek [Fei] bereitgestellt. Zum Anzeigen der einzelnen Listen können die oben beschriebenen Fragments wiederverwendet werden. Außerdem enthält die `WriteActivity` ein Objekt der Klasse `TrajectorySurfaceView`. In der `TrajectorySurfaceView` wird die vom Benutzer auf dem Tablet geschriebene Trajektorie aufgenommen und gespeichert. Des Weiteren überprüft die Klasse die Zeit, in der nicht geschrieben wird. Überschreitet die gemessene Zeit einen definierten Grenzwert, stößt die `TrajectorySurfaceView` die Erkennung in der `WriteActivity` an. Die tatsächliche Erkennung findet über die Klasse `HandwritingController` statt. Sie bildet die Schnittstelle zur Erkennung. Abbildung 2.10 stellt die Zusammenhänge der zuvor vorgestellten Klassen dar.

Wie in Abschnitt 2.9 beschrieben, werden bei verschiedenen Displaygrößen unterschiedliche GUI-Varianten genutzt. Daher müssen mehrere Layouts bereitgestellt werden und für diese Layouts unterschiedliches Verhalten implementiert sein. Das Bereitstellen verschiedener Layouts wird durch die Android-Architektur stark vereinfacht. Beim Erstellen einer Activity wird der Activity eine XML-Datei zugewiesen, in der die graphische Oberfläche definiert wird [Anda]. Diese Datei befindet sich normalerweise im Ordner `res/layout`. Zusätzlich kann eine alternative Datei mit gleichen Namen in anderen Ordnern (z. B. `res/layout-large`) abgelegt werden [Andb]. Anhand des Ordnernamens (z. B. `layout-large` oder `layout`) entscheidet Android beim Erzeugen der Activity, welches Layout genutzt wird. Bei großen Displays wird zuerst versucht die Layoutdefinition in `layout-large` zu nutzen. Wenn sie dort nicht existiert, wird die Datei aus `layout` genutzt. Bei kleinen Displays wird direkt die Datei aus `layout`

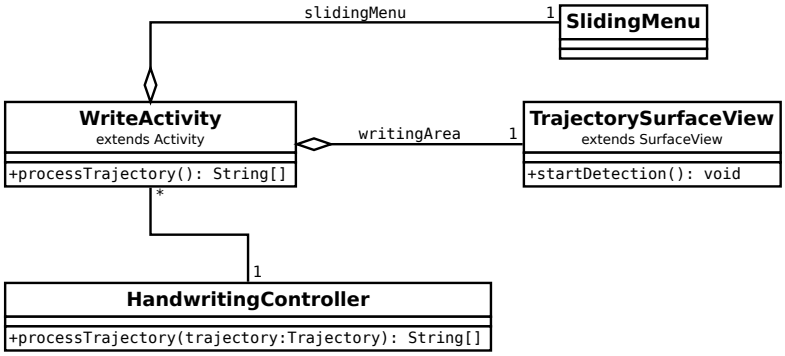


Abbildung 2.10. – Klassendiagramm für die handschriftliche Benutzereingabe.

verwendet. Dieses Verfahren nutzt die App, indem für die `TaskListActivity` zwei Layout-Dateien existieren. Zum Einen gibt es für große Displays ein Layout, das in zwei Spalten ein `TaskListFragment` und ein `TaskFragment` anzeigt. Zum Anderen gibt es für kleinere Displays ein Layout, dass nur ein `TaskFragment` enthält (vgl. Abbildung 2.1 und Abbildung 2.7).

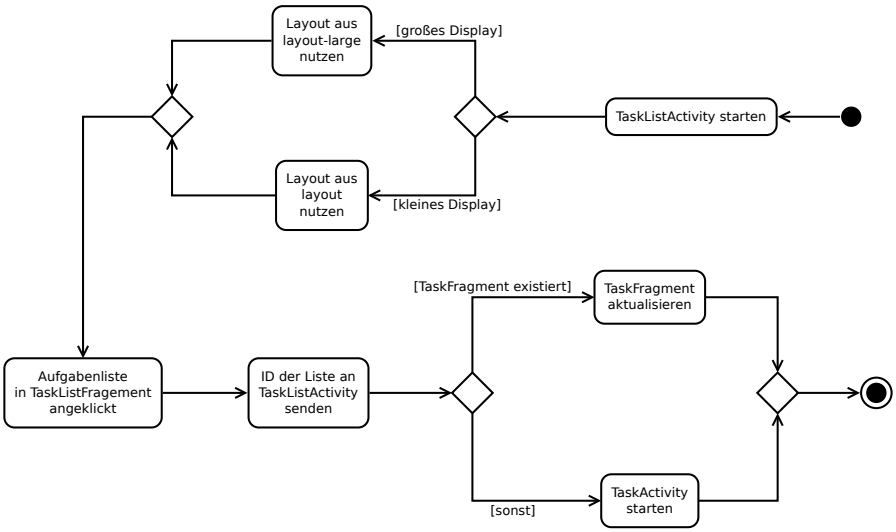


Abbildung 2.11. – Aktivitätsdiagramm zur Erzeugung der TaskListActivity und Verhalten beim Auswählen einer TaskList in Bezug auf die Unterstützung von verschiedenen Displaygrößen.

Unter Umständen müssen sich unterschiedliche Layouts unterschiedlich verhalten. Beim Anklicken einer Aufgabenliste, um die darin enthaltenen Aufgaben anzuzeigen, muss zum Beispiel das Verhalten angepasst werden. Bei großen Displays muss das `TaskFragment` aktualisiert werden, während bei kleinen Displays die TaskActivity gestartet werden muss. Abbildung 2.11 zeigt den Ablauf ei-

nes solchen Prozesses. Nachdem eine Aufgabenliste angeklickt wurde, meldet das `TaskListFragment` der `TaskListActivity` mit Hilfe der ID, welche Liste angeklickt wurde. Die `TaskListActivity` überprüft nun, ob das große oder das Standard-Layout geladen wurde. Dies geschieht, indem getestet wird, ob die `TaskListActivity` ein `TaskFragment` beinhaltet. Ist dies der Fall, kann das `TaskFragment` aktualisiert werden. Ist kein `TaskFragment` vorhanden, handelt es sich um ein kleines Display und die `TaskListActivity` startet eine neue `TaskActivity`, um die Aufgabenliste anzuzeigen.

2.11. Unterstütze Android-Versionen

Die App ist kompatibel zu Android-Versionen ab Android 3.0 (API Level 11). Einer der Gründe hierfür liegt darin, dass diese Version ein Update speziell für Tablets ist. Auch wurden mit Android 3.0 einige wichtige GUI-Elemente (z. B. `Fragments` und `ActionBar`) eingeführt. Mit einer angepassten GUI sollte die App auch auf Endgeräten mit Android 2.x funktionieren. Entwickelt und getestet wurde sie hauptsächlich auf Android 4.2 (API Level 17), der zur Zeit der Entwicklung aktuellen Android-Version.

3. Online-Handschrifterkennung

Dieses Kapitel dient der Einführung in die Handschrifterkennung, mit deren Hilfe ein nützliches Werkzeug für Textaufgaben in Android-Anwendungen entworfen wurde. Mittels handschriftlicher Texteingabe kann dem Benutzer das Tippen einzelner Buchstabentasten bei der Benutzung eines Soft-Keyboards abgenommen werden. Dazu müssen Methoden genutzt werden, die Handschrifteingaben transkribieren können.

3.1. Grundbegriffe

Nachfolgend sind einige grundlegende Bezeichnungen erklärt, um visuelle Elemente der Handschrift präzise benennen zu können. Abbildung 3.1 dient dabei als anschauliches Beispiel und die folgenden Abschnitte beziehen sich jeweils auf die dort eingeführten Begriffe.

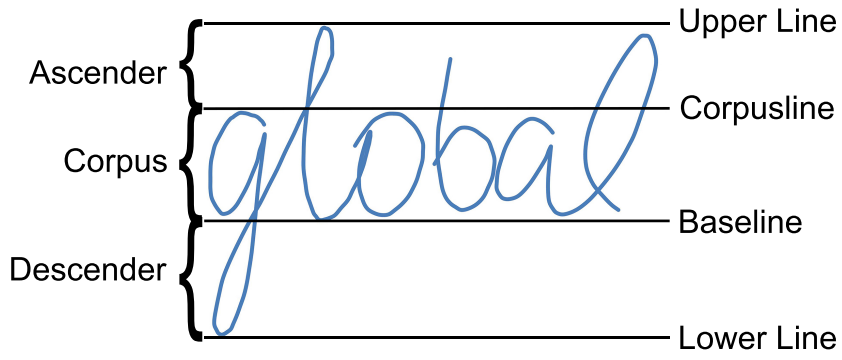


Abbildung 3.1. – Der Schriftzug „global“ ist eingebettet in Linien, die beispielsweise in Schreibheften der Grundschule Einsatz finden [JMW00].

Trajektorie. Die Trajektorie beschreibt den Stiftverlauf auf der Oberfläche des Tablets. Der sichtbare Teil der Trajektorie (*pen down*) ist als „digitale Tinte“ auf dem Tablet zu erkennen. Wird der Stift nach einer Eingabe angehoben, bezeichnet man die nachfolgenden Datenpunkte als *pen up*. Nicht alle Geräte unterstützen ein sogenanntes *pen up tracking*, also die Verfolgung des über dem Tablet erhobenen Stiftes. In Abbildung 3.1 ist der *pen-down*-Teil der Trajektorie des Wortes „global“ sichtbar. Vor und nach dem Buchstaben „b“ ist der Stift bzw. Finger abgesetzt worden und *pen ups* sind dort entstanden.

Baseline. Als Baseline (Grundlinie) bezeichnet man die Linie, auf der das Wort „global“ in Abbildung 3.1 aufsetzt. Große Druckbuchstaben setzen prinzipiell auf dieser Grundlinie auf.

Corpusline. Sie begrenzt die meisten Kleinbuchstaben nach oben hin. In Abbildung 3.1 betrifft dies die Buchstaben „g“, „o“ und „a“. Hilfreich ist die Corpusline vor allem bei der Unterscheidung von Klein- und Großbuchstaben, die in beiden Fällen das gleiche Erscheinungsbild haben, wie zum Beispiel bei „x“ und „X“.

Corpus. Der Corpus bildet genau den Bereich zwischen Baseline und Corpusline, wie Abbildung 3.1 zeigt. Die Distanz zwischen beiden Orientierungslinien ist als Corpus Height definiert.

Ascender und Descender. Die in Abbildung 3.1 dargestellten Geraden *Upper Line* und *Lower Line* begrenzen sämtliche Buchstaben und Ziffern nach oben bzw. unten. Der resultierende Oberlängenbereich zwischen Corpusline und Upper Line (Ascender) umfasst Bereiche von Buchstaben, die über die Corpusline hinaus ragen, etwa ein Stück vom „l“ in Abbildung 3.1. Analog beschreiben Unterlängen bzw. Descender den Bereich zwischen Baseline und Lower Line. Der schwungvolle Fuß des „g“ bildet folglich in Abbildung 3.1 einen Descender.

3.2. Einführung in die Handschrifterkennung

Die Idee, einen von Hand geschriebenen Text durch ein automatisches Erkennungssystem in eine digitale Repräsentation umzuwandeln, ist ein viel behandeltes Thema in der Mustererkennung. Das elektronische Lesen bzw. Durchsuchen von handschriftlichen Texten beschleunigt Prozessabläufe in vielen Anwendungsgebieten, wie etwa das automatische Einlesen von Adresszeilen auf Briefumschlägen. Es folgt eine Präzisierung des Begriffs der Handschrifterkennung und eine Beleuchtung ihrer Einsatzgebiete.

3.2.1. Handschrifterkennung: Offline vs. Online

Man unterscheidet in der Handschrifterkennung grundlegend zwischen der Offline- und der Online-Erkennung. Beim Offline-Konzept liegt in der Regel eine Bilddatei eines Textes vor. Dabei könnte es sich z. B. um einen maschinell einzulesenden Überweisungsschein oder Briefumschlag handeln [MBS99]. Ein essentieller Schritt ist, den Text vom Hintergrund zu unterscheiden, da anfangs lediglich Bilddaten vorliegen, die keinerlei Informationen zur Position der Textbausteine bereitstellen. Zur sequentiellen Verarbeitung wird der identifizierte Text nachfolgend in einzelne Zeilen unterteilt und durchläuft anschließend die in Abschnitt 3.3 vorgestellte Verarbeitungsschritte [PF09].

Die Online-Handschrifterkennung bietet hingegen eine zeitliche Komponente, die die Unterscheidung von Text und Hintergrund in einer Bilddatei unnötig macht. Während des Schreibvorgangs werden diskrete Punktdaten extrahiert, die den Verlauf des Stiftes in digitaler Form repräsentieren [PF09]. Dies geschieht z. B. bei der Nutzung eines Tablet Computers mit kapazitivem Display. Ebenso können Tafelanschriften mithil-

fe eines *tracking device* zusätzlich digital erfasst werden. Damit ist der Schreibverlauf im Gegensatz zur Offline-Erkennung explizit gegeben. Eine Herausforderung der Online-Handschrifterkennung besteht bei der Verwendung von Tablet Computern darin, dem Nutzer einen hohen Interaktivitätsgrad zu bieten, damit seine Stifteingaben möglichst unverzüglich und korrekt interpretiert werden.

3.2.2. Anwendungsgebiete

Die Offline-Handschrifterkennung erleichtert Prozessabläufe, wie z. B. das Einlesen von Adressen auf Briefumschlägen und ähnlichen standardisierten Dokumenttypen (Überweisungen, Schecks, usw.). Darüber hinaus werden heute auch Abbilder von Schriften offline verarbeitet, z. B. Videoaufnahmen von Whiteboard-Texten [LB05a].

Die Einsatzgebiete der Echtzeit-Handschrifterkennung konzentrieren sich auf mobile Geräte, wie Smartphones, Tablet Computer und ähnliche Hardware. Überall dort, wo der Nutzer Stifteingaben tätigt, die nicht einfach als Bild gespeichert, sondern als Trajektorie erfasst werden, geschieht im Hintergrund einer Anwendung Online-Handschrifterkennung.

3.3. Aufbau von Online-Handschrifterkennungssystemen

Zum grundsätzlichen Design eines Handschrifterkennungssystems zählt im Vorfeld der Programmierung die Analyse der Problemdomäne. Da man Muster aus der realen Welt erkennen möchte, ist es ratsam, die Auswahl an Mustern, die man erwartet, möglichst genau zu beschreiben. Ein Mustererkennungssystem kann auf unpassenden Daten versagen, da es schlicht nicht für den Einsatz in einem anderen Problemkreis konzipiert ist. Plötz und Fink betrachten etwa das Problem der optischen Zeichenerkennung, die zum Beispiel in der automatischen Post-Sortierung beim Lesen von Adressdaten zum Einsatz kommt, als weitgehend gelöst, während tiefergreifende Technologie, wie das automatische Lesen von kursiver Handschrift, an ihre Grenzen stößt [PF09]. Ein zeichenbasierter Entwurf würde also nicht fähig sein, seine Leistungsfähigkeit bei Blockschrift auf zusammenhängende Handschrift zu übertragen. Je detaillierter und zahlreicher die Beschreibung der erwarteten Muster ausfällt und je passender diese auf die später betrachteten Eingaben zutrifft, desto besser ist die Ausgangssituation für die Erkennungsaufgabe; ganz im Sinne von Bob Mercer: „There is no data like more data“.

Bezüglich der zugrunde liegenden Thematik der Handschrifterkennung sind Angaben über die gewählte Schriftfamilie dementsprechend unerlässlich. Ob lateinische, arabische, oder asiatische Schriftzeichen: Entwürfe eines Schrifterkenners für die eine Schriftgattung würden auf einem anderen Schriftkorpus Erkennungseinbußen verzeichnen, etwa bei der Suche nach einer Grundlinie in der Schrift [PF09]. Hier sind weiterführende Fragen über das Einsatzgebiet des Erkenners hilfreich, um eine möglichst maßgeschneiderte Lösung finden zu können:

- Muss der Text unmittelbar oder erst zu einem späteren Zeitpunkt erkannt werden? (Charakteristik der Eingabe)

- Muss das System einer gewissen Fehlerrate gerecht werden? (Sicherheitsrisiken)
- Welche Personen treten mit welchen Erwartungen an das System heran? (Nutzer-Akzeptanz)

Wenn möglichst viele derartiger Fragen formuliert und beantwortet werden können, ist eine vielversprechende Grundlage für eine solide Schrifterkennung gefunden worden.

3.3.1. Verarbeitungs-Pipeline

Aus der historischen Entwicklung der Handschrifterkennung hat sich heute eine Struktur in der Musterverarbeitung heraus kristallisiert, die in den meisten der etablierten Erkennungssysteme Einsatz findet. Die in Abbildung 3.2 dargestellte Pipeline zeigt die wesentlichen Schritte, die in der Mustererkennung vollzogen werden, jedoch spezialisiert auf die Schrifterkennungsaufgabe, deren Grundlage der Stiftverlauf auf dem Bildschirm bzw. die Trajektorie ist. In der Vorverarbeitung werden Störeffekte vermindert (siehe Kapitel 4), anschließend repräsentieren die auf Sequenzen der Trajektorie generierten Merkmale die jeweiligen Schriftabschnitte. Gebräuchliche Merkmale für Handschrifterkennung werden in Kapitel 5 beschrieben. Im letzten Schritt geschieht bei der Klassifikation die Zuordnung der gelesenen Eingabe zu einer Klasse von Schriftzeichen bzw. deren Bestandteilen. Dazu kann unter anderem ein Hidden Markov Model (siehe Kapitel 6) genutzt werden, welches vor seinem Gebrauch Merkmalsrepräsentationen der Zeichen anhand einer annotierten Stichprobe erlernt. Die dargestellten Schritte basieren auf der allgemeinen Darstellung für Klassifikationsaufgaben [Nie83].

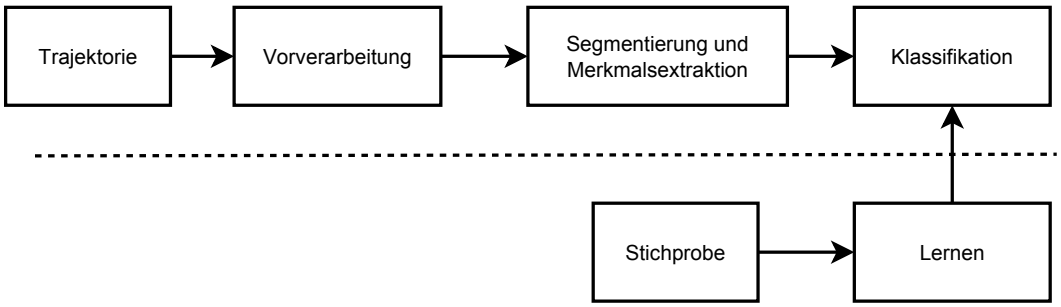


Abbildung 3.2. – Die geläufigen Schritte in der Verarbeitung von Trajektorie sind Aufnahme der Trajektorie, diverse Vorverarbeitungsmaßnahmen, Segmentierung in einzelne Stücke zur Merkmals-Extraktion und Klassifikation auf Basis einer erlernten Stichprobe (vgl. [Nie83]).

Das Ziel dieser Projektgruppe ist die Erkennung von Handschrift auf mobilen Rechnersystemen; somit beruht das Softwaregrüst auf dem in Abbildung 3.2 dargestellten Schema für Online-Handschrifterkennung. Um dem Nutzer möglichst großen Freiraum beim Schreiben zu lassen, soll die Nutzereingabe per Schrift weitgehend restriktionslos sein; lediglich eine horizontale Grundlinie soll dem Anwender zu einem möglichst geraden Schriftbild verhelfen.

3.3.2. Trainingsdaten

Ein essentieller Bestandteil bei der Entwicklung einer Handschrifterkennung ist ein annotierter Datensatz von Schriftproben, der zum Trainieren des Hidden Markov Model herangezogen wird. Dieser Schritt ist notwendig, um dem Klassifikator Kenntnisse über Eigenschaften der in der Anwendung zu erwartenden Schriftzeichen zu geben. Da die Handschrifterkennung aus den zu Grunde liegenden Trainingsdaten heraus erstellt wird, ist sie für Muster geeignet, die den Trainingsdaten ähneln. Ein mit arabischen Schriftzeichen und Wörtern trainiertes Modell wäre beispielsweise nicht in der Lage, in der Praxis auf dem lateinischen Alphabet erfolgreich arbeiten zu können. Auch der Umfang der Trainingsdatenbank ist ein einflussreicher Faktor bezüglich der Erkennungsleistung [LB05b]. Da auf einem Tablet sowohl (kursive) Handschrift als auch Blockschrift erwartet werden können, müssen beide Schriftstile in der ausgewählten Datenbank vertreten sein; dem wird durch die Nutzung der *Unipen*-Datenbank Rechnung getragen.

Das *Unipen*-Konsortium stellt einen Zusammenschluss von 40 Firmen und Instituten dar, die sich unter dem Konzept eines vereinheitlichten Datenbankformats das Ziel gesetzt haben, eine umfangreichere Datenansammlung zu etablieren, als öffentlich zugängliche Datenbanken sonst bieten [Uni]. Das Ergebnis ist eine seit 2011 frei erhältliche Datenbank von Online-Handschriftdaten, bestehend aus vielen Beiträgen der involvierten Institutionen, die sich zu einer Gesamtheit von über 80.000 annotierten Wort-Stichproben von über 2200 verschiedenen Probanden zusammenfügen. Des weiteren sind auch Städtenamen und Adresszeilen integriert. Da das gewählte *Unipen*-Format keine Restriktionen bezüglich der Aufnahme-Hardware stellt, sind die enthaltenen Trajektorien qualitativ unterschiedlich zu bewerten.

Die im Vergleich zu anderen Handschrift-Datenbanken immense Größe bedeutet im Fall der *Unipen*-Datenbank auch, dass verschiedene Hardware zur Aufzeichnung der Schriftzüge eingesetzt wurde. Letztendlich ist das *Unipen*-Projekt eine Sammlung verschiedener ursprünglich vertraulicher Datensätze, die im Zeitraum von 1990 bis 1996 erstellt und unter einem vereinbarten Standard schließlich publik gemacht wurden. Mehrheitlich mit 52% fanden die zeitgenössischen Modelle der Firma Wacom (unter Anderem Wacom PL-100V, Wacom SD-xxx und Wacom HD-xxx) ihren Einsatz zur Digitalisierung der Handschriften. Bei 10% der Daten ist der Tablet-PC NCR 3125 genutzt worden; der verbleibende Rest von 38% der Trajektorien wurde auf Eigenbauten, Versuchsgeräten oder weniger verbreiteten kommerziellen Geräten produziert.

Auf Grundlage dieser Beobachtung haben wir die zur Verfügung stehenden Datensätze gefiltert und jene aussortiert, die auf Geräten mit zu geringer Auflösung, zu geringer Abtastrate oder bei unbekanntem Versuchsaufbau angelegt worden sind. Dabei gab es Konfigurationen mit Pixeldichten von 80 ppi (Pixel per inch) bei einem unbekannten Versuchsaufbau bis hin zu 1000 ppi auf dem Wacom 420-L Grafiktablet. Weiterhin musste die Ausgewogenheit verschiedener Schriftstile, wie Druckschrift, Kursivschrift etc., aufrecht erhalten werden. Insgesamt reduzierte sich das verwendete Trajektorien-Volumen auf etwa 40.000 annotierte Wort-Stichproben, was einem Anteil von 51,9% der Gesamtdaten entspricht.

4. Vorverarbeitung

Die zu erwartenden Eingaben auf einem Tablet Computer unterliegen gewissen Störeffekten, denen mit einigen Vorverarbeitungsschritten bereits vor der eigentlichen Klassifikation entgegengewirkt werden kann. Abbildung 4.1 stellt diese Schritte schematisch in den Zusammenhang der gesamten Verarbeitungsreihenfolge. Dieses Kapitel erklärt die angewandten Methoden der Vorverarbeitung. Da zum Teil in der Testphase verschiedene Ansätze erprobt wurden, werden diese ebenfalls erklärt. In Abschnitt 4.2 wird darauf eingegangen, welche Methoden tatsächlich in TabScript zum Einsatz kommen.

Das Ziel der Vorverarbeitung ist, die aufgenommene Trajektorie geeignet aufzubereiten, um im Anschluss verschiedene Merkmale zu berechnen. In der Regel werden die Trajektorien unterschiedlicher Schreiber stark in ihrem Erscheinungsbild variieren. Je nachdem, mit welcher Hand geschrieben wurde, ist die Schrift mehr nach links oder rechts geneigt. Die Anzahl und der Abstand der Abtastpunkte hängt stark von der Schreibgeschwindigkeit ab. Es sind noch weitere Abwandlungen vorstellbar, beispielsweise eine Variation der Größe oder eine nicht waagerecht ausgerichtete Schrift. Die Vorverarbeitung wird eingesetzt, um die unterschiedlichsten Trajektorien so zu präparieren, dass sie ein uniformes Erscheinungsbild aufweisen. Dadurch ist die Variation in den im Anschluss berechneten Merkmalen geringer, und die Klassifikation wird erleichtert.

4.1. Verfahren

Die Vorverarbeitung beginnt zunächst mit einer Glättung der aufgezeichneten Trajektorie. Anschließend wird eine Korrektur der Schriftsteigung (*slope*) und eine Nor-

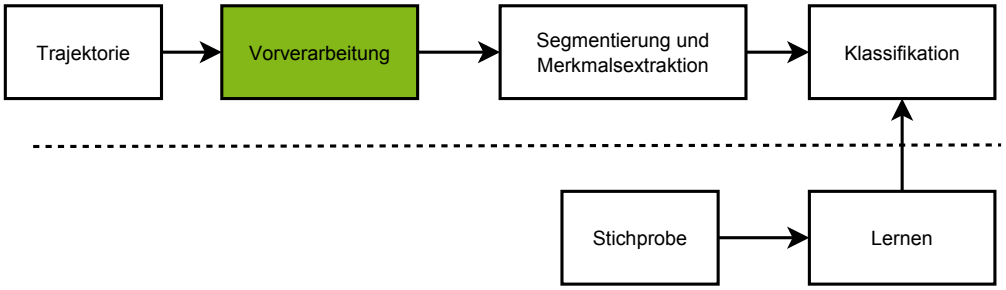


Abbildung 4.1. – Verarbeitungschritte der Trajektorie (vgl. [Nie83]).

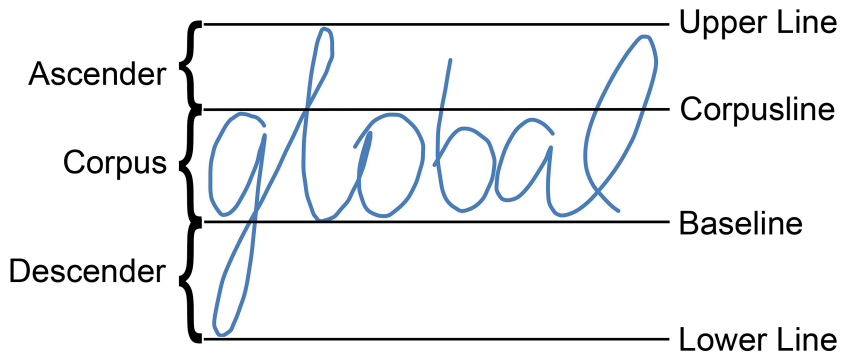


Abbildung 4.2. – Die verschiedenen Linien anhand des Wortes „global“ visualisiert (vgl. [JMW00]).

malisierung der Schriftgröße durchgeführt. Um später eine korrekte Segmentierung und Merkmalsextraktion zu gewährleisten, wird eine Trajektorie mit äquidistanden Abtastpunkten benötigt. Zu diesem Zweck wird die aufgenommene Trajektorie nach der Normalisierung der Schriftgröße neu abgetastet. Zuletzt erfolgt die Berechnung der sogenannten Base- und Corpusline, welche, wie in Abbildung 4.2 dargestellt, den Kern des Wortes von seinen Ober- bzw. Unterlängen abgrenzen. Diese Linien werden ebenfalls für die spätere Merkmalsberechnung benötigt (siehe Kapitel 5).

4.1.1. Glättung der Trajektorie

Vor Anwendung aller anderen Vorverarbeitungsschritte wird die Trajektorie geglättet. Die Glättung (engl. *Smoothing*) dient der Beseitigung von „Zittern“ im Trajektorienverlauf [JMW00], beispielsweise verursacht durch eine zitterende Hand bei der Handschrifteingabe oder ein zu grobes Raster bei der Punkterfassung durch die Hardware. Um die Trajektorie zu glätten wird ein Tiefpassfilter angewendet, bei dem jeder Punkt $(x(t), y(t))$ durch einen neuen Punkt $(x'(t), y'(t))$ mit den Koordinaten

$$x'(t) = \frac{1}{4} \cdot x(t-1) + \frac{1}{2} \cdot x(t) + \frac{1}{4} \cdot x(t+1)$$
$$y'(t) = \frac{1}{4} \cdot y(t-1) + \frac{1}{2} \cdot y(t) + \frac{1}{4} \cdot y(t+1)$$

ersetzt wird. Dies entspricht der Ersetzung eines Punktes durch den gewichteten Mittelwert seiner jeweiligen Nachbarn (diskreter Gauß-Filter). Der erste und der letzte Punkt der Trajektorie werden nicht geglättet, sondern unverändert übernommen.

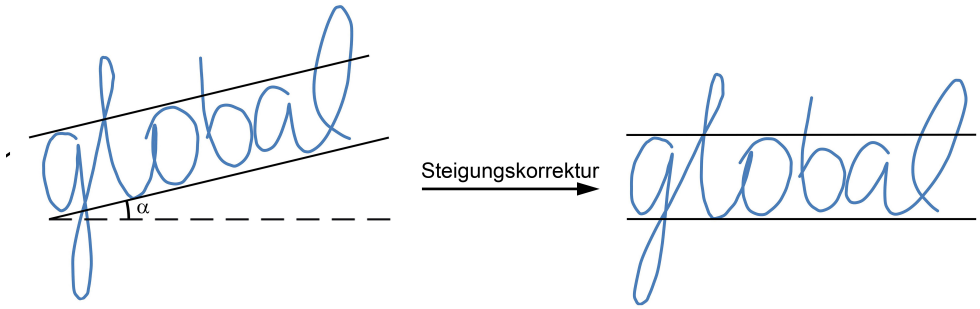


Abbildung 4.3. – Die Steigungskorrektur am Beispiel des Wortes „global“ (vgl. [JMW00])

4.1.2. Korrektur der Steigung

Nach erfolgter Glättung der Trajektorie wird eine Korrektur der Schriftsteigung (*slope*) durchgeführt. Ziel dieses Schrittes ist es, den Schriftzug horizontal auszurichten. Das Verfahren kann in mehrere Schritte unterteilt werden:

1. Zunächst werden alle lokalen Extremstellen der Trajektorie berechnet. Die Lokalität ist durch eine Nachbarschaft von 18 Punkten definiert.

2. Anhand der im vorangegangenen Schritt identifizierten Minima wird mittels der Methode der kleinsten Quadrate eine lineare Regressionsgerade berechnet.

3. Die Steigung α der Regressionsgerade wird als die Steigung des Schriftzuges angenommen. Daher wird die Trajektorie um diesen Winkel um den Ursprung gedreht. Jeder Punkt (x, y) der Trajektorie wird dabei auf einen neuen Punkt (x', y') mit

$$x' = x_{min} + \cos(\alpha) \cdot (x - x_{min}) - \sin(\alpha) \cdot (y - y_{min})$$
$$y' = y_{min} + \sin(\alpha) \cdot (x - x_{min}) + \cos(\alpha) \cdot (y - y_{min})$$

   projiziert. Dabei sind x_{min} und y_{min} die Koordinaten des linkesten unteren Punktes der Trajektorie.

In Abbildung 4.3 ist ein Beispiel einer Steigungskorrektur anhand des Wortes „global“ visualisiert.

4.1.3. Größennormalisierung

Eine Normalisierung der Größe ist erforderlich, da die Schrittweite der Neuabtastung direkt von der Höhe der Trajektorie abhängt. Zwei Ansätze zur Normalisierung wurden für TabScript umgesetzt. Zum einen kann die Bounding Box der Trajektorie berechnet und auf eine feste Höhe skaliert werden. Die Bounding Box ist durch die vier maximalen Eck-Koordinaten der Trajektorie bestimmt (Minimum und Maximum

in x- und y-Richtung). In TabScript ist eine Größennormalisierung implementiert, die die Trajektorie auf eine Höhe von 40000 skaliert. Der Nachteil dieser Methode ist allerdings die Abhängigkeit von Ascendern und Descendern (siehe Abschnitt 3.1). Enthält die Trajektorie beispielsweise ein „g“ mit großer Unterlänge, so dominiert diese die Größenskalierung. Ist die Unterlänge kleiner, so nimmt der Textcorpus mehr Raum der skalierten Trajektorie ein. Alternativ kann die Größennormalisierung durch Skalierung der Corpushöhe erfolgen, d.h. dem Abstand zwischen Base- und Corpusline; dies ist in Abbildung 4.4 dargestellt.

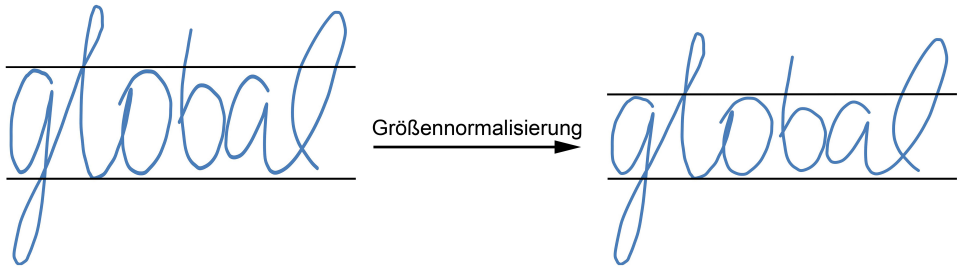


Abbildung 4.4. – Die Größennormalisierung anhand der Corpushöhe am Beispiel des Wortes „global“

4.1.4. Neuabtastung

Da die Punkte der Trajektorie bei der Aufnahme durch das Tablet zeitlich konstant erfasst werden, unterscheidet sich der Punktabstand je nach Schreibgeschwindigkeit. Bei der Neuabtastung wird die Trajektorie so abgetastet, dass die neuen Punkte räumlich äquidistant sind. Auf diese Weise werden die Einflüsse von Schreibgeschwindigkeit des Schreibers und Abtastrate der Hardware in der Trajektorie eliminiert [JMW00]. Als neuer Abstand werden in TabScript 300 Pixel verwendet. Abbildung 4.5 veranschaulicht an einem Beispiel, wie der schwankende Abstand zwischen den von der Hardware gelieferten Punkten durch eine Abtastung mit festem Abstandsmaß behoben wird.

4.1.5. Berechnung von Base- und Corpusline

Wie bereits genannt, zeigt Abbildung 4.2 die Base- und Corpusline eines beispielhaften Schriftzuges. Dabei markiert die Baseline die Standlinie der Buchstaben bzw. die Grundlinie, auf der das Wort geschrieben wurde. Im Gegensatz hierzu beschreibt die Corpusline eine Gerade, welche die Höhe der Kleinbuchstaben ohne Oberlänge (hier: „o“, „a“) anzeigt.

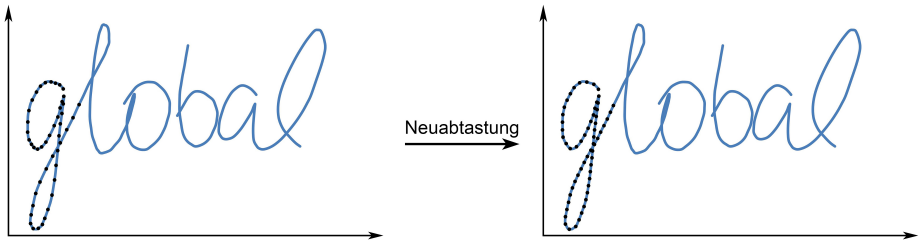


Abbildung 4.5. – Erneute Abtastung der Trajektorie zur Erzeugung äquidistanter Punktdaten eliminiert Einflüsse, wie Schreibgeschwindigkeit, Abtastrate, usw.

Eine Möglichkeit zur Berechnung der Baseline ist, zunächst die lokalen Minima der Trajektorie zu ermitteln. Anhand dieser Punkte wird anschließend nach der Methode der kleinsten Quadrate eine lineare Regressionsgerade errechnet. Analog wird die Corpusline durch Berechnung der lokalen Maxima bestimmt.

Eine weitere Möglichkeit ist die Berechnung der Base- und Corpusline mithilfe eines vertikalen Dichtehistogramms der Trajektorie (siehe [AzH09]). Das Verfahren kann in folgende Schritte unterteilt werden:

1. Berechnung eines vertikalen Dichtehistogramms.
2. Anwendung eines Schwellwertverfahrens.
3. Festlegung der Base- und Corpusline.

Im ersten Schritt wird ein vertikales Dichtehistogramm $h_0(y)$ berechnet mit $y \in [0, h_{tam}]$, wobei h_{tam} die Höhe der Trajektorie beschreibt. Wie in Abbildung 4.6(a) dargestellt, gibt das Dichtehistogramm für jeden Wert der y-Achse die Anzahl der Abtastpunkte der Trajektorie wieder, welche selbigen y-Wert aufweisen. Anschließend wird das Dichtehistogramm geglättet:

$$s_0(y) = \sum_{i=-\Delta y}^{i=\Delta y} h_0(y+i). \tag{4.1}$$

Der Grad der Glättung lässt sich dabei mithilfe des Parameters Δy einstellen. Je größer die Trajektorie ist, desto größer sollte auch Δy gewählt werden. Für Texte in Lateinischer Schrift wird der Wert $\Delta y = 0,1 \cdot h_{tam}$ empfohlen [AzH09].

Im nächsten Schritt wird mithilfe des Otsu-Verfahrens [Ots79] ein Schwellwert $t \in [0, h_{tam}]$ berechnet. Dann wird das Maximum des Dichtehistogramms bestimmt und die Menge der Schnittpunkte zwischen Dichtehistogramm und Schwellwert ermittelt:

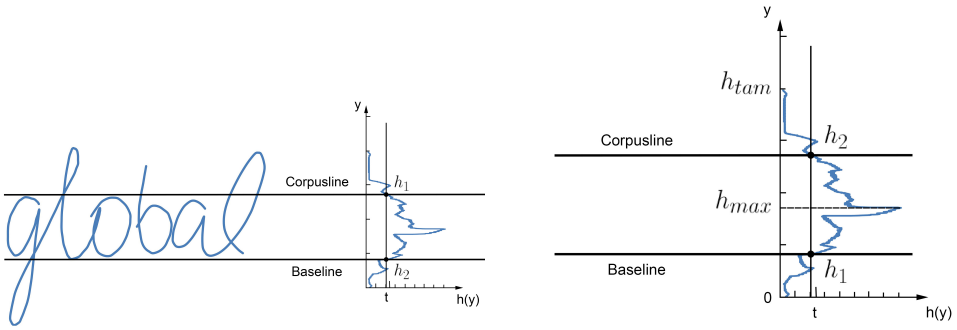
$$\mathcal{M} = \{h \in [0, h_{tam}] | h = t\}. \tag{4.2}$$

Aus der Menge $\mathcal{M}$ werden anschließend zwei Punkte h_1, h_2 mit $h_1 < h_{max}$ und $h_2 > h_{max}$ so gewählt, dass sie den geringsten Abstand zum Maximum des Dichtehistogramms h_{max} aufweisen, d.h.

$$\forall h_{m1} \in \mathcal{M}_1 : h_{m1} \leq h_1 \text{ mit } \mathcal{M}_1 = \{h \in \mathcal{M} | 0 \leq h \leq h_{max}\} \text{ und} \tag{4.3}$$

$$\forall h_{m2} \in \mathcal{M}_2 : h_{m2} \geq h_2 \text{ mit } \mathcal{M}_2 = \{h \in \mathcal{M} | h_{max} \leq h \leq h_{tam}\}. \tag{4.4}$$

Base- und Corpusline werden so bestimmt, dass sie parallel zur $h(y)$-Achse des Dichtehistogramms bzw. zur Schreibrichtung der Trajektorie verlaufen und die Punkte h_1 bzw. h_2 schneiden (vgl. Abbildung 4.6(b)).



(a) Base- und Corpusline (b) vertikales Dichtehistogramm

Abbildung 4.6. – Base- Corpusline Berechnung nach [AzH09].

4.1.6. Neigungskorrektur

Unter der Neigung der Trajektorie versteht man, wie kursiv die einzelnen Buchstaben geschrieben werden. Um die Neigung zu korrigieren, also die Trajektorie aufzurichten, werden zunächst alle Winkel zwischen jeweils zwei aufeinanderfolgenden Punkten berechnet. Die Winkel werden dabei mit dem Abstand zwischen den beiden Punkten gewichtet und in Schritten von jeweils zehn Grad zusammengefasst. Der am häufigsten vorkommende Winkel wird als die Neigung des Schriftzuges angenommen, und die gesamte Trajektorie wird um ihn geschert. In Abbildung 4.7 ist das Wort „global“ mit starker Neigung und mit korrigierter Neigung zu sehen.

4.1.7. Entfernen von Delayed Strokes

Ein weiterer Bestandteil der Vorverarbeitung ist die Behandlung von *delayed strokes*, welche zeitlich verzögerte Stifteingaben an einem Zeichen darstellen, etwa der Punkt über „i“ oder der horizontale Strich durch „t“. Man kann schlecht voraussagen, wann ein solches nachträgliches Korrekturereignis auftritt, wie es Hürst, Yang und Waibel bezeichnen [HYW98]. Da im Allgemeinen aber kontinuierliche Schreibvorgänge erwartet

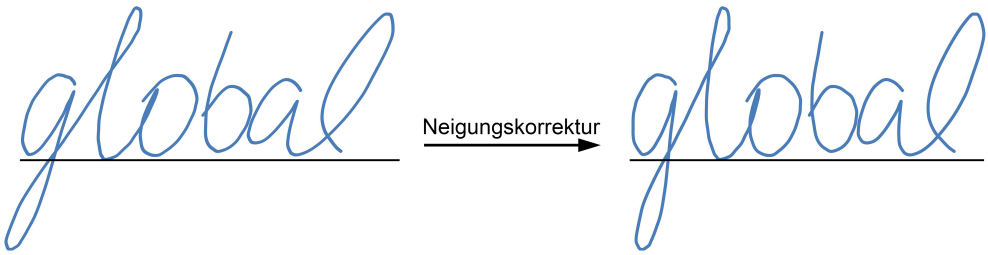


Abbildung 4.7. – Die Steigungskorrektur anhand des Wortes „global“. Im Anschluss sind alle Buchstaben sichtbar mehr aufgerichtet. (vgl. [JMW00])

werden, sind solche lokalen Rücksprünge ein Bruch mit der Annahme, dass die gelesene Trajektorie unverändert bleibt. Verschiedene Ansätze bedienen sich der Offline-Handschrifterkennung [Man98] oder löschen *delayed strokes* aus der Trajektorie mit einem Vermerk für die Weiterverarbeitung [JMW00].

4.2. Verwendete Vorverarbeitungsschritte

Von den im vorangegangenen Abschnitt vorgestellten Vorverarbeitungsschritten werden die folgenden in der finalen Applikation eingesetzt:

- Glättung,
- Skalierung,
- Steigungskorrektur,
- Neigungskorrektur und
- Neuabtastung.

Für die Skalierung wird die anhand der Bounding Box berechnete Höhe der Trajektorie benutzt. Die Zielhöhe beträgt 40000 Pixel. Zur Steigungskorrektur wird die Baseline auf Grundlage der lokalen Minima der Trajektorie extrahiert und anschließend iterativ rotiert.

Die oben genannten Methoden haben sich in der beschriebenen Reihenfolge als am erfolgreichsten im Hinblick auf das Erkennungsergebnis erwiesen. In Abschnitt 7.2 wird diese Entscheidung im Detail beleuchtet.

5. Merkmalsberechnung

Die Merkmalsberechnung ist für die erfolgreiche Handschrifterkennung ein essentieller Bestandteil, der bei der semantischen Unterscheidung unterschiedlicher Handschriftdaten hilft. Die berechneten Merkmale bilden dabei eine Komponente, die es ermöglicht, effizient Schrift klassifizieren zu können. Die handgeschriebene Trajektorie bekommt erst durch die Merkmalsberechnung Werte zugewiesen, die der Klassifikator verarbeiten kann. Eine Einordnung der Merkmalsberechnung in die Verarbeitungspipeline ist in Abbildung 5.1 gegeben.

5.1. Ziel der Merkmalsberechnung

Ein Merkmal ist ein „charakteristisches, unterscheidendes Zeichen, an dem eine bestimmte Person, Gruppe oder Sache, auch ein Zustand erkennbar wird“ [Dud]. Im Zusammenhang mit der Handschrifterkennung werden Merkmale benutzt, um Gemeinsamkeiten und Unterschiede innerhalb der Handschrift zu finden.

Wenn zwei geschriebene Worte miteinander verglichen werden, so kann ein Leser diese sehr schnell unterscheiden oder Gemeinsamkeiten finden. Zwei ausgeschriebene Worte mit dem Inhalt „Schrift“ werden so in den Zusammenhang gebracht, dass sie als gleich betrachtet werden, da sie die gleiche Semantik, sprich die gleiche Bedeutung haben. Sind die Schreiber der beiden Wörter unterschiedlich, so ist es wahrscheinlich, dass die ausgeschriebenen Worte „Schrift“ ein abweichendes Erscheinungsbild haben. Für den Leser bleiben die Bedeutungen der Wörter gleich, doch er ist ebenso in der Lage die Unterschiede und Gemeinsamkeiten in den Schriftbildern auszumachen. Die Unterschiede können darin bestehen, dass einzelne Buchstaben unterschiedlich ausgeschrieben werden, indem beispielsweise Kurvenzüge geschwungener oder spitzer

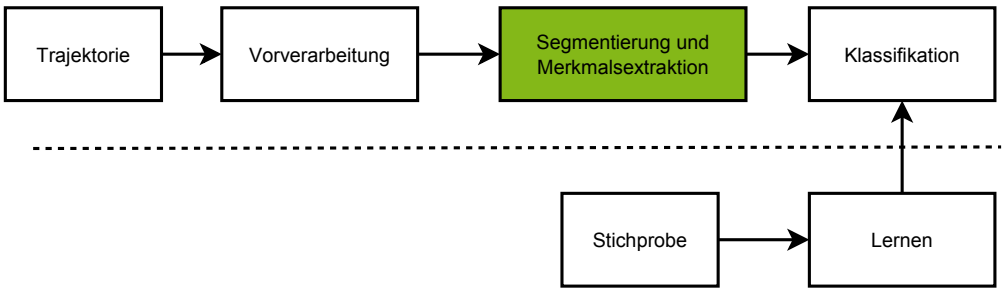


Abbildung 5.1. – Verarbeitungschritte der Trajektorie [Nie83].

sind. Dennoch sind die Unterschiede häufig gering genug, sodass zwei Buchstaben mit unterschiedlichem Schreibstil die gleiche Semantik zugeordnet werden kann.

In diesem Kontext geht es in der Merkmalsberechnung um die Berechnung von Merkmalen auf Schriftbildern, die es später ermöglichen soll, Schriftabschnitte miteinander vergleichen und diesen Abschnitten Semantiken zuordnen zu können. Für das Auge mag bereits ein ganzes Wort ein Merkmal sein, das einer Semantik zugeordnet und mit anderen Wörtern verglichen werden kann. Bei der Online-Handschrifterkennung jedoch ist ein ganzes Wort zu komplex. Zwei geschriebene Wörter mit der gleichen Bedeutung können sich in ihrem Schriftbild soweit unterscheiden, dass es schwer ist, durch Vergleich beider Wörter von der Semantik des ersten Wortes auf die Bedeutung des zweiten Wortes zu schließen. Daher wird in der Handschrifterkennung die Schrift weiter unterteilt und die Merkmale auf Teilabschnitten berechnet. Die Unterteilung, auch Segmentierung genannt, geht in der Regel auch eine Stufe tiefer als die Buchstabenebene, da diese ebenfalls noch zu komplex ist, sodass Buchstaben meist in mehrere Abschnitte unterteilt werden. Auf diesen segmentierten Trajektorien-Abschnitten lassen sich so Merkmale sinnvoll berechnen.

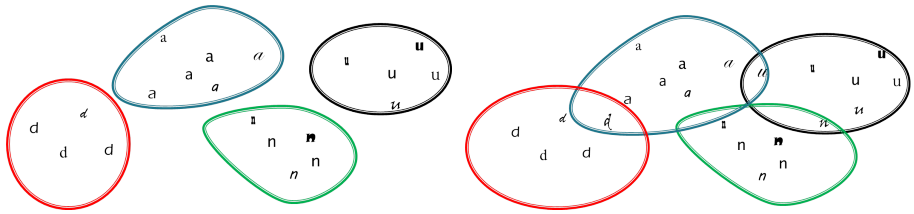
Weiterhin ist die Merkmalsberechnung für ein Segment nicht auf ein Merkmal beschränkt, sondern beschäftigt sich mit mehreren Merkmalen, die als ein Merkmalsvektor zusammengefasst werden. Zur Veranschaulichung kann ein großes „P“ und ein kleines „p“ betrachtet werden. Nimmt man die Form des Buchstaben als Merkmal, so unterscheiden sich beide Buchstaben in diesem Merkmal kaum voneinander und doch sind sie verschieden. Erst ein Merkmal, dass die Höhe bestimmt, könnte auf diese Weise den Unterschied herausstellen. Eine ebenfalls ähnliche Form haben die Buchstaben „q“ und „d“. Eine Unterteilung dieser in Halbkreis und Strich würde zur Klassifikation nicht ausreichen. Auch hier ist eine Positionierung der Teilbereiche eine zusätzliche Möglichkeit zur Unterscheidung beider Buchstaben.

5.2. Merkmalsrepräsentation

Bei der Merkmalsberechnung spielt eine gute Repräsentation der Merkmale eine große Rolle. Eine solche Repräsentation ermöglicht ein einfacheres Vergleichen und Unterscheiden der Merkmale. Drei Eigenschaften für eine gute und somit bedeutungsunterscheidende Repräsentation, wie in [Wie03] erläutert, werden im Folgenden näher betrachtet.

5.2.1. Diskriminativität

Beim Vergleich von zwei Merkmalsausprägungen unterschiedlicher Segmente kommt die Diskriminativität zum Tragen. Bei Zugehörigkeit zur selben Klasse, sprich bei starker Ähnlichkeit beider Segmente, fordert die Eigenschaft der Diskriminativität, dass die Merkmalsausprägungen sich nicht sonderlich unterscheiden sollen. Im umgekehrten Fall, wenn zwei Merkmalsausprägungen nicht zur selben Klasse gehören, sollen sie sich besonders stark voneinander unterscheiden. Wie in Abbildung 5.2 verdeut-



(a) (b)

Abbildung 5.2. – Veranschaulichung der Diskriminativität. Kreise stellen Klassen und ihre Grenzen dar. Buchstaben stehen für die Merkmalsausprägungen. Unterschiedliche Buchstaben in verschiedenen Schriftarten verdeutlichen die Gemeinsamkeiten bzw. die Unterschiede der Merkmalsausprägungen. (a) Alle Klassen sind strikt voneinander getrennt. (Diskriminativität) (b) Merkmale unterschiedlicher Klassen sind teilweise sehr ähnlich. (keine Diskriminativität)

licht, verlangt die Diskriminativität also, dass sich Merkmalsausprägungen der selben Klasse wenig, Merkmalsausprägungen unterschiedlicher Klassen hingegen stark unterscheiden, wodurch eine bessere Zuordnung zu den einzelnen Klassen im späteren Verlauf ermöglicht wird. [Wie03]

5.2.2. Robustheit

Ein gewähltes Merkmal soll möglichst robust sein. Bei einer zugehörigen Klasse sind geringe Schwankungen der Merkmalsausprägung erlaubt, bevor die Ausprägung nicht mehr zu dieser Klasse gehört [Wie03]. Abbildung 5.3 zeigt beispielhaft die Veränderungen einer Merkmalsausprägung anhand des Buchstabens „a“ in einer robusten Klasse.

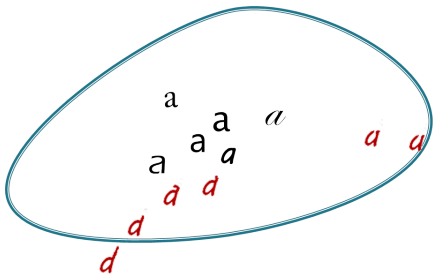


Abbildung 5.3. – Geringe Veränderung des Merkmals, hier am Buchstaben „a“ verdeutlicht, sind erlaubt, ohne dass die Klasse (blauer Kreis) verlassen wird.

5.2.3. Kompaktheit

Zusätzlich zur Diskriminativität und Robustheit eines Merkmals ist die Kompaktheit der Eingangsdaten wichtig [Wie03]. Da die Online-Handschrifterkennung mög-

lichst schnell erfolgen soll, ist die Verarbeitung auf kompakten Daten wünschenswert und fördert Geschwindigkeit und Effizienz des Erkenners. Zum einen soll die Merkmalsausprägung sich in einem definiertem Bereich befinden und zum anderen ist ein Merkmalsvektor mit einer niedrigeren Dimension wünschenswert.

5.3. Merkmalsextraktion

Die Merkmalsextraktion bei der Handschrifterkennung beschäftigt sich mit der Berechnung von Merkmalen aus Schriftbildern. Bei den Merkmalen kann zwischen zwei Arten unterschieden werden (vgl. [Wie03]), den sogenannten *High-Level*- und den *Low-Level*-Merkmalen. Die *High-Level*-Merkmale können nur berechnet werden, indem ein größerer Teil des Schriftbilds analysiert wird und spezifische Eigenschaften zu einem Merkmal vereint werden. Zu den *High-Level*-Merkmalen zählen unter anderem Schleifen (s. Abbildung 5.4(a)). Dabei werden Schleifen, sprich Teile eines Schriftzugs, die am gleichen Punkt anfangen und enden, ohne zwischenzeitliche Unterbrechungen (z.B. der obere Teil eines kursiv geschriebenen „h“), innerhalb des Schriftbilds gezählt und zu einem Merkmal aufaddiert. Ein weiteres *High-Level*-Merkmal kann berechnet werden, indem Kreuzungspunkte innerhalb eines Schriftbilds gezählt werden (s. Abbildung 5.4(b)). Ein *High-Level*-Merkmal bilden auch die Cusps, die Bereiche im Schriftbild markieren, die große Krümmungen aufweisen (s. Abbildung 5.4(c)). Zusätzlich kann bei der Online-Handschrifterkennung die Schreibgeschwindigkeit und insbesondere deren Minima oder Maxima ein *High-Level*-Merkmal darstellen. [Wie03]

(a) Schleifen (b) Kreuzungspunkte (c) Cusps, Bereiche der hohen Krümmung.

Abbildung 5.4. – Beispiele für *High-Level*-Merkmale (aus [Wie03]).

Im Gegensatz zu den *High-Level*-Merkmalen stehen die *Low-Level*-Merkmale. Diese werden lokal berechnet und benötigen somit nicht das gesamte Schriftbild, sondern lediglich einen Abschnitt. In der Offline-Handschrifterkennung werden diese Abschnitte bzw. Segmenten erzeugt, indem das Schriftbild in mehrere Fenster unterteilt wird (s. Abbildung 5.5). Diese Fenster sind in der Regel kleiner als ein einzelner Buchstabe und können sich überlappen. Bei der Online-Handschrifterkennung werden Segmente direkt über der Trajektorie errechnet. Diese bestehen dann aus einer bestimmten Anzahl von Trajektorienpunkten, die zeitlich hintereinander erzeugt wurden (vgl. Abschnitt 5.4).

Die Berechnung der Merkmale bei TabScript beschränkt sich, aufgrund des sequenziell gegebenem Schriftbilds, hauptsächlich auf die Berechnung von *Low-Level*-Merkmalen. Die berechneten Merkmale, wie *Schreibrichtung* oder *Krümmung* werden in

Abschnitt 5.5 näher betrachtet. Um zusätzliche Information aus der Umgebung eines Segments in die Auswertung der Schrift zu erhalten, werden mithilfe der errechneten *Low-Level*-Merkmale sogenannte *dynamische* Merkmale erzeugt, die durch diskrete Ableitungen berechnet werden können (vgl. [ST95]). Diese werden in Unterabschnitt 5.5.9 erläutert.

Abbildung 5.5. – Segmente einer offline Handschrifterkennung (aus [Wie03]).

5.4. Segmentierung

Wie im vorhergehenden Abschnitt erläutert, basieren die Merkmale, die in dieser Applikation berechnet werden (vgl. Abschnitt 5.5), auf Segmenten der Trajektorie. Um eine Zerlegung der Trajektorie vorzunehmen, auch *Segmentierung* genannt, gibt es verschiedene Möglichkeiten. Eine Variante ist beispielsweise die Segmentierung anhand lokaler Minima und Maxima. Die in der Applikation implementierte Segmentierung teilt die Trajektorie an Punkten maximaler lokaler Krümmung auf (vgl. Abbildung 5.6). Dabei wird auf eine Mindestlänge der Trajektoriensegmente geachtet, damit die extrahierten Segmente für die Berechnung von Merkmalen geeignet sind. Um möglichst alle Informationen der Trajektorie zu erfassen, sowie den Übergang von Segment zu Segment zu glätten, werden zusätzliche Segmente erstellt, die jeweils 50% zweier aufeinanderfolgender Segmente vereinen (vgl. [Wie03]). So gehen auch die lokalen Krümmungen, nach denen zuvor geteilt wurde, in die Merkmalsberechnung ein.

5.5. Merkmale

Die in dieser Applikation berechneten Merkmale sind mithilfe heuristischer Verfahren erzeugt worden. Solche Verfahren der Merkmalsberechnung basieren auf Plausibilitäts-Überlegungen und berechnen Merkmale, die sinnvoll erscheinen [Wie03]. So können sie zur Lösung der Handschrifterkennung beitragen, müssen es aber nicht. Ein heuristisches Merkmal in diesem Sinne ist beispielsweise die Schreibrichtung. Es ist eine plausible Annahme, dass die Schreibrichtung viel Information zu einem Zeichen liefern kann. Ob dies der Fall ist, kann jedoch erst durch die Evaluation der Handschrifterkennung empirisch belegt werden. Neben den heuristischen Verfahren werden analytische Verfahren eingesetzt, die geeignete Merkmale aus dem Merkmalsraum filtern sollen, um eine bessere Repräsentation zu erzielen. Ein gängiges Verfahren ist dabei die *Hauptkomponentenanalyse*, auch kurz PCA genannt (engl: *Principal Component Analysis*,

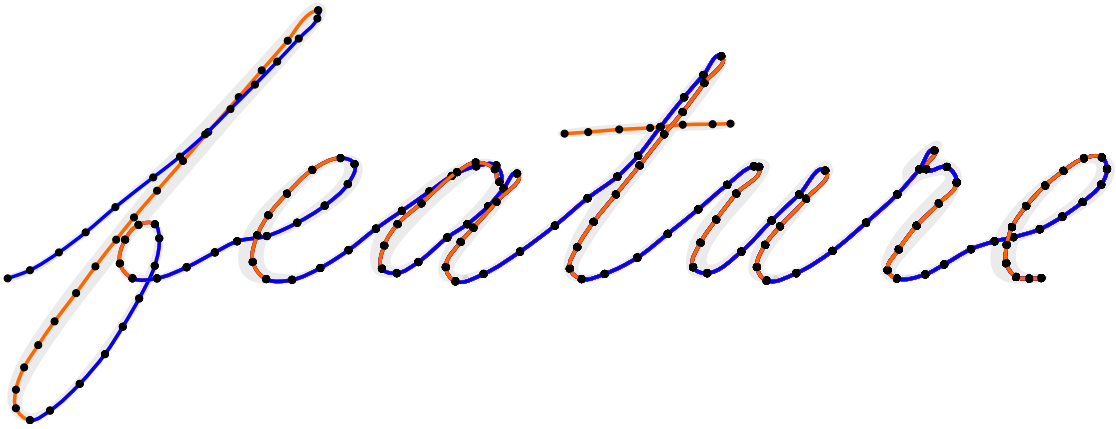


Abbildung 5.6. – Veranschaulichung der Segmentierung einer Trajektorie. Segmente (farbig hervorgehoben) werden durch die maximalen lokalen Krümmungen in der gesamten Trajektorie begrenzt.

vgl. [Nie83]), deren Ziel die Dekorrelation von Merkmalen sowie die Dimensionsreduktion ist (vgl. [Wie03]). Für weitere Informationen zum Thema *analytische Verfahren* sei hier auf [Nie83, Kapitel 3.1] verwiesen.

Im Folgenden sind einige gängige heuristische Merkmale aufgelistet, die in der Applikation umgesetzt wurden. Dabei bezeichnen $x(t)$ und $y(t)$ jeweils die Position eines Trajektoriepunktes p_t auf der x- bzw. y-Achse zum Zeitpunkt t innerhalb eines Segments S mit $S = \{t_1, \ldots, t_n\}$. X bildet die Menge der x-Werte, Y die Menge der y-Werte aller Punkte eines Segments. Falls eine Normierung eines Merkmals über dem Segment vorgenommen wird, geschieht dies durch

$$M_i(S) = \frac{1}{|S|} \sum_{t \in S} M_i(t), \tag{5.1}$$

wobei $M_i(S)$ das Merkmal für ein ganzes Segment und $M_i(t)$ das Merkmal zum Zeitpunkt t bezeichnet. Die Wahl der Merkmale wird im anschließenden Abschnitt erläutert.

5.5.1. Schreibrichtung

Die *Schreibrichtung* ist ein Merkmal, das mithilfe der zeitlich umliegenden Punkte errechnet wird und die Schreibrichtung in Form eines Winkels repräsentiert. Dabei werden zwei Merkmale durch den *Sinus* und *Kosinus* des Winkels erzeugt. Die *Schreibrichtung* kann, wie in Abbildung 5.7 visualisiert, durch Berechnungen der Gleichungen 5.2, 5.3, 5.4, 5.5 und 5.6 bestimmt werden [JMW00]. Der *Kosinus* und *Sinus* für die Richtung kann aus den Gleichungen 5.2 und 5.3 berechnet werden. Dabei werden der x- (Gleichung 5.5) bzw. y-Abstand (Gleichung 5.6) der umliegenden Punkte p_{t-1} und p_{t+1}, sowie die euklidische Länge des Vektors $\Delta s(t)$ zum Punkt p_t (Gleichung 5.4) zu-

hilfe genommen. Für die Berechnung der Schreibrichtung über einem Segment werden die Schreibrichtungen über die Anzahl der Punkte im Segment normiert.

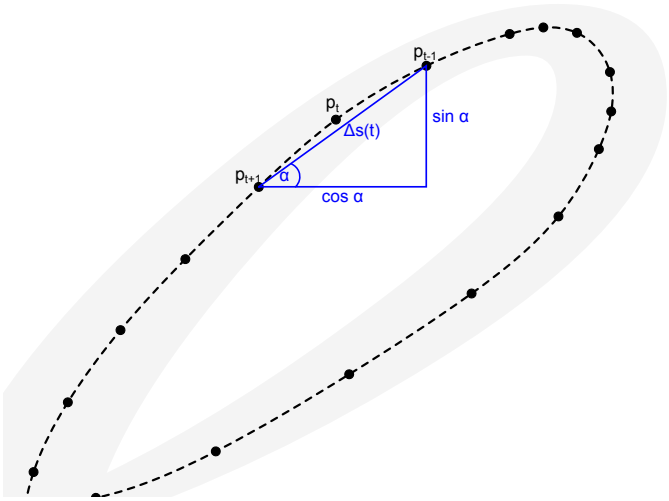


Abbildung 5.7. – Visualisierung des Schreibrichtungsmerkmals an einem Trajektorienabschnitt (vgl. [JMW00])

$$\cos\alpha(t) = \frac{\Delta x(t)}{\Delta s(t)}. \tag{5.2}$$

$$\sin\alpha(t) = \frac{\Delta y(t)}{\Delta s(t)}. \tag{5.3}$$

$$\Delta s(t) = \sqrt{\Delta x^2(t) + \Delta y^2(t)}. \tag{5.4}$$

$$\Delta x(t) = x(t-1) - x(t+1). \tag{5.5}$$

$$\Delta y(t) = y(t-1) - y(t+1). \tag{5.6}$$

5.5.2. Krümmung

Wie auch die Schreibrichtung (Unterabschnitt 5.5.1), wird die Krümmung mithilfe der zeitlich umliegenden Punkten errechnet und durch einen Winkel und dessen *Sinus* bzw. *Kosinus* repräsentiert, der angibt, wie weit die Trajektorie lokal gekrümmt ist. Die Idee des Informationsgehalts der Krümmung kann durch die unterschiedlichen Formen der Buchstaben plausibel erklärt werden. Ein Buchstabe wie *o* beispielsweise hat einen global ähnlichen Krümmungsverlauf, während der Buchstabe *z* zwei besonders stark gekrümmte Stellen hat. Unter Zuhilfenahme der Gleichungen 5.2 und 5.3 kann die Krümmung durch die Gleichungen 5.7 und 5.8 berechnet werden. Abbildung 5.8

beschreibt dieses Prinzip. Die Normierung über die Segmentpunkte ergibt wiederum das zugehörige Krümmungsmerkmal für ein Segment.

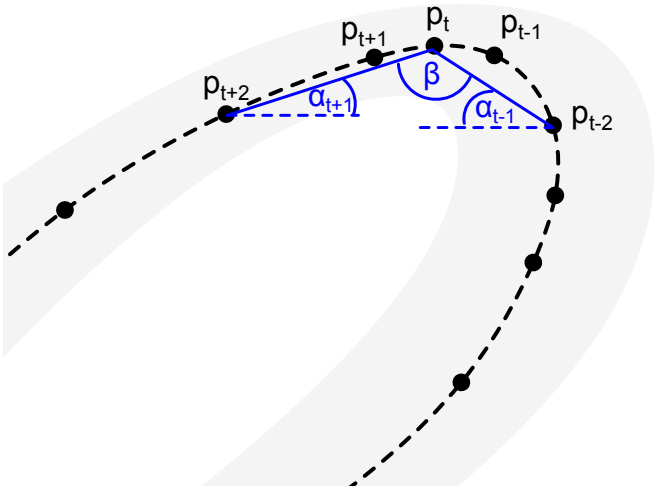


Abbildung 5.8. – Visualisierung der Berechnung eines Krümmungsmerkmals an einem Trajektorienabschnitt (vgl. [JMW00]).

$$\cos\beta(t) = \cos\alpha(t-1)\cos\alpha(t+1) + \sin\alpha(t-1)\sin\alpha(t+1). \tag{5.7}$$

$$\sin\beta(t) = \cos\alpha(t-1)\sin\alpha(t+1) - \sin\alpha(t-1)\cos\alpha(t+1). \tag{5.8}$$

5.5.3. Stiftdruck

Das Stiftdruck-Merkmal gibt an, ob sich der Stift zu einem Zeitpunkt t auf dem Tablet bzw. auf der Schreibunterlage befunden hat (*pen down*), oder nicht (*pen up*) und gibt somit einen möglicherweise wichtigen Hinweis auf die Bedeutung des Segmentabschnitts für die Erkennung. Für den Fall, dass Trajektorienpunkte zwischen zwei *Pen-down*-Phasen, also während der *Pen-up*-Phase interpoliert werden (vgl. Abbildung 5.9), lässt sich dieses Merkmal nicht mehr aus dem Schriftbild herauslesen (vgl. Interpolation in Kapitel 4) und muss daher zuvor gespeichert werden. In genau diesem Fall könnte es wichtig sein, zu wissen, ob der Stift auf dem Tablet war, oder nicht [JMW00]. Jeder Trajektorienpunkt erhält je nach Stiftstatus, einen negativen (*Pen-Down*-Phase) oder positiven (*Pen-Up*-Phase) Wert. Durch Summierung und Normierung dieser Werte erzeugt man so ein gemitteltes Stiftdruck-Merkmal für das Segment.

5.5.4. Aspect Ratio

Das *Aspect-Ratio*-Merkmal (s. Abbildung 5.10) errechnet sich durch Gleichung 5.11 aus dem Verhältnis eines umgebenden Rahmens um ein Segment von Punkten, wobei

Abbildung 5.9. – Visualisierung des Stiftdruck-Merkmals. Unterbrechungen in der Trajektorie, beispielsweise durch Absetzen des Stiftes, werden interpoliert (hier blau dargestellt) und markiert. Der Stiftdruck ergibt sich durch Einbeziehen der Stiftposition in der dritten Dimension.

sich Δx und Δy mithilfe der Punkte im Segment aus Gleichung 5.9 und Gleichung 5.10 ergeben [JMW00]. Von größerer Bedeutung könnte das Merkmal bei länglichen Segmenten, z.B. Ausschnitte des Buchstaben *t* im Vergleich zu eher kompakten Daten wie z.B. der Buchstabe *o*.

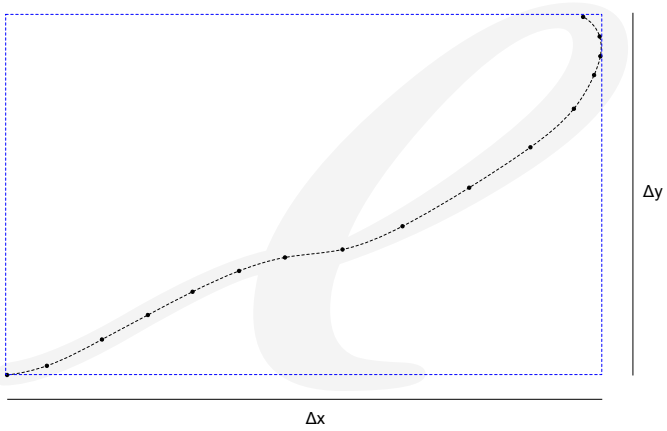


Abbildung 5.10. – Visualisierung eines *Aspekt-Ratio*-Merkmals an einem Trajektorienabschnitt (vgl. [JMW00]).

$$\Delta x = \max\{X\} - \min\{X\} \tag{5.9}$$

$$\Delta y = \max\{Y\} - \min\{Y\} \tag{5.10}$$

$$A = \frac{2\Delta y}{\Delta x + \Delta y} - 1 \tag{5.11}$$

5.5.5. Ablenkung

Das Ablenkungsmerkmal (s. Abbildung 5.11) wird durch den Vergleich der euklidischen Länge L des Trajektoriepfades eines Segments (vgl. Gleichung 5.12) mit der größten Länge des umgebenden Rahmens berechnet, wie in Gleichung 5.13 definiert und gibt an, inwieweit das Segment sich zu einer der beiden Seiten gekrümmt wird. Δx und Δy ergeben sich dabei durch Gleichungen 5.9 und 5.10.

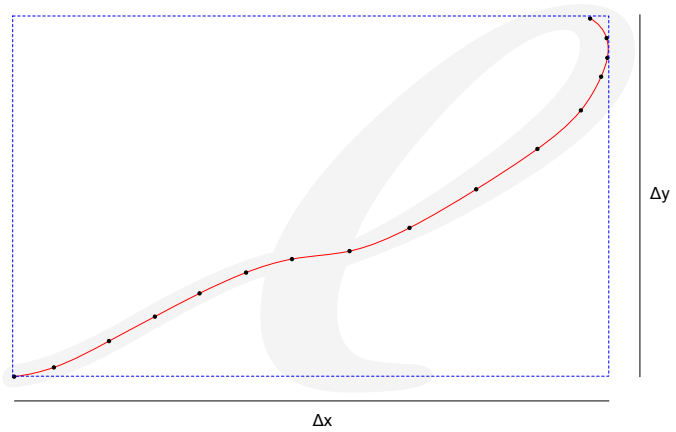


Abbildung 5.11. – Visualisierung der Berechnung eines Ablenkungsmerkmals an einem Trajektorienabschnitt (vgl. [JMW00]).

$$L = \sum_{i=2}^{n} \sqrt{(x(t_i) - x(t_i - 1))^2 + (y(t_i) - y(t_i - 1))^2} \tag{5.12}$$

$$C(t) = \frac{L}{\max\{\Delta x, \Delta y\}} - 2 \tag{5.13}$$

5.5.6. Linearität

Das Linearitätsmerkmal dient der approximativen Ermittlung der Geradlinigkeit eines Schriftabschnittes (s. Abbildung 5.12). Dazu werden die aufsummierten und gemittelten Abstände der Trajektorienpunkte zu einer Mittelgeraden errechnet (s. Gleichung 5.14). N gilt dabei als Anzahl der Punkte des Trajektorienabschnitts und d_i als kürzeste euklidische Distanz eines Punktes zur Geraden zwischen dem ersten und letzten Punkt in dem Segment. [JMW00]

$$LN(t) = \frac{1}{N} \sum_{i} d_i^2 \tag{5.14}$$

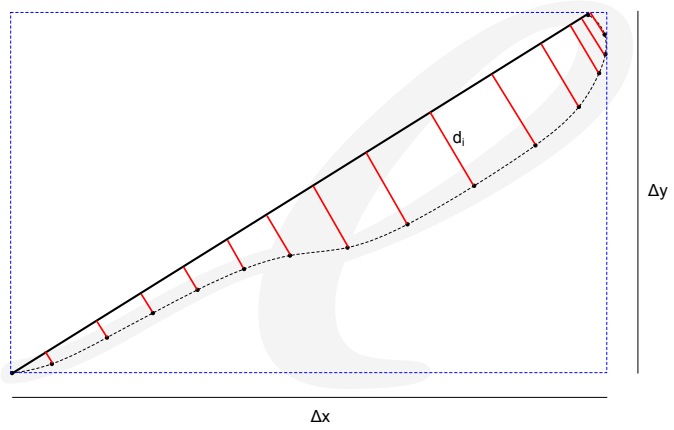


Abbildung 5.12. – Visualisierung der Berechnung eines Linearitätsmerkmals an einem Trajektorienabschnitt (vgl. [JMW00]).

5.5.7. Anstieg

Das Merkmal des Anstiegs ergibt sich aus dem Winkel zwischen der horizontalen Linie und der Geraden, die den ersten und den letzten Punkt eines Segments miteinander verbindet (s. Abbildung 5.13) und kann als eine weitere Möglichkeit der Schreibrichtungsermittlung angesehen werden [JMW00].

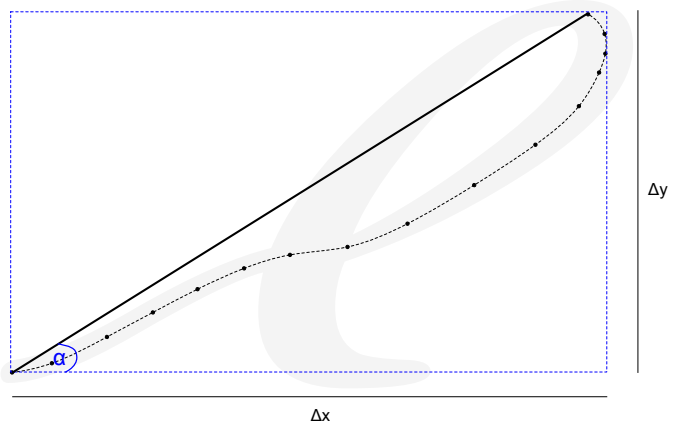


Abbildung 5.13. – Veranschaulichung eines Anstiegmerkmals (vgl. [JMW00]).

5.5.8. Ascenders und Descenders

Die Merkmale *Ascenders* und *Descenders* werden berechnet, indem die Anzahl der Punkte aufsummiert wird, die sich oberhalb der Corpusline, bzw. unterhalb der Baseline, innerhalb eines Segments befinden. Punkte, die sich innerhalb eines Toleranzbereichs b über bzw. unter den Linien befinden, werden nicht einbezogen. Damit werden

Buchstaben erfasst, die über den Mittelbereich nach oben oder unten hinausgehen, wie „t“ oder „g“. Kleine Abweichungen im Schriftbild werden durch den Schwellwert b nicht erfasst [JMW00].

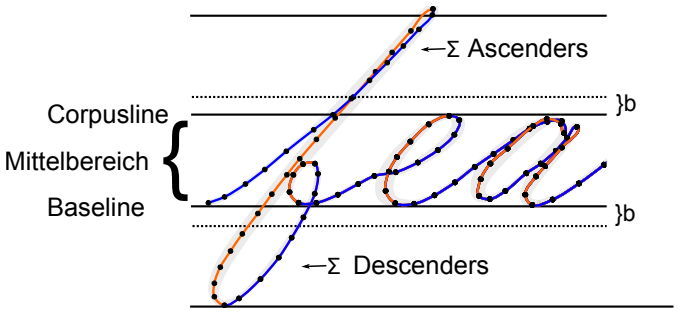


Abbildung 5.14. – Veranschaulichung der *Ascender*- und *Descender*-Merkmale (vgl. [JMW00]).

5.5.9. Dynamische Merkmale

Neben den bereits vorgestellten Merkmalen bieten sich dynamische Merkmale an, um weitere Information aus den berechneten Größen zu gewinnen. Die bisherigen Merkmale wurden jeweils für ein Segment berechnet und sind daher im Bereich der *Low-Level*-Merkmale anzusiedeln. Die Idee dynamischer Merkmale ist, den Kontext eines Segments mit in den Merkmalsvektor einzubeziehen. Dabei basiert ein dynamisches Merkmal auf der Berechnung eines anderen zuvor definierten und berechneten Merkmals, sowohl auf Segment S_t als auch S_{t-1} und S_{t+1}, wobei t der Index der nacheinanderfolgenden Segmente einer Trajektorie ist (vgl. [ST95]). Die berechneten dynamischen Merkmale stellen somit die diskreten Ableitungen der jeweiligen Merkmale dar. Die detaillierte Berechnung ist in Gleichung 5.15 aufgeführt und in Abbildung 5.15 illustriert. $D_i(S)$ bezeichnet hierbei das dynamische Merkmal zu Merkmal i und Segment S; $M_i(S)$ liefert die Ausprägung des Merkmals i von Segment S. Zusätzlich bezeichnet S_0 das erste und S_n das letzte Segment der Trajektorie.

In Abbildung 5.15 sind in der obersten Zeile die Segmente S aufgelistet. Pro Segment werden Merkmalsvektoren mit den Merkmalen M_0 bis M_m berechnet, die in der Abbildung mittig beschrieben werden. Die diskreten Ableitungen (in der Abbildung unten) werden durch die, zu den jeweiligen Merkmalen äquivalenten, dynamischen Merkmalen D_0 bis D_m repräsentiert. Sie errechnen sich als jeweiliges Mittel der zeitlich vorherigen und nachfolgenden Merkmale (durch Pfeile illustriert).

$$D_i(S_t) = \begin{cases} D_i(S_{t+1}), & \text{falls } S_t = S_0 \\ D_i(S_{t-1}), & \text{falls } S_t = S_n \\ \dfrac{M_i(S_{t+1}) - M_i(S_{t-1})}{2}, & \text{sonst} \end{cases} \tag{5.15}$$

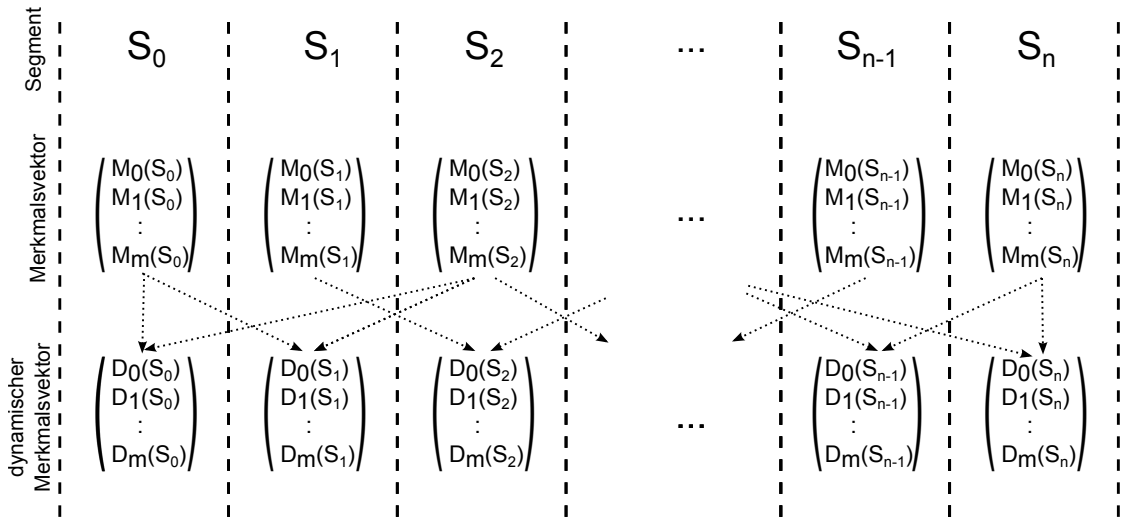


Abbildung 5.15. – Visualisierung der Berechnung dynamischer Merkmale. Dynamische Merkmale werden durch Kombination von bereits berechneten Merkmalen der umliegenden Segmente berechnet.

5.6. Verwendete Merkmale

Die berechneten Merkmale einer Trajektorie werden für den weiteren Verlauf der Handschrifterkennung an den Klassifizierer weitergeleitet, in dem das zu erkennende Wort statistisch berechnet werden soll (vgl. Kapitel 6). Die zuvor erläuterten Merkmale wurden aufgrund ihrer intuitiven Herleitung durch Plausibilitäts-Überlegungen sowie ihrer praktischen Relevanz (vgl. [Wie03] und [JMW00]) in der Applikation implementiert. Anschließend wurden mithilfe einer Evaluation an einem Test- und Trainingsdatensatz unterschiedliche Kombinationsmöglichkeiten von implementierten Merkmalen ausgewertet. Aufgrund der in Abschnitt 7.3 näher betrachteten Auswertung, wurden die folgenden Merkmale für die Handschrifterkennung verwendet:

- Schreibrichtung,
- Krümmung,
- Aspect Ratio,
- Dynamische Merkmale.

6. Erkennung

Zum Zeitpunkt, an dem die Erkennung in TabScript gestartet wird, wurde eine aufgenommene Trajektorie vorverarbeitet, segmentiert und für jedes der enstandenen Segmente wurde ein Merkmalsvektor berechnet (vgl. Kapitel 4 und Kapitel 5). Ein Wort liegt somit als Sequenz von Merkmalsvektoren vor. Für die Erkennung wird nun die Annahme getroffen, dass ein statistisches Modell, bestehend aus mehreren Zuständen, existiert, welches diese Sequenz generiert hat. Wird die Sequenz als Ausgabe des Modells interpretiert, kann im Folgenden die Wahrscheinlichkeit bestimmt werden, mit der dieses Modell die Sequenz erzeugt hat und welche der Zustände dabei durchlaufen wurden. Dieser Schritt wird als *Erkennung* bezeichnet.

In der TabScript-Applikation werden *Hidden-Markov-Modelle* verwendet um Handschrift zu modellieren. Der Aufbau und die Funktionsweise von Hidden-Markov-Modellen werden in Abschnitt 6.1 erläutert. Die zuvor beschrieben Erkennung (auch *Dekodierung* genannt) ist in Unterabschnitt 6.1.1 beschrieben. Bevor das Modell allerdings eingesetzt werden kann, muss es durch bereits vorhandene Daten trainiert werden, indem die Parameter des Modells statistisch geschätzt werden. Dieses Training wird in Unterabschnitt 6.1.2 erklärt. Für den Einsatz von Hidden-Markov-Modellen wurde der Projektgruppe das *ESMERALDA*-Toolkit zur Verfügung gestellt, welches in Abschnitt 6.2 vorgestellt wird. Hier wird zudem auf die Parameterkonfiguration für den Einsatz von Esmeralda in TabScript eingegangen.

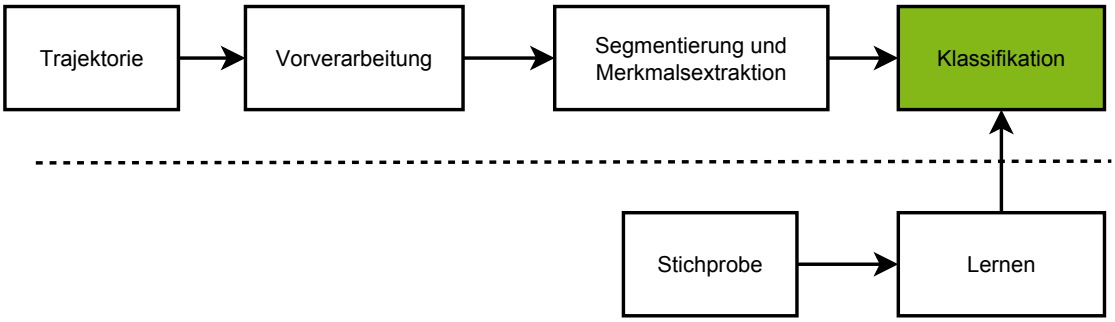


Abbildung 6.1. – Einordnung des Erkennungsschrittes in die Verarbeitungspipeline (vgl. [Nie83])

Wie Abbildung 6.1 zeigt, ist die Erkennung bzw. Klassifikation der letzte und entscheidende Schritt der Verarbeitungspipeline. Sie dient als Schnittstelle zwischen statistischem Modell und der Benutzeroberfläche, auf der das Erkennungsergebnis angezeigt wird. Das Modell wird dabei mit Hilfe einer Stichprobe trainiert, auf deren Daten

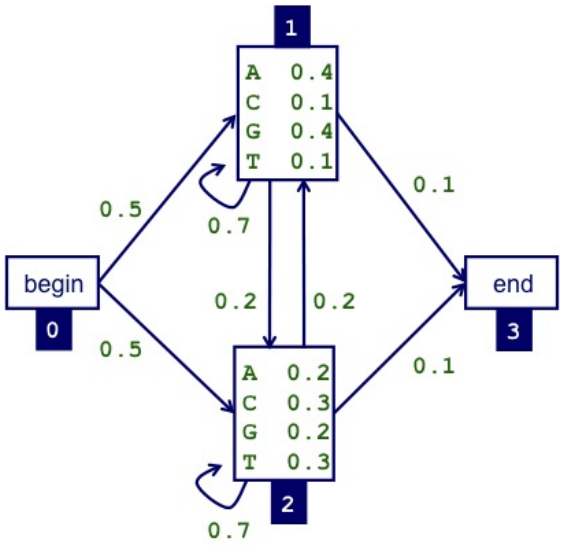


Abbildung 6.2. – Beispiel eines diskreten Hidden-Markov-Modells für den Fall der DNA-Modellierung[1]. Jeder Zustand hat eine Wahrscheinlichkeitsverteilung über alle Emissionen (A, C, G, T).

die gleichen Folge von Operationen (Vorverarbeitung, Segmentierung und Merkmalsextraktion) angewendet wird.

6.1. Hidden-Markov-Modelle

Hidden-Markov-Modelle (HMM) werden zur Modellierung von sequenziellen Daten verwendet (vgl. [Fin08]). Ihren Ursprung haben sie in der Modellierung von Sprachsignalen (zum Beispiel Sequenzen von Phonemen). Ein weiteres Einsatzgebiet ist die Darstellung von Gesetzmäßigkeiten beim Bau von DNA [Rah09].

Ein Hidden-Markov-Modell ist im Wesentlichen ein endlicher Automat mit Ausgaben (im Folgenden auch: Emissionen), der sich zu jedem Zeitpunkt t in einem Zustand q befindet. Aus der endlichen Menge Q an Zuständen ist eine Teilmenge $Q_0 \subseteq Q$ zudem als Menge von Startzuständen gekennzeichnet. Aus dieser Menge wird zum Zeitpunkt $t = 0$ anhand einer modellabhängigen Verteilung der Startzustand bestimmt. In jedem Zustand q wird gemäß einer zustandsabhängigen Übergangsverteilung $a_{q,q'}$ ein zufälliger Übergang zum Nachfolgezustand q' durchgeführt. Dieser Übergang (und insb. der Folgezustand) ist nur vom aktuellen Zustand q abhängig. Es folgt anhand einer ebenfalls zustandsabhängigen Emissionsdichte (bei diskreten HMMs: Wahrscheinlichkeitsverteilung) die Generierung einer Ausgabe im jetzt aktuellen Zustand q'. Ein HMM wird als *diskret* bezeichnet, wenn es Emissionen generiert, die aus einem endlichen Ausgabealphabet stammen, wie beispielsweise im Fall von DNA (gezeigt in

[1] Quelle: `http://www.biostat.wisc.edu/bmi576/hmm.jpg`, zuletzt abgerufen am 27.11.2013

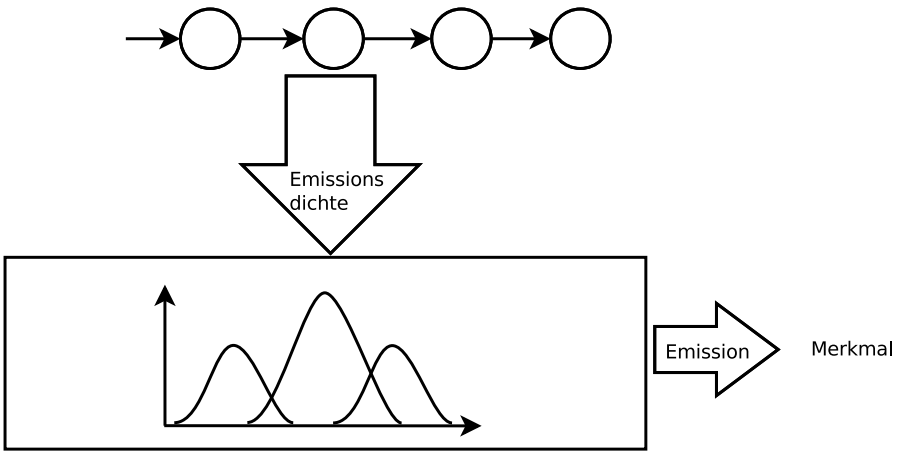


Abbildung 6.3. – Schematische Darstellung eines kontinuierlichen Hidden-Markov-Modells mit vier Zuständen für ein Merkmal. Jeder Zustand generiert Emissionen in Abhängigkeit seiner Emissionsdichten (hier: Mischverteilung für das Merkmal).

Abbildung 6.2). In diesem Beispiel besteht die Emissionsmenge aus den vier DNA-Bausteinen (A, C, G, T). Jeder Zustand hat eine vollständige Wahrscheinlichkeitsverteilung über alle Emissionen. *Kontinuierliche* HMMs generieren Ausgaben in Abhängigkeit von Wahrscheinlichkeitsdichten (bspw. Merkmalsvektoren), wie im Fall der umgesetzten Handschrifterkennung. Ein Beispiel für ein HMM mit nur einem Merkmal, dessen Verteilungsdichte eine Mischverteilung aus drei Normalverteilungen ist, ist in Abbildung 6.3 dargestellt. Für den allgemeinen Fall von mehreren Merkmalen werden die Dichten durch Mittelwertvektoren und Kovarianzmatrizen beschrieben, um Abhängigkeiten zwischen den Merkmalen zu erfassen.

Die Generierung von Daten durch das HMM erfolgt anhand von zwei stochastischen Prozessen: der Folge von Zuständen, die von den Übergangswahrscheinlichkeiten der einzelnen Zustände abhängt, und der Folge von Emissionen, die von den Emissionsdichten jedes Zustands abhängt. Besonders zu beachten ist hierbei jedoch, das lediglich die Emissionsfolge beobachtbar ist, die Zustandsfolge hingegen nicht (versteckt, engl. *hidden*). Aus dieser Eigenschaft folgt, dass es zu einer Emissionsfolge keine eindeutige Folge von Zuständen gibt, die diese erzeugt haben, sondern mehrere Zustandsfolgen hierfür infrage kommen. Wie trotz dieser Schwierigkeit die „richtige“ Zustandsfolge bestimmt wird, erläutert Unterabschnitt 6.1.1. Das Aufdecken der versteckten Zustandsfolge ist der Erkennungsprozess und wird auch als *Dekodierung* bezeichnet.

6.1.1. Dekodierung

Die Dekodierung eines Hidden-Markov-Modells bezeichnet die Bestimmung der wahrscheinlichsten Zustandsfolge für eine gegebene Folge von Beobachtungen (=Merkmalsvektoren). Wie in Abschnitt 5.4 beschrieben, wird die durch das Tablet

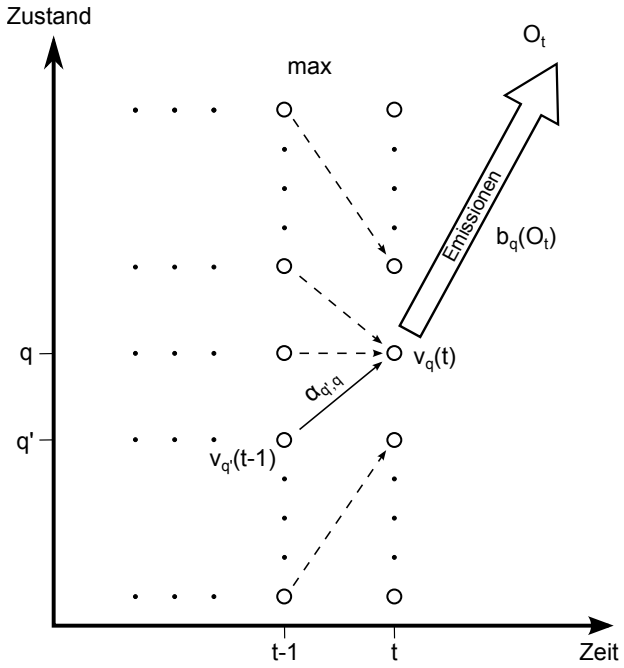


Abbildung 6.4. – Schematische Darstellung der Bestimmung einer Viterbi-Variablen (vgl. [Fin08])

erfasste Trajektorie in Segmente aufgeteilt und für jedes dieser Segmente ein Merkmalsvektor berechnet. Es wird nun die wichtige Annahme getroffen, dass diese Folge von Merkmalsvektoren durch ein vorliegendes Hidden-Markov-Modell λ als Folge von Emissionen $\mathbf{O}$ erzeugt wurde. Wenn von dieser Annahme ausgegangen wird, dann identifiziert die Zustandsfolge

$$\mathbf{s}^* = \underset{\mathbf{s}}{\operatorname{argmax}}\ P(\mathbf{s}|\mathbf{O}, \lambda), \tag{6.1}$$

das Wort, das am wahrscheinlichsten geschrieben wurde. $P(\mathbf{s}|\mathbf{O}, \lambda)$ ist die bedingte Wahrscheinlichkeit dafür, dass, gegeben das Modell λ, die Zustandsfolge $\mathbf{s}$ die beobachtete Emissionsfolge $\mathbf{O}$ generiert hat. Naiv kann diese, die Produktionswahrscheinlichkeit maximierende, Zustandsfolge bestimmt werden, indem die Wahrscheinlichkeit aller möglichen Zustandsfolgen des vorliegenden Modells berechnet werden. Dies ist jedoch besonders bei größeren Modellen und auf leistungsschwächerer Hardware nicht praktikabel.

Eine effiziente Alternative zur Berechnung stellt der sogenannte Viterbi-Algorithmus dar (vgl. [Fin08]). Die die Berechnung maximierenden Zustandsfolge erfolgt hierbei durch die Definition von Hilfsvariablen, den Viterbi-Variablen [Rah13]

$$v_q(t) = \max_{S_1, S_2, \ldots S_{t-1}} P(O_0, \ldots, O_t, S_1, \ldots, S_{t-1}, S_t = q|\lambda), \tag{6.2}$$

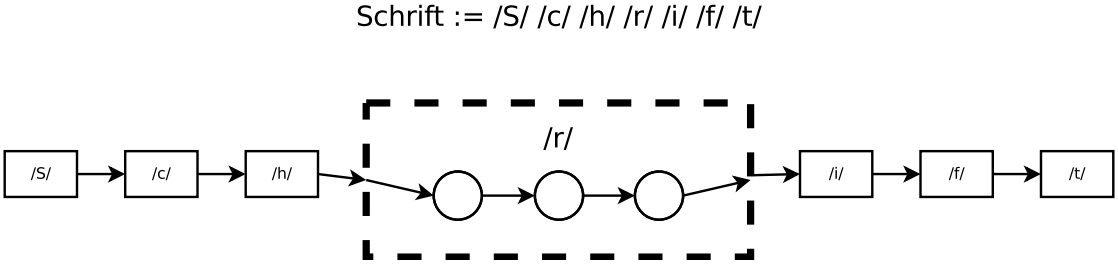


Abbildung 6.5. – Definition des Wortes „Schrift“ als Kombination von Buchstabenmodellen und das daraus entstehende virtuelle Wortmodell, welches aus mehreren kleinen Hidden-Markov-Modellen besteht (hier am Beispiel von „r“ hervorgehoben).

welche die Wahrscheinlichkeiten aller Pfade maximieren, die zum Zeitpunkt t in Zustand i enden [Sch04]. Dieser Berechnungsschritt ist schematisch in Abbildung 6.4 dargestellt. Die Hilfsvariable $v_q(t)$ wird iterativ durch

$$v_q(0) = a_{q_0,q} \cdot b_q(O_1), \qquad \text{für alle } q \in Q \tag{6.3}$$

$$v_q(t) = \max_{q' \in Q}(v_{q'}(t-1) \cdot a_{q',q}) \cdot b_q(O_t), \qquad q \in Q, t = 1, \ldots, T \tag{6.4}$$

berechnet, wobei q_0 der Startzustand ist und $b_q(O_1)$ die Wahrscheinlichkeit ist, in Zustand q die Emission O_1 zu erzeugen [Rah13]. Zur Beschreibung des Algorithmus wird hier ein Modell mit diskreten Wahrscheinlichkeitsverteilungen der Ausgaben $b_q(\cdot)$ angenommen. Für den Zeitpunkt T wird abschliessend der letzte Zustand

$$S_T = \max_{q \in Q} v_q(T) \tag{6.5}$$

von s^* bestimmt. Die gesuchte Zustandsfolge $s^* = S_1 \ldots S_T$ lässt sich ausgehend von S_T durch Rückverfolgung (engl. *Backtracing*) ermitteln, indem zu jedem Zeitpunkt der die Viterbi-Variablen maximierende Zustand bestimmt wird [Rah13].

In Abschnitt 6.1 wurde bereits angedeutet, dass das HMM in TabScript lediglich aus Buchstabenmodellen besteht. Aus diesen Buchstabenmodellen müssen während der Dekodierung dynamisch Modelle für Wörter zusammengesetzt werden. Dies passiert anhand von *Konzepten*. Konzepte sind Definitionen und Regeln für das dynamische Erstellen von komplexeren virtuellen Modellen aus einfachen Modellen. In Abbildung 6.5 ist die Verwendung von Konzepten in der TabScript-Applikation verdeutlicht. Das zu erkennende Wort „Schrift“ wird als Folge von Buchstabenmodellen definiert. Anhand dieser Definition wird bei der Dekodierung dynamisch das dargestellte „Schrift“-Modell erstellt, für das anschliessend mit dem Viterbi-Algorithmus die Produktionswahrscheinlichkeit für eine gegebene Folge von Merkmalsvektoren berechnet werden kann. Zuvor werden alle Wortmodelle zu einem sogenannten Lexikonbaum zusammengefügt, welcher ein großes HMM ist, auf dem der Viterbi-Algorithmus arbeitet. Der schematische Aufbau dieses Baums ist in Abbildung 6.6 dargestellt. Jede

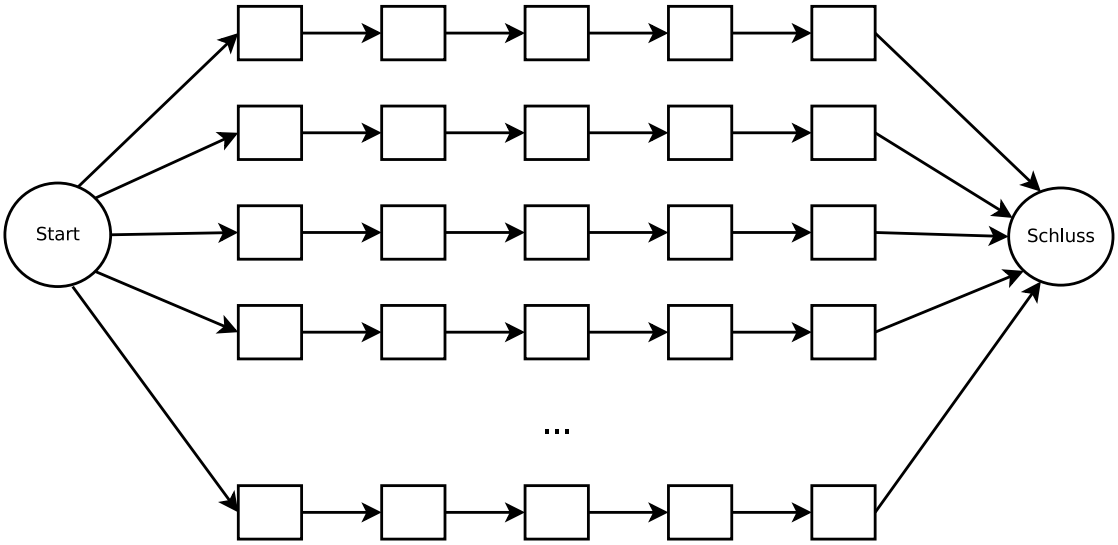


Abbildung 6.6. – Aufbau eines Lexikonbaums bei der Dekodierung. Jede Zeile entspricht einem Wortmodell (siehe Abbildung 6.5). Die einzelnen Wortmodelle werden durch die Pseudozustände „Start“ und „Schluss“ zu einem größeren HMM zusammengefügt.

Zeile besteht aus einem Wortmodell, das wiederum (wie zuvor erläutert) aus mehreren Buchstabenmodellen besteht. Die Wortmodelle werden durch zwei Pseudoknoten am Anfang und Ende verbunden.

6.1.2. Training

Im Folgenden wird das Training des HMM skizziert, welches für das Hidden-Markov-Modell in TabScript zum Einsatz kommt. Eine allgemeinere Betrachtung der Parameterschätzung von HMM ist unter anderem in [Fin08] zu finden. Das Training wird mit den gewählten Daten der Unipen-Datenbank (siehe Unterabschnitt 3.3.2) durchgeführt. Diese Daten liegen jeweils als Folge von Merkmalsvektoren vor, die, wie in Kapitel 5 erläutert, berechnet wurden. Der gesamte Datensatz wird vor Beginn des Trainings aufgeteilt in ein Trainingsset, das ungefähr 90% der Daten beinhaltet, und ein Testset (restliche 10%), welches zur späteren Evaluierung des erstellten Modells dient. Die Parameter des Modells werden aussschliesslich mit den Trainingsdaten geschätzt.

Als erstes wird auf den gegebenen Daten der *k-means*-Algorithmus angewendet (vgl. [Fin08]). Dieser schätzt Mischverteilungen im Merkmalsraum, jeweils durch Mittelwert und Kovarianz-Matrix definiert, um Häufungsgebiete und die allgemeinen Ausprägungen der Merkmale zu erfassen. Die Mischverteilungen werden zur weiteren Verwendung im sogenannten *Codebuch* gespeichert. Die Bestimmung einer Verteilung ist schematisch in Abbildung 6.7 dargestellt: für die rot dargestellten Ausprägungen zweier generischer Merkmale wurde eine Mischverteilung (bestehend aus zwei Normalverteilungen) geschätzt, die durch die schwarzen Ellipsen angedeutet wird. Aufgrund die-

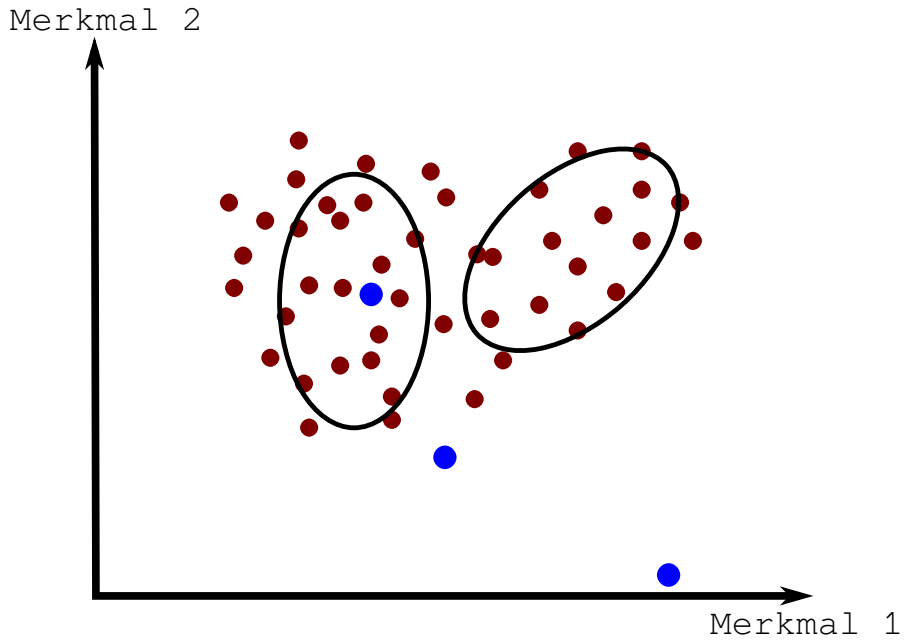


Abbildung 6.7. – Beispiel für durch *k-means* geschätzte Normalverteilungen (schwarz) für eine Menge bekannter Ausprägungen von zwei Merkmalen (rot). Für jeden unbekannten Punkt (blau) kann nun die Wahrscheinlichkeit bestimmt werden, mit der dieser Punkt von jeder der Verteilungen erzeugt wurde.

ser Verteilung kann im weiteren Verlauf des Trainings und der Dekodierung die Wahrscheinlichkeit bestimmt werden, mit dem unbekannte Punkte (hier in blau) zu dieser Verteilung gehören.

Jede Mischverteilung ist eine Klasse, anhand derer die Zustände des HMM definiert werden. Jeder Zustand ist über Mischverteilungsgewichte definiert, die die Wahrscheinlichkeit angeben, mit der der Zustand Ausgaben mit der jeweiligen Klasse generiert. Zusammengefasst ist die Struktur des HMM also gespeichert als

1. Codebuch, dass die Klassen (Mischverteilungen) enthält,
2. Liste der Zustände, definiert anhand des Codebuchs durch Mischverteilungsgewichte und
3. Definition der Buchstabenmodelle durch Angabe der Zustände in jedem Modell.

Wie in Unterabschnitt 3.3.2 beschrieben wird, liegen die zum Training verwendeten Trajektorien mit Annotationen auf Wort-Basis vor. Dennoch besteht das verwendete Modell aus vielen kleineren Modellen für Buchstaben, mit deren Hilfe bei der Dekodierung dynamisch Wortmodelle erzeugt werden (siehe Unterabschnitt 6.1.1). Die Anzahl der Zustände jedes Buchstabenmodells ist abhängig von der Art der Initialisierung des HMMs:

S	c	h	r	i	f	t
$[0 \dots 3]$	$[4 \dots 5]$	$[6 \dots 8]$	$[9 \dots 10]$	$[11 \dots 12]$	$[13 \dots 16]$	$[17 \dots 20]$

Abbildung 6.8. – Beispiel für ein Alignment eines Wortes (oben) durch die Beschriftung mit den zugehörigen Merkmalsvektoren (unten).

- Beim sogenannten „Flatstart“ wird davon ausgegangen, dass noch keine Informationen zur Komplexität der einzelnen Buchstaben und damit zur benötigten Zustandsanzahl jedes Buchstabenmodells vorliegen. Jedes Buchstabenmodell besteht hier aus vier Zuständen. Für jeden Zustand liegen ebenfalls noch keine Informationen zur Emissionsverteilung vor, weshalb für die Mischverteilungsgewichte die Gleichverteilung angenommen wird.
- Wurde bereits ein Modell erstellt, so besteht die Möglichkeit, die verwendeten Trainingsdaten zu beschriften (engl. *labeln*). Hierbei wird für jedes Trainingsdatum, und somit für jedes Wort, ein sogenanntes Alignment gesucht, dass jedem Buchstaben des Wortes die wahrscheinlichste (Teil-) Sequenz von Merkmalsvektoren zuordnet. Betrachtet man beispielsweise die in Abbildung 6.8 dargestellte Beschriftung für das Wort „Schrift“ so wurden für den ersten Buchstaben „S“ vier Merkmalsvektoren zugeordnet, die in dem Modell mit hoher Wahrscheinlichkeit von vier Zuständen ausgegeben wurden. Wird ein Modell mithilfe dieser Beschriftung erstellt, so orientieren sich die Zustandsanzahlen jedes Buchstabens an den Beschriftungen der Trainingsdaten. Die initiale Emissionsverteilung der Zustände wird hier ebenfalls anhand der Alignments geschätzt.

Die Parameter des HMM (Zustandsübergangswahrscheinlichkeiten und Emissionsdichten) werden mithilfe der erzeugten Verteilungen unter Anwendung des *Baum-Welch*-Algorithmus statistisch geschätzt und verbessert. Dies geschieht in mehreren Iterationen durch die Optimierung der Produktionswahrscheinlichkeit: für ein Modell λ und eine Emissionsfolge $\mathbf{O}$ gilt mit einem verbesserten Modell $\hat{\lambda}$

$$P(\mathbf{O}|\hat{\lambda}) \geq P(\mathbf{O}|\lambda), \tag{6.6}$$

wobei P die Produktionswahrscheinlichkeit bezeichnet. Die Anzahl der Iterationen ist dabei im Grunde frei wählbar, eine Terminierung, wenn die erreichte Verbesserung unter einen gewählten Schwellwert fällt, ist allerdings vorteilhaft. Eine genauere Beschreibung des Baum-Welch-Algorithmus ist in [Fin08] zu finden.

Das erstellte Modell wird im Anschluss an das Training evaluiert. Das Kriterium für die Qualität des Modells, was für TabScript hierbei angewendet wird, ist die Wortakkuratheit (im Folgenden auch vereinfacht Erkennungsrate genannt). Die Wortakkuratheit gewichtet die Anzahl von Verwechslungen (N_V), Einfügungen (N_E) und Auslassungen (N_A) von Symbolen [Eul06] und soll anhand eines Beispiels (nach [Eul06]) erläutert werden. Sei durch

Test:	a	b	c	d	f
Erkennung:	a	c	d	e	e

ein Test des Erkenners für die Symbolfolge „abcdf“ gegeben. Das HMM erkennt die Symbolfolge „acdee“. Zur Bestimmung der Wortakkuratheit wird nun eine Ausrichtung beider Symbolfolgen gesucht, die den erzeugten Fehler minimiert. Dazu gibt es häufig mehrere Möglichkeiten. Eine Lösung für obiges Beispiel ist etwa durch

Test:	a	b	c	d	-	f
Erkennung:	a	-	c	d	e	e

beschrieben. In dieser Aufstellung ergeben sich drei Fehler: die Auslassung von „b“, das Hinzufügen von „e“ und die falsche Erkennung von „f“. Die Wortakkuratheit ist nun definiert durch [Eul06]

$$WA = \frac{N_W - N_V - N_E - N_A}{N_W}. \tag{6.7}$$

N_W beschreibt dabei die Länge der Test-Symbolfolge. Für das Beispiel ergibt sich $WA = \frac{3}{6}$.

Die Evaluierung geschieht anhand einer Kreuzvalidierung, bei der die Trainingsdaten in fünf gleichgroße Mengen aufgeteilt werden. Auf vier dieser Mengen wird das Modell trainiert und anschliessend auf der verbleibenden Menge getestet. Nach allen fünf Durchläufen (jede Menge wird einmal als Test verwendet) wird die Erkennungsrate über alle Tests gemittelt. Zur Anwendung wird davon das Modell mit der höchsten Erkennungsrate gewählt und auf den zu Anfang aussortierten 10% des gesamten Datensatzes getestet. Dieser Test ist zur Bestimmung der Praxistauglichkeit des Modells wichtig, da hier auf für das Modell „unbekannten“ Daten gearbeitet wird. Die Auswahl der Meta-Parameter des Modells (bspw. Anzahl der Klassen und der Baum-Welch Iterationen) erfolgte vor Beginn des Trainings und wird in Unterabschnitt 6.2.5 erläutert.

6.2. Esmeralda

Für die Klassifikation sowie das Training der Handschriftdaten wurde das ESMERALDA-Toolkit[2] verwendet. Das *integrated **E**nvironment for **S**tatistical **M**odel **E**stimation and **R**ecognition on **A**rbitrary **L**inear **D**ata **A**rrays* enthält Tools, die das Trainieren und Evaluieren eines Hidden-Markov-Modells mit der zugehörigen Dekodierung und Klassifikation ermöglichen. ESMERALDA, ursprünglich in C geschrieben, wurde für TabScript, das auf Java basiert, mithilfe des *Java-Native-Interface* eingebunden. Zuvor musste der C-Code für die ARM-Prozessorarchitektur angepasst werden, was mithilfe des Android-NDK realisiert wurde[3]. Im Folgenden werden die wichtigsten Bestandteile von ESMERALDA beschrieben sowie eine Übersicht über die Verwendung in der Applikation gegeben.

[2] `http://sourceforge.net/projects/esmeralda/`, zuletzt abgerufen am 12.03.2013

[3] nähere Informationen unter `http://www.kandroid.org/ndk/docs/STANDALONE-TOOLCHAIN.html`, zuletzt abgerufen am 09.03.2013

6.2.1. Datenstruktur

ESMERALDA verwendet zur Konfiguration und Speicherung von Modellen eine Mischung aus Klartext- und Binärdateien. Für die Initialisierung und das Training eines Modells werden Merkmalsvektoren benötigt, welche als Binärdateien vorliegen müssen. Deshalb werden die generierten Merkmalsvektoren einer Trajektorie in sogenannten *UFV*-Dateien (kurz für ***U**nified **F**eature **V**ector*) gespeichert. Werte der Merkmale werden dabei binär hintereinander in die Datei geschrieben. Alle weiteren Dateien, wie zum Beispiel die Trainingsdaten-Zuweisung oder Konfigurationsdateien, werden als Klartextdateien erstellt.

6.2.2. Initialisierung

Die Initialisierung bei ESMERALDA erfolgt in zwei Schritten. Zunächst müssen die Cluster über den Trainingsdaten erstellt werden. Dazu werden zum einen Trainingsdaten herangezogen, die in Form von Merkmalsvektordateien (UFV-Dateien) vorliegen. Zum anderen wird die Größe eines Merkmalsvektors benötigt, sowie eine gewünschte Anzahl an Dichten und die gewünschte Dichtefunktion (beispielsweise Gaußverteilung) übergeben. Die verwendete Methode zur Berechnung der Cluster ist der *k-means-Algorithmus* (vgl. [McQ67]).

Im zweiten Schritt wird auf Grundlage der erstellten Cluster (Codebuch) das Hidden-Markov-Modell initialisiert. Dazu wird ein gewünschtes Modell inklusive der Topologie (z.B. linear) übergeben, beispielsweise ein Buchstabenmodell für die Erkennung von Buchstaben. Außerdem werden die gewünschte Anzahl an Zuständen pro Modell festgelegt. Daraufhin wird eine Datei mit den Zuständen, sowie eine Datei mit den Verweisen der Zustände zu Modellen (beispielsweise zu Buchstabenmodellen) erstellt.

6.2.3. Training

Ist die Initialisierung erfolgt, wie im vorherigen Abschnitt erklärt, so kann das Hidden-Markov-Modell trainiert werden. Dazu werden neben den erstellten Daten wiederum Trainingsdaten in Form von binären Merkmalsvektor Dateien benötigt, diesmal jedoch inklusive der Beschreibung des durch die Merkmalsvektoren repräsentierten Symbols (z.B. ein Wort). Wenn die bisherige Initialisierung des Modells auf einem Buchstabenmodell basierte, so benötigt das Training bei Wort-basierten Daten, die Definition der Wort-Modelle. Das heißt, es wird eine Datei benötigt, die angibt, aus welchen Buchstaben-Modellen die Wörter bestehen (vgl. Abbildung 6.5 oben). Das Training, das auf dem *Baum-Welch* Algorithmus aufbaut (vgl. [Fin08, Kap.5.7]), erzeugt neben dem aktualisiertem Zustandsmodell mit den Übergangswahrscheinlichkeiten auch ein aktualisiertes Codebuch. Durch wiederholtes Trainieren auf den aktualisierten Daten kann das Modell verbessert werden.

6.2.4. Dekodierung

Der letzte Schritt besteht aus der Dekodierung des Modells, die auf dem Viterbi-Algorithmus beruht (vgl. Unterabschnitt 6.1.1). Zur Dekodierung werden die Testdaten in Form von Merkmalsvektoren verwendet. Hinzu kommt das Codebuch, sowie das Zustandsmodell mit den Übergangswahrscheinlichkeiten. Als weitere Parameter werden die Definitionen des Modells, sowie Parameter zur Größe des Suchraums (auch *Beambreite* genannt) innerhalb des HMM angegeben. Auch kann eine Wortstrafe angegeben werden, die die Entscheidung mit beeinflusst, indem kurze oder lange Wörter bzw. Buchstaben bei der Erkennung bevorzugt werden (vgl. [ST95, S.266]). Das Ergebnis der Dekodierung ist eine Segmentierung der Daten, aus der das erkannte Datum herausgelesen werden kann.

6.2.5. Konfiguration

Die Konfiguration von ESMERALDA und somit des Hidden-Markov-Modells ist der letzte Schritt zur Auswertung der Handschrifterkennung. Eine falsche Konfiguration kann das Ergebnis stark beeinflussen, weshalb verschiedene Einstellungen evaluiert werden mussten, um ein für die Anwendung passendes Modell zu erstellen.

Das für die Applikation verwendete Modell wurde mithilfe der Unipen-Datenbank erzeugt (vgl. Unterabschnitt 3.3.2). Die Erstellung erfolgte im Batch-Betrieb. Dafür wurden die Daten der Unipen-Datenbank eingelesen und mit den in den vorherigen Kapiteln vorgestellten Methoden verarbeitet. Dadurch wurde gewährleistet, dass sowohl Traingsdaten der Datenbank, als auch Trajektorien- bzw. Merkmalsdaten, die auf dem Tablet erzeugt werden, vergleichbar bleiben.

Für das HMM wurden im ersten Initialisierungsschritt Buchstabenmodelle erzeugt (Flatstart). Aufbauend darauf wurden im weiteren Verlauf Wortmodelle generiert, mit denen auch die Applikation arbeitet. Für die Bestimmung weiterer Parameter mussten mehrere Evaluierungsschritte vollzogen werden, die in Unterabschnitt 7.4.2 beschrieben werden. Als praxistauglich erwiesen haben sich folgende Meta-Parameter:

- Anzahl der Dichten: 1400
- Dichtefunktion: Gauß
- Modelltopologie: Linear
- Modelllänge: 4
- Angenommenes Auftreten pro Modell: 100
- Modellart: Wortmodelle auf Buchstabenmodellen aufbauend
- Training-Iterationen: 18
- Beambreite: 75
- Wortstrafe: 15

7. Evaluierung

Allen umgesetzten Softwarekomponenten lagen gewisse Designentscheidungen zu Grunde, welche im Folgenden erläutert und evaluiert werden. Abschnitt 7.1 beschreibt den Entwicklungsprozess zur Gestaltung der Benutzeroberfläche und Funktionalität von TabScript. Die Auswahl der Vorverarbeitungsschritte und Merkmale, welche einen großen Einfluss auf die Erkennungsleistung haben, werden in Abschnitt 7.2 und Abschnitt 7.3 dargelegt. In Abschnitt 7.4 werden die Entwicklung der Handschrifterkennung anhand unterschiedlicher Trainingsdaten und finale Ergebnisse zur Erkennungsleistung präsentiert. Den Abschluss des Kapitels bildet die Beschreibung und Auswertung zur durchgeführten Nutzerbefragung in Abschnitt 7.5.

7.1. Applikation

Die ersten Entwürfe der graphischen Oberfläche der Anwendung TabScript sind in Abbildung 7.1 dargestellt. Ursprünglich waren zwei getrennte Bildschirm-Ansichten für die Anzeige der Listen und der Aufgaben geplant. In einer Menüleiste am oberen Bildschirmrand war ein Button für das Anwendungsmenü vorgesehen, der Ansichts-Name sowie zwei Buttons zum Löschen oder Hinzufügen von Listen bzw. Aufgaben.

Im unteren Bereich der beiden Ansichten sollten dann jeweils die Namen aller erstellten Listen bzw. der Aufgaben der aktuell ausgewählten Liste angezeigt werden. Für Aufgaben waren Kästchen vorgesehen, mit Hilfe dessen sich eine Aufgabe als erledigt markieren lassen sollte. Beide Ansichten sollten jeweils den gesamten Bildschirm abdecken. Für den Übergang von der Listenansicht zur Aufgabenansicht für eine konkrete Liste, war geplant, dass dieser durch einen Klick auf eine Liste in der Listenansicht ausgelöst werden sollte.

Während des Entwicklungsprozesses der Anwendung haben sich als Folge von wiederholten Benutzertests einige Änderungen an der graphischen Oberfläche, wie sie zunächst im Pflichtenheft (siehe Anhang C) geplant war, ergeben. Die finale Version dieser ist in Abbildung 7.2 dargestellt.

Ein zentraler Unterschied zum alten Design aus dem Pflichtenheft besteht darin, dass die neue Version der Oberfläche (auf Endgeräten mit großem Display) aus einer zweispaltigen Ansicht besteht, welche sowohl die Listen als auch die Aufgaben einer konkreten Liste anzeigen kann. Diese Ansicht ersetzt beide Bildschirm-Ansichten aus dem ursprünglichen Entwurf. Der Grund für das Vornehmen dieser Änderung liegt in der besseren Übersichtlichkeit des neuen Designs. Dadurch, dass sich Listen und Aufgaben in einer Ansicht befinden, wird das ständige Wechseln zwischen den beiden Ansichten

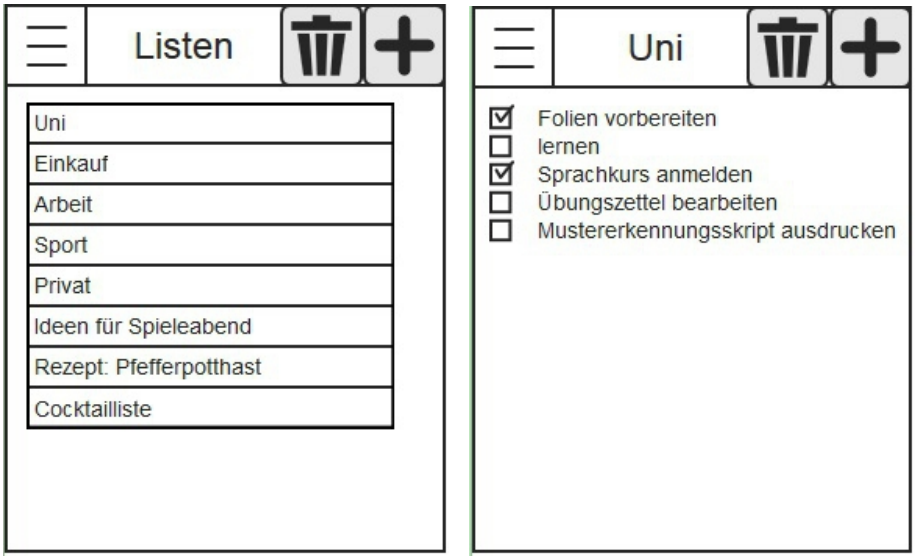


Abbildung 7.1. – Ursprünglicher Entwurf der graphischen Oberfläche der Anwendung.

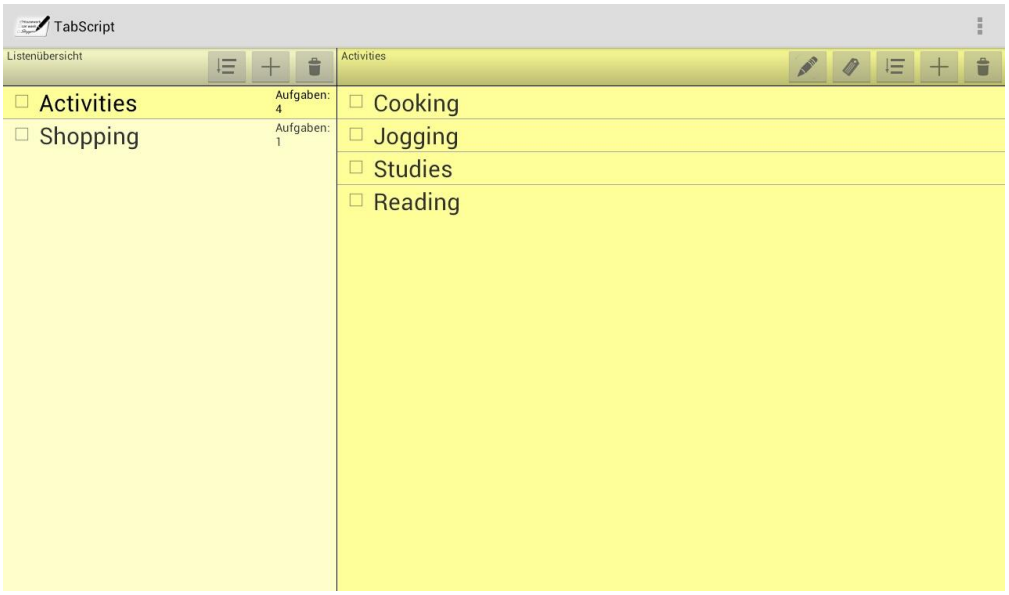


Abbildung 7.2. – Überblick über die graphische Oberfläche von TabScript.

vermieden. Alle erstellten Listen sind somit ständig im Blickfeld des Nutzers und auch die Inhalte verschiedener Listen lassen sich wesentlich schneller anzeigen.

Des Weiteren wurden ein paar kleinere Modifikationen am Erscheinungsbild und der Funktion der Oberfläche vorgenommen: Abbildung 7.2 zeigt, dass deutlich mehr Menü-Buttons in die Ansicht hereingenommen wurden als dies ursprünglich geplant war. Neben den Buttons zum Löschen und Hinzufügen von Aufgaben, welche bereits in Abbildung 7.1 zu sehen sind, enthält die finale Version auch Buttons zum Sortieren von Listen und Aufgaben nach verschiedenen Kriterien. Hinzugekommen sind außerdem ein Button zum Bearbeiten des Listennamens der ausgewählten Liste, sowie ein Button, über den eine Liste mit einem Tag versehen werden kann. In der Listenübersicht wurde weiterhin eine Anzeige mit der Anzahl der enthaltenden Aufgaben der jeweiligen Liste, hinzugefügt.

Beibehalten wurde zum einen der generelle Aufbau der Listen- und Aufgabenansicht mit einer oberen Menüleiste und den Einträgen in Form einer Auflistung darunter. Zum anderen wurde auch die Option zur Kennzeichnen von Aufgaben als „erledigt“ übernommen. Ergänzt wurde jedoch eine Anzeige, die auch für ganze Listen angibt, ob sie abgearbeitet sind. Die Namen der beiden Ansichten für Listen und Aufgaben wurden leicht abgewandelt in „Listenübersicht“ und „Aufgabenübersicht“.

7.2. Vorverarbeitung

Zur Bewertung der implementierten Vorverarbeitungsmethoden ist es notwendig, sich nicht allein auf das subjektiv verbesserte Erscheinungsbild der Trajektorie zu fokussieren. Die Aufbereitung der Eingabedaten in die statistische Auswertung verlangt vielmehr eine Begutachtung des gesamten Klassifikationssystems [Nie83]. Folglich lässt sich die Sachdienlichkeit von verschiedenen Eingriffen in die Trajektorie unter anderem mit Verbesserungen in der Erkennungsleistung feststellen. Die in diesem Abschnitt evaluierten Implementierungen der in Kapitel 4 erläuterten Vorverarbeitungsschritte sind

- Glättung,
- Skalierung,
- Steigungskorrektur,
- Neigungskorrektur und
- Neuabtastung.

Grundlage der Evaluierung ist ein Datensatz von rund 19000 Schriftproben aus der Unipen-Datenbank, mit dessen Hilfe diverse Kombinationen von Vorverarbeitungstechniken analysiert wurden. Der Evaluierungsdatensatz bietet Druck- und Schreibschrift sowie gemischte Schreibtypen in ausgewogenem Verhältnis, womit dem breiten Spektrum an real existierenden Erscheinungsformen von Handschrift Rechnung getragen wird. Bei der Bewertung einzelner Vorverarbeitungsschritte wurde aus dem

vorgestellten Datensatz ein Referenzwert für die Erkennungsleistung nur mit essentiell benötigter Vorverarbeitung ermittelt. Anschließend gaben Experimente mit hinzugeschalteten Methoden Aufschluss darüber, inwiefern diese einen feststellbaren Effekt auf die Handschrifterkennung hatten. In den Testläufen wurden folgende Merkmalsberechnungen genutzt: Schreibrichtung (Sinus und Cosinus), Krümmung (Sinus und Cosinus), Seitenverhältnis, Stiftdruck sowie die dynamischen Varianten (s. Abschnitt 5.5). Als Schriftmodell wurde ein Hidden Markov Model mit linearen Buchstabenmodellen eingesetzt, welches über 15 Iterationen trainiert wurde (s. Kapitel 6).

Bei der Erprobung diverser Implementierungen zeigte sich, gerade vor dem Hintergrund der Android-Plattform und der benutzten Unipen-Datenbank, dass die Vorverarbeitungsschritte der Skalierung auf eine Zielgröße und der Neuabtastung notwendig waren, um die Transkription überhaupt betreiben zu können. Ein initialer Versuch ohne jegliche Vorverarbeitung führte zu einem vorzeitigen Abbruch der Erkennung.

Die zur Verfügung stehenden Evaluierungsdaten aus der Unipen-Datenbank boten teilweise eine nicht ausreichende Trajektorienpunktemenge und -dichte, sodass die Segmentierung anhand maximaler lokaler Krümmung (siehe Abschnitt 5.4) große Teilmengen von Trajektorienabschnitten gänzlich verwarf, da auf den extrahierten Segmenten keinerlei verbleibende Datenpunkte existierten. Die spärliche Datenmenge musste demnach durch Neuabtastung aufbereitet werden, um Klassifikation durchführen zu können.

Weiterhin erwies sich die Notwendigkeit einer Skalierung, da einige Experimente mit ausschließlicher Neuabtastung äußerst schlechte Erkennungsraten von unter einem Prozent lieferten. Schwankende Schriftgrößen im Evaluierungsdatensatz sorgten dafür, dass die Merkmalsrepräsentationen für einzelne Buchstaben zu undifferenziert waren. Die Skalierung ermöglichte zusammen mit der als notwendig erachteten Neuabtastung auf dem Evaluierungsdatensatz eine Wortgenauigkeit von 72,17%.

Neuabtastung und Skalierung stellen somit eine funktionstüchtige Konfiguration dar, sodass die Erkennungsleistung auf dem Evaluierungsdatensatz als Referenzwert für die weiteren Vorverarbeitungsmethoden verwendet wird.

Vorverarbeitung	Wortakkuratheit	Verbesserung	Konfidenzintervall
Skalierung + Neuabtastung	72,17%	Referenz	± 1,5%
Gauß-Glättung	72,82%	+ 0,65%	± 1,5%
Steigungskorrektur	76,43%	+ 4,26%	± 1,6%
Neigungskorrektur	75,80%	+ 3,63%	± 1,6%

Tabelle 7.1. – Die Vorverarbeitungsschritte der Glättung, Steigungskorrektur und Neigungskorrektur wurden jeweils einzeln zu den essentiellen Schritten der Skalierung und Neuabtastung hinzugefügt und evaluiert.

Wie in Tabelle 7.1 dargestellt, wurden zu den zum Erkennungsbetrieb benötigten Schritten die Implementierungen von Gauß-Glättung, Steigungskorrektur bzw. Neigungskorrektur hinzugefügt, um den individuellen Effekt auf die Erkennungsleistung am Ende der Erkennungspipeline feststellen zu können. Dabei erzielte die Glättung

der Trajektorie mit einem diskreten Gauß-Filter, der in Abschnitt 4.1 erläutert wird, eine Verbesserung von 0,65%. Eine initiale Rotation der Trajektorie per Steigungskorrektur erzielte eine um 4,26% höhere Wortgenauigkeit, während die Bearbeitung kursiver Schriftelemente durch Neigungskorrektur eine um 3,63% erhöhte Wortgenauigkeit hatte. Das Konfidenzintervall von $\pm$ 1,5% zeigt, dass die Glättung der Trajektorie keinen signifikanten Effekt besitzt, da der gemessene Effekt von 0,65% vom Konfidenzintervall dominiert wird. Bei der Steigungs- und Neigungskorrektur ist unter Beachtung des Konfidenzintervalls von $\pm$ 1,6% indessen eine positive Wirkung auf die Erkennungsleistung nachzuweisen.

Nachdem die einzelnen Effekte der Variation in der Vorverarbeitung auf die Gesamtleistung der Handschrifterkennung auf dem Evaluierungsdatensatz betrachtet wurden, schließt sich nachfolgend eine Betrachtung der Anordnung der benannten Schritte in der Vorverarbeitungskette an. Zum Vergleich dient die Verarbeitungsreihenfolge des *NPen*++-Erkenners [JMW00, JMRW01], die in Abbildung 7.3 veranschaulicht ist.

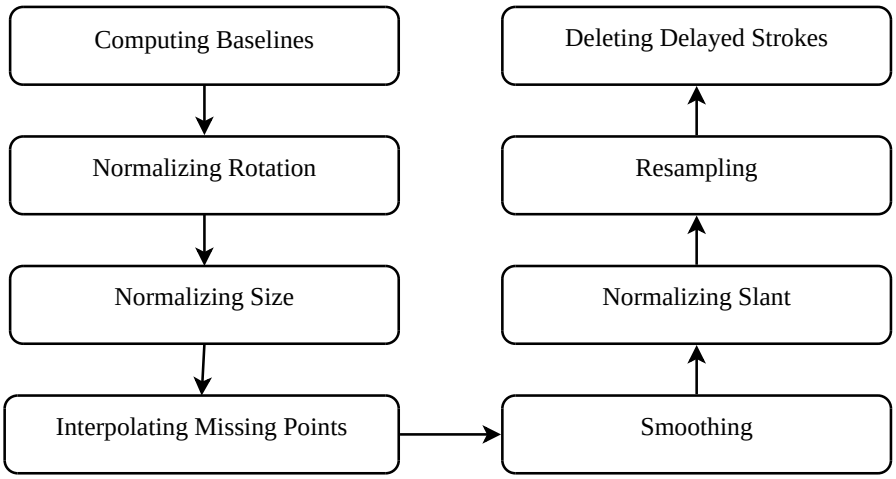


Abbildung 7.3. – Die Vorverarbeitungsreihenfolge des NPen++-Systems zeigt gängige Mechanismen auf (vgl. [JMRW01]).

Die Manipulation der Trajektorie beginnt in der *NPen*++-Pipeline mit der Steigungskorrektur („Normalizing Rotation“), die auf Basis einer zuvor extrahierten Grundlinie („Computing Baselines“) vorgenommen wird. Anschließend wird die Trajektorie auf ein normiertes Maß skaliert („Normalizing Size“). Die Interpolation fehlender Punkte ist in unserer Implementierung eine Unterfunktion der Neuabtastung und verschiebt sich somit bei Betrachtung von Abbildung 7.3 in den Schritt „Resampling“. Die Glättung der Trajektorie (hier „Smoothing“) rangiert in unserer Kette an erster Stelle, was wir aus dem Esmeralda-Werkzeug übernommen haben. Die Schritte der Neigungskorrektur und Neuabtastung kommen bei der Tabscript-Applikation ebenso zum Einsatz, hingegen wird keine Entfernung von *Delayed Strokes* vorgenommen, da deren zuverlässige Detektion sich nicht unter angemessenem Zeitaufwand bewerkstelligen lies.

Zusammenfassend ist die angepasste Verarbeitungskette der Tabscript-Anwendung in Abbildung 7.4 zu sehen.

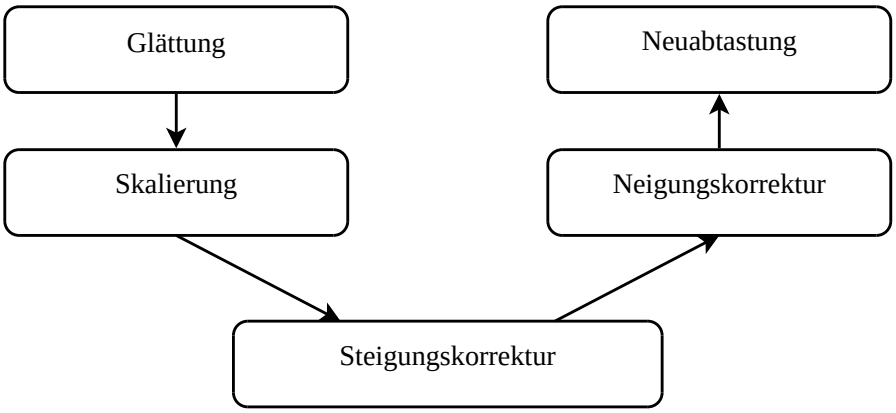


Abbildung 7.4. – Die schematische Darstellung der Tabscript-Vorverarbeitung zeigt die Reihenfolge der einzelnen implementierten Methoden.

Der genutzte Unipen-Datensatz ist in einigen Belangen von den beteiligten Instituten während der Aufnahme der Schriftproben bearbeitet worden, bevor er zur Nutzung freigegeben wurde. So wurden zum Beispiel schräg eingegebene Schriftproben so korrigiert, dass sie anschließend horizontal ausgerichtet in der Datenbank verzeichnet waren. Bei der Mehrheit der Aufnahmegeräte waren darüber hinaus auch Hilfslinien als Orientierungshilfe vorgegeben. Um Reaktionen des Systems auf *herausfordernde* Trajektorien außerhalb des Unipen-Korpus im Hinblick auf die App-Benutzung betrachten zu können, wurden Tests in der Projektgruppe mit selbst geschriebenen Wörtern durchgeführt. Die verwendeten Trajektorien wurden betont kursiv bzw. schräg über den Tablet-Bildschirm geschrieben, um in der Praxis zu erwartende Härtefälle zu simulieren.

Als Grundlage dienten zehn Wörter unterschiedlicher Länge (vier bis 14 Buchstaben), von denen insgesamt 50 Schriftproben innerhalb der Projektgruppe gesammelt wurden. Aus Gründen der Vergleichbarkeit wurde zunächst eine Stichprobe des gewohnten Schreibstils erhoben, um anschließend den Qualitätsverlust durch kursive oder schräge Schreibweise desselben Wortes betrachten zu können. Dabei wurde die grafische Oberfläche der Tabscript-Applikation genutzt, die die drei wahrscheinlichsten Hypothesen über das eingegebene Wort zurückliefert; in der Vorverarbeitungskette waren alle erläuterten Methoden aus Abbildung 7.3 aktiviert.

Abbildung 7.5 zeigt die Ergebnisse der internen Evaluierung für natürliche Schreibweise (links), für absichtlich kursive Handschrift (Mitte) und für schräg (etwa 30 Grad; aufsteigend und absteigend) über den Bildschirm geschriebene Worte (rechts). Bei natürlicher Schreibweise der Probanden wurden 66% der Worte direkt erkannt, 80% bzw. 82% erst an zweiter bzw. dritter Stelle. Der Einfluss kursiver Handschrift sorgte für einen Abfall der korrekten Ersterkennung auf die Hälfte aller Wörter. Unter Einbezug der zweiten Hypothese erreichte das System 66% Wortgenauigkeit und unter den

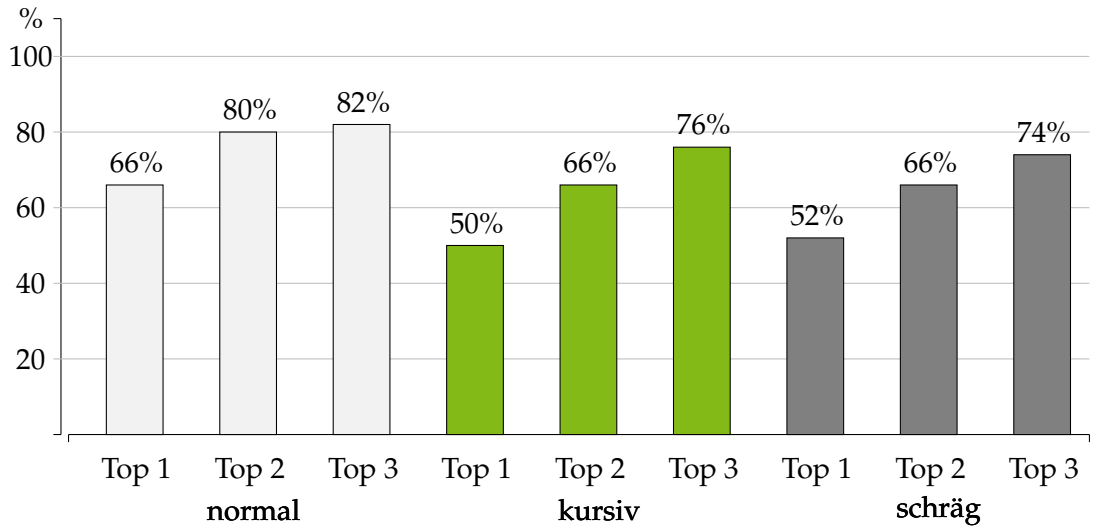


Abbildung 7.5. – Die Erkennungsraten der internen Evaluierung bei normaler (links), kursiver (Mitte) bzw. schräger (rechts) Schreibweise der Probanden.

ersten drei Hypothesen befanden sich bei kursiver Schreibweise insgesamt 76% aller Wörter. Bei Missachtung der Hilfslinien der Tabscript-Schreibansicht in Form von quer über den Bildschirm gezeichneter Trajektorie wurden zunächst 52% der Wörter in erster Instanz korrekt eingeordnet. 66% bzw. 74% Erkennungsleistung ergaben sich durch Inklusion des zweiten bzw. dritten Wortvorschlags.

Es zeigt sich, dass die Erkennungspräzision unter dem Einfluss von kursiver bzw. schräger Handschrift um mehr als 10% schwindet, jedoch sind immer noch rund drei Viertel aller Wörter spätestens an dritter Stelle korrekt entschlüsselt worden. Mit einem Rotationswinkel von zirka 30 Grad um die vorgegebenen Hilfslinien stellte die schräge Schreibweise zudem einen extremen Härtefall dar, da somit auch Schreibfläche auf dem Tablet verloren ging und es sich bei den Probanden als schwierig erwies, lange Wörter adäquat auf dem Bildschirm zu positionieren.

7.3. Merkmale

Die aus den Trajektorien erstellten Merkmale dienen in der Online-Handschrifterkennung als Bindeglied zwischen Trajektorie und Klassifikator. Vor der Verwendung der Merkmale in der finalen Version der Applikation, wurden daher die Erkennungsraten unterschiedlicher Konfigurationen von Merkmalen ausgewertet. Von den in Kapitel 5 vorgestellten Merkmalen wurden die folgenden implementiert und evaluiert:

- Kosinus und Sinus der Schreibrichtung,
- Kosinus und Sinus der Krümmung,
- Aspect Ratio,

- Stiftdruck,
- Ablenkung,
- Linearität,
- Steigung,
- Ascender und Descender.

Zu diesen 11 Merkmalen kommen noch jeweils die dynamischen Merkmale, sodass insgesamt 22 verschiedene Merkmale zur Verfügung stehen. Durch Evaluieren, auf Grundlage der Erkennungsraten auf einem Testdatensatz, haben sich nur die ersten 5 Merkmale als beste Kombination herausgestellt. Insgesamt werden daher unter Hinzunahme von dynamischen Merkmalen 10 Merkmale verwendet, die bei der Erstellung des finalen Schriftmodells und in der Applikation eingesetzt werden. Im Folgenden wird der Evaluierungsprozess der Merkmale aufgezeigt.

Zum Evaluieren der Merkmale wurden zu Beginn der Entwicklung Daten der Städtedemo (vgl. Unterabschnitt 7.4.1) verwendet. Mithilfe dieser Trainingsdaten wurden lineare HMMs für alle Kombinationen der ersten 9 Merkmale inklusive dynamischer Merkmale erzeugt, welche mit Testdaten aus der Städtedemo evaluiert wurde. Die Merkmale Ascender und Descender wurden gesondert betrachtet und werden weiter unten beschrieben.

Tabelle 7.2 listet die Erkennungsraten der 40 besten Merkmalskombinationen der Evaluierung auf. Die vorletzte Spalte zeigt dabei die Erkennungsrate auf den Testdaten in Prozent. Das Konfidenzintervall lässt sich aus der letzten Spalte ermitteln. Das beste Ergebnis mit 90,05% +/-2,9% lieferten im Test die verwendeten Merkmale Schreibrichtung, Krümmung, Aspect Ratio sowie die zugehörigen dynamischen Merkmale.

Beim Vergleich des besten Ergebnisses mit den etwas schlechteren Ergebnissen fällt auf, dass sich die Erkennungsrate geringfügig verschlechtert, während die benutzten Merkmalskombinationen vereinzelt stark variieren. Zugleich werden jedoch Merkmale wie Linearität meistens nicht verwendet, während andere Merkmale wie Schreibrichtung oder Krümmung in den besten Erkennungsergebnissen häufig vorkommen. Werden die Ergebnisse für die einzelnen Merkmale isoliert angeschaut, so wird deutlich, welche Merkmale einen besonderen Stellenwert haben. In Tabelle 7.3 sind die Evaluierungsergebnisse für einzelne Merkmale angegeben. Das Merkmal mit der irrelevantesten Information scheint Linearität zu sein mit einer Erkennungsrate von lediglich 7,46%. Ablenkung und Stiftdruck sind nur wenig besser. Von besonderem Interesse erweisen sich Krümmung und Schreibrichtung. Allein mit dem Schreibrichtungsmerkmal werden über 75% der Testdaten richtig erkannt, was eine 25% bessere Worterkennungsrate als das Krümmungsmerkmal an zweiter Stelle darstellt.

Der Vergleich von Linearität und Ablenkung ist ebenfalls von besonderem Interesse. Wie in Tabelle 7.3 zu sehen, beinhalten beide Merkmale zwar eher weniger Informationen, denoch wird Ablenkung verglichen zur Linearität sehr oft in den besten Merkmalskombinationen verwendet. Beim Betrachten der unterschiedlichen Erkennungsraten in Tabelle 7.4 wird deutlich, dass Ablenkung die etwas besseren Ergebnisse liefert.

Schreibrichtung	Krümmung	Aspect Ratio	Stiftdruck	Ablenkung	Linearität	Steigung	Dyn. M.	Erkennung	Konf.-Int.
true	true	true	false	false	false	false	true	90,05%	2,9%
true	true	false	true	false	false	false	false	89,55%	3%
true	true	false	false	false	false	false	false	89,05%	3,1%
true	true	true	true	false	false	true	false	89,05%	3,1%
true	true	false	true	true	false	false	false	88,31%	3,1%
true	true	true	true	false	false	false	false	87,56%	3,2%
true	true	true	true	true	false	false	false	87,56%	3,2%
true	true	false	true	false	false	true	false	87,31%	3,3%
true	true	false	false	true	false	false	false	86,82%	3,3%
true	true	false	false	false	false	false	true	85,82%	3,4%
true	true	false	true	true	false	true	false	85,82%	3,4%
true	true	true	false	false	false	false	false	85,82%	3,4%
true	true	true	false	true	false	false	false	85,32%	3,5%
true	true	false	false	true	false	true	false	85,07%	3,5%
true	true	true	true	true	false	true	false	84,83%	3,5%
true	true	true	true	false	true	false	true	84,58%	3,5%
false	true	true	false	true	false	false	true	84,33%	3,6%
true	true	true	false	true	false	true	false	84,33%	3,6%
true	true	false	false	false	false	true	false	84,08%	3,6%
true	true	false	false	true	false	false	true	84,08%	3,6%
false	true	true	false	false	false	false	true	83,83%	3,6%
true	true	true	false	false	false	true	false	83,83%	3,6%
false	true	false	false	true	false	true	false	83,33%	3,6%
true	true	false	true	true	false	false	true	83,33%	3,6%
true	false	false	false	false	false	false	true	83,08%	3,7%
false	true	true	true	true	false	true	false	82,59%	3,7%
true	true	true	true	false	false	false	true	82,59%	3,7%
true	true	true	false	true	false	false	true	82,34%	3,7%
false	true	false	true	true	false	true	false	82,09%	3,7%
false	true	true	false	true	false	false	false	82,09%	3,7%
true	false	false	false	true	false	false	true	82,09%	3,7%
false	true	true	false	true	false	true	false	80,85%	3,8%
false	true	false	false	false	false	true	true	80,35%	3,9%
true	true	false	true	false	false	false	true	80,35%	3,9%
true	false	false	false	true	false	false	false	80,1%	3,9%
true	false	false	true	false	false	false	true	79,85%	3,9%
false	true	false	false	true	false	true	true	79,6%	3,9%
true	false	true	false	false	false	true	false	79,6%	3,9%
true	true	true	true	true	false	false	true	79,6%	3,9%
true	true	false	true	true	true	false	true	79,35%	3,9%

Tabelle 7.2. – Auswertung der Merkmale mithilfe der Evaluierung der Klassifikation auf der Städtedemo (hier: die besten 40 Ergebnisse).

Schreibrichtung	Krümmung	Aspect Ratio	Stiftdruck	Ablenkung	Linearität	Steigung	Dyn. M.	Erkennung	Konf.-Int.
true	false	false	false	false	false	false	false	77,11%	4,1%
false	true	false	false	false	false	false	false	51,49%	4,9%
false	false	false	false	false	false	true	false	19,65%	3,9%
false	false	true	false	false	false	false	false	18,16%	3,8%
false	false	false	true	false	false	false	false	12,44%	3,2%
false	false	false	false	true	false	false	false	9,7%	2,9%
false	false	false	false	false	true	false	false	7,46%	2,6%

Tabelle 7.3. – Klassifikationsergebnisse isolierter Merkmale. Schreibrichtung liefert die meisten Informationen.

Mehr noch verschlechtert die Hinzunahme von Linearität die Erkennung, was gegen die Verwendung dieses Merkmals spricht.

Schreibrichtung	Krümmung	Aspect Ratio	Stiftdruck	Ablenkung	Linearität	Steigung	Dyn. M.	Erkennung	Konf.-Int.
true	true	true	false	true	false	false	false	85,32	3,5
true	true	false	true	true	false	false	true	83,33	3,6
true	true	false	true	true	true	false	true	79,35	3,9
true	true	false	true	false	true	false	true	78,61	4
false	false	false	false	true	false	false	true	28,36	4,4
false	false	false	false	true	true	false	true	22,89	4,1
true	true	true	false	true	true	false	false	21,64	4
false	false	false	false	false	true	false	true	18,91	3,8
true	true	true	false	false	true	false	false	18,16	3,8
false	false	false	false	true	false	false	false	9,7	2,9
false	false	false	false	true	true	false	false	8,46	2,7
false	false	false	false	false	true	false	false	7,46	2,6

Tabelle 7.4. – Vergleich der Erkennungsrate induziert durch die Merkmale Ablenkung und Linearität.

Der Informationsgewinn durch die dynamischen Merkmale wird nicht sofort durch die besten Ergebnissen der Evaluierung deutlich. Obwohl das beste Erkennungsergebnis dynamische Merkmale generiert, werden diese in den besten 10 Ergebnissen lediglich ein weiteres Mal erzeugt. Nichtsdestotrotz sind Verbesserungen der Klassifikation durch dynamische Merkmale mithilfe von Tabelle 7.5 zu belegen. Verglichen zu Tabelle 7.3 sind die Ergebnisse signifikant besser geworden, weshalb sie auch in der finalen Implementierung verwendet werden.

Die Evaluation der Merkmale mithilfe der Städtedemo hat geholfen eine Auswahl von Merkmalen zu finden, die eine gute Erkennung auf den Unipen Daten liefern könnte. Die Überprüfung der Werte auf der Unipen-Datenbank war dennoch notwendig, da dies die verwendete Trainingsbasis der Handschrifterkennung von TabScript ist. Auf Grundlage der Datenbank wurde eine beste Erkennungsrate von 79% erzielt und die hier angegebenen Merkmale, Schreibrichtung, Krümmung, Aspect Ratio und dynamische Merkmale daher in die finale Konfiguration aufgenommen (vgl. Unterabschnitt 7.4.2).

Schreibrichtung	Krümmung	Aspect Ratio	Stiftdruck	Ablenkung	Linearität	Steigung	Dyn. M.	Erkennung	Konf.-Int.
true	false	false	false	false	false	false	true	83,08%	3,7%
false	true	false	false	false	false	false	true	61,69%	4,7%
false	false	true	false	false	false	false	true	39,05%	4,7%
false	false	false	false	false	false	true	true	34,33%	4,6%
false	false	false	false	true	false	false	true	28,36%	4,4%
false	false	false	true	false	false	false	true	26,12%	4,3%
false	false	false	false	false	true	false	true	18,91%	3,8%

Tabelle 7.5. – Klassifikationsergebnisse unter Benutzung von isolierten Merkmalen mit zugehörigen dynamischen Merkmalen.

Die Merkmale Ascenders und Descenders wurden separat getestet und aufgrund zweier Kriterien abgelehnt. Zum einen lieferten die Werte auf den angewendeten Test und Trainingsdaten nicht die gewünschten Verbesserungen. Zum anderen war für die Berechnung der Ascenders und Descenders die Implementierung einer Berechnung der Corpus- und Baseline nötig (vgl. Abschnitt 4.1), die jedoch für die Verwendung in der Applikation zu langsam war, sodass sie die Klassifikation deutlich gebremst hätte.

7.4. Erkennung

Für die Erstellung des Modells in TabScript wurden zwei Datensätze verwendet, auf deren Eigenschaften in Unterabschnitt 7.4.1 und Unterabschnitt 7.4.2 eingegangen wird. In Unterabschnitt 7.4.3 wird die Verwendung von Wörterbüchern für die Erkennung erläutert. Anhand des finalen Modells wurde, neben der Erkennung von einzelnen Wörtern, die Leistung bei der Erkennung von Wortfolgen evaluiert. Die Auswertung dieses Versuchs wird in Unterabschnitt 7.4.4 präsentiert.

7.4.1. Modelltraining mit der Städtedemo

Für erste Experimente bei der Erstellung des Schriftmodells für TabScript erschien die Benutzung eines Datensatzes sinnvoll, für den aus vorheriger Anwendung des Lehrstuhls gute Ergebnisse erzielt wurden. Dieser Datensatz enthielt 201 verschiedene, deutsche Städtenamen, deren handschriftliche Version jeweils von 13 Schreibern aufgenommen wurde. Zu Evaluationszwecken wurde dieser Datensatz in Trainingsset (11 Schreiber, 85%) und Testset (2 Schreiber, 15%) aufgeteilt.

Auf dem definierten Testset wurden dabei mit späteren Versionen der Software Erkennungsergebnisse von bis zu 93% erreicht, was sich mit den Ergebnissen des Lehrstuhls deckt. Der Einsatz einer Kreuzvalidierung erschien für den gegebenen Stand der Erkennung und dem nicht finalen Datensatz aufgrund von Aufwand und Berechnungszeit nicht sinnvoll. Ein zusätzlicher Test der Erkennung wurde jedoch durch einen Versuch in Verbindung mit der zu entwickelnden Applikation durchgeführt, indem von zwei Personen jeweils 100 Wörter geschrieben wurden, und die jeweils wahrscheinlichsten drei Hypothesen ausgegeben wurden. Dabei stimmte in 73% der Fälle

das richtige Ergebnis mit der wahrscheinlichsten Hypothese überein, in 84% der Fälle war das richtige Erkennungsergebnis zumindest unter den ersten drei Hypothesen. Die Abweichung der Ergebnisse von Testset und manuellem Test ist vor allem auf die relativ geringe Größe des Datensatzes und damit auf die kleine Auswahl an unterschiedlichen Schriftstilen zurückzuführen.

7.4.2. Modelltraining mit Unipen-Daten

Für das Erstellen des Schriftmodells, welches letztendlich in TabScript Anwendung fand, war ein Datensatz erforderlich, der eine ausreichende Verteilung von Kleinbuchstaben und Großbuchstaben enthielt. Hierfür wurde ein Ausschnitt der Unipen-Datenbank als geeignet angesehen (siehe dazu auch Unterabschnitt 3.3.2). Mithilfe dieser Auswahl an Daten, die in etwa 40000 annotierte Trajektorien umfasst, wurde das finale Modell erstellt und die Parameter der Dekodierung (vgl. Abschnitt 6.2) eingestellt. Im Gegensatz zur Städtedemo (siehe Unterabschnitt 7.4.1) wurden hier jedoch lediglich 10% der Daten als Testset deklariert, um möglichst viele Daten für das Training nutzen zu können. Für die finale Kombination von Merkmalen und Vorverarbeitungsschritten wurde mit diesen Trainingsdaten ein Modell erstellt, dass eine Erkennungsrate von 79% auf den Testdaten erreichte. Dieses Modell wurde auch für die durchgeführte Befragung zur TabScript-Applikation verwendet (siehe Abschnitt 7.5).

7.4.3. Wörterbücher

Die ideale Lösung für eine Android-Applikation mit Handschrifterkennung beinhaltet die Möglichkeit zur Erkennung beliebiger Wörter. Dies ist aus mehreren Gründen in TabScript nicht möglich. Beim Modelltraining mithilfe der Städtedemo (Unterabschnitt 7.4.1) und der Unipen-Datenbank (Unterabschnitt 7.4.2) wurden, wie in den jeweiligen Abschnitten beschrieben, gute Erkennungsergebnisse auf Testdaten erzielt. Diese Testdaten bestehen aus ganzen Wörtern, die vom Modell klassifiziert werden müssen. Um beliebige Wörter erkennen zu können, muss die Erkennungsleistung des Modells so gut sein, dass es nicht mehr auf Konzepte zur Wortdefinition angewiesen ist, um Wörter zusammenzusetzen. In diesem Fall wäre die Erkennungsleistung auf einzelnen Buchstaben ebenso gut. Dieses Ergebnis wurde mit den verschiedenen Modellen in TabScript nicht erreicht. Hierfür lassen sich mehrere Gründe angeben, welche hauptsächlich, aber nicht aussschliesslich, dafür verantwortlich sind. Zum einen umfasst der Datensatz, welcher zum Training des finalen Modells verwendet wurde, zwar in etwa 40000 Wörter, allerdings sind Klein- und Großbuchstaben nicht in der selben Größenordnung vertreten, da es sich um einen englischen Wortsatz handelt. Daraus ergibt sich ein Defizit für das Training von Großbuchstaben, das insbesonders bei der späteren Erkennung von deutschen Wörtern Schwierigkeiten bereitet. Zudem erschie-

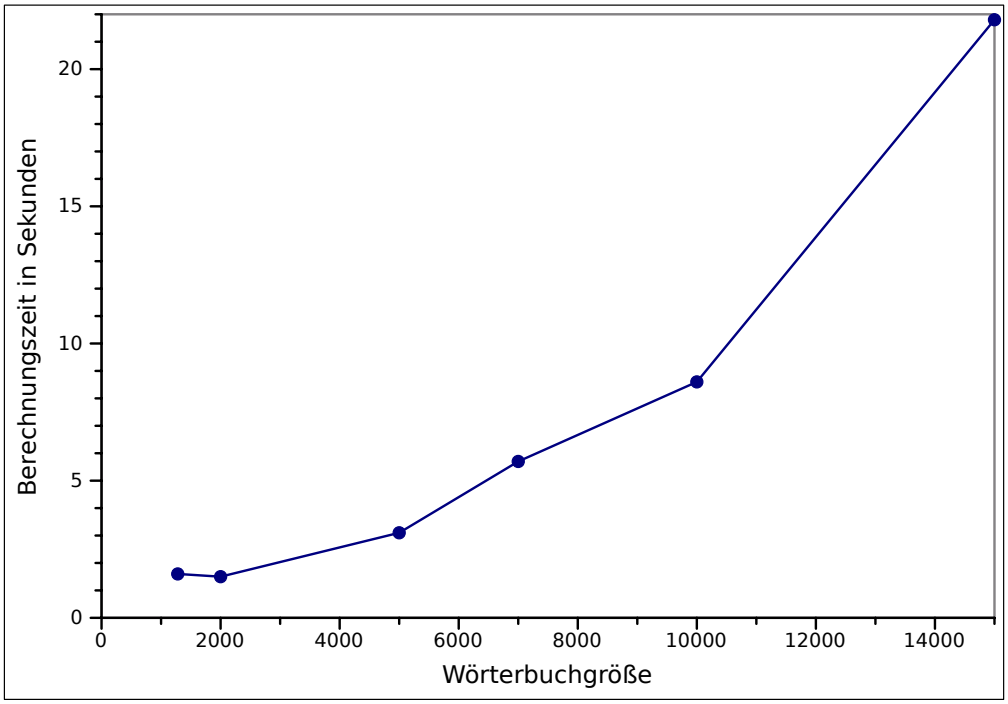


Abbildung 7.6. – Laufzeit der Erkennung in Abhängigkeit der Wörterbuchgröße

nen die Trajektorien der Unipen-Datenbank bereits an einigen Stellen vorverarbeitet[1]. Es kann also vermutet werden, dass die Trainingsdaten das Schriftbild, welches allgemein erwartet wird, etwas verfälschen, indem typische Schrift-„Fehler“ gar nicht erst auftreten.

Da also die Erkennung von einzelnen Zeichen aufgrund der Trainingsdaten nicht möglich ist, werden Wörterbücher verwendet, die jedes von der Applikation erkennbare Wort aus den Buchstabenmodellen (vgl. Unterabschnitt 6.1.1) definieren. Ein Nachteil dieser Methode ist die Laufzeit der Erkennung, welche mit steigender Größe des verwendeten Wörterbuchs zunimmt. Zur Bestimmung einer in der Praxis nutzbaren Wörterbuchgröße wurden die Laufzeiten der Erkennung in Abhängigkeit verschiedener Wörterbuchgrößen gemessen. Abbildung 7.6 zeigt das Ergebnis dieser Messungen, wobei zwischen gemessenen Datenpunkten linear interpoliert wird. Es ist offensichtlich, dass die Laufzeit stark mit wachsender Wörterbuchgröße zunimmt. Die finale Version der Applikation verwendet eine Anzahl von 2500 Wörtern, die zu einer Laufzeit zwischen $1 - 3$ Sekunden pro Wort führt. Die tatsächliche Laufzeit hängt zusätzlich von der Länge des geschriebenen Wortes, der Menge der abgetasteten Punkte und der Leistung des benutzten Android-Gerätes ab. Die gewählte Wörterbuchgröße war auf allen getesteten Geräten praxistauglich.

[1] Es gibt beispielsweise kaum kursive oder schräg geschriebene Wortbeispiele.

Neben der Größe des Wörterbuchs musste auch die Auswahl der Sprache festgelegt werden. Für den allgemeinen Gebrauch enthält TabScript zwei Wörterbücher mit häufigen Begriffen der deutschen und englischen Sprache. Für Präsentationszwecke und die Anwendung der Applikation in der durchgeführten Umfrage (siehe Abschnitt 7.5) wurde ein weiteres Wörterbuch mit Lebensmitteln aufgenommen um einen thematischen Rahmen beim Anlegen von Listen vorzugeben (z. B. Rezepte, Einkaufslisten etc.).

7.4.4. Erkennung von mehreren Wörtern

Im Hinblick auf die Nutzbarkeit der Notizapp wurde die Möglichkeit, mehrere Wörter hintereinander weg zu schreiben, und diese im Anschluss gemeinsam zu erkennen, getestet. Dies sollte die Eingabe natürlicher gestalten, da der Benutzer nicht nach jedem Wort warten muss, bis die Erkennung startet.

Zur Evaluierung haben jeweils neun Probanden je fünf englische und deutsche Sätze geschrieben. Es wurden 171 deutsche und 225 englische Wörter geschrieben, insgesamt also 396 Wörter. Ausgewertet wurde zum einen, wie oft der komplette Satz vollständig richtig erkannt unter den ersten drei Erkennungsergebnissen war. Zusätzlich wurde der Anteil der insgesamt erkannten Wörter ermittelt.

Englisch	Anteil vollständig erkannte Sätze	Anteil erkannte Wörter	Wörter je Satz
Satz 1	5/9	27/36	4
Satz 2	3/9	24/36	4
Satz 2	0	13/36	4
Satz 3	3/9	24/36	4
Satz 4	1/9	11/27	3
Gesamt:	$12/45 \triangleq 26,67\%$	$99/171 \triangleq 57,89\%$	

Englisch	Anteil vollständig erkannte Sätze	Anteil erkannte Wörter	Wörter je Satz
Satz 1	0/9	12/54	6
Satz 2	1/9	25/54	6
Satz 3	1/9	13/36	4
Satz 4	0/9	12/36	4
Satz 5	0/9	24/45	5
Gesamt:	$2/45 \triangleq 4,44\%$	$86/225 \triangleq 38,22\%$	

Deutsch & Englisch	Anteil vollständig erkannte Sätze	Anteil erkannte Wörter
alle Sätze	$14/90 \triangleq 15,56\%$	$185/396 \triangleq 46,71\%$

Tabelle 7.6. – Die Erkennungsrate bei mehreren geschriebenen englischen bzw. deutschen Sätzen. Jeder Satz wurde von neun verschiedenen Schreiben geschrieben. Es wird die Anzahl der erkannten Sätze bzw. Wörter in Abhängigkeit von der Gesamtanzahl Sätze resp. Wörtern angegeben.

Tabelle 7.6 stellt die Ergebnisse in Tabellenform dar. Insgesamt 99 von 171 englischen Wörtern wurden unter den besten Erkennungsergebnissen erkannt, das entspricht 57,9%. Betrachtet man jedoch, wie häufig jeweils der gesamte Satz vollständig unter den ersten drei Erkennungsergebnissen war, liegt die Quote nur bei 12 von 45 Sätzen. Es wurden also nur 26,67% der Sätze komplett erkannt. Bei den deutschen Sätzen ergab sich eine Worterkennungsrate von 38,22%. Es wurden nur 2 Sätze zur Gänze erkannt, also 4,44%.

Insgesamt wurden also 185 von 396 Wörtern korrekt erkannt, wobei jeweils das Erkennungsergebnis betrachtet wurde, was dem tatsächlichen Satz am ähnlichsten ist. Von den Sätzen wurden 14 von 90 zur Gänze erfasst.

Zusammenfassend lässt sich sagen, dass der Ansatz, mehrere Wörter gleichzeitig zu erkennen, nicht zu einer besseren Benutzbarkeit führt, da diese Methode mit einem hohen Korrekturaufwand seitens des Nutzers verbunden ist.

7.5. Umfrage

Die Projektgruppe hat eine Umfrage erstellt, um die Meinung von Nutzern zur Applikation zusammenzustellen. Dabei handelt es sich um eine Nutzerstudie, die im Folgenden beschrieben wird. In diesem Kapitel werden der Aufbau (Unterabschnitt 7.5.1) und die Ergebnisse der Umfrage zur TabScript-Applikation vorgestellt. Dabei sind die Ergebnisse unterteilt in die Beschreibung der Testpersonen (Unterabschnitt 7.5.2), die erzielte Erkennungsrate (Unterabschnitt 7.5.3), die Benutzerzufriedenheit (Unterabschnitt 7.5.4) und die Verbesserungsvorschläge (Unterabschnitt 7.5.5), welche von den Testpersonen gemacht worden sind.

7.5.1. Aufbau

Für die Erstellung der Umfrage zu TabScript fiel das Augenmerk besonders auf zwei Kernbereiche, die Qualität der Handschrifterkennung und die Benutzerfreundlichkeit. Diese Ergebnisse wurden mit Informationen zu der jeweiligen Testperson erweitert. Ferner wurden Verbesserungsvorschlägen der Probanden gesammelt, um mögliche Defizite in der Applikation aufzudecken.

Die Umfrage wurde unter Verwendung des „Lebensmittel-Wörterbuchs“ (siehe Unterabschnitt 7.4.3) durchgeführt. Damit können die Probanden einen besseren Bezug zum Nutzen der Applikation entwickeln, da ihnen die Möglichkeit gegeben wird ein Rezept zu schreiben.

Die Umfrage wurde mit Hilfe eines Fragebogens (siehe Anhang B) durchgeführt, der eine klare Aufteilung in die zwei oben genannten Hauptabschnitte (Qualität der Erkennung und Benutzerfreundlichkeit), sowie die zwei unterstützenden Bereiche (Informationen zu den Testpersonen und Verbesserungsvorschläge) aufweist. Die ersten drei Fragen beziehen sich auf Informationen zu den Testpersonen. Dazu gehören das Alter und das Geschlecht des Probanden, sowie dieser Links- oder Rechtshändler ist.

Diese Informationen sind vor Allem für die Beschreibung der Umfrage von Nöten. Ferner können für einen größeren Datensatz bei der Erkennung oder der Benutzerfreundlichkeit die Ergebnisse zwischen Männern und Frauen oder Links- und Rechtshändern verglichen werden.

Im zweiten Teil wurden die Probanden gebeten zehn Wörter ihrer Wahl aus dem „Lebensmittel-Wörterbuchs“ in die Write-Activity zu schreiben. Dabei wird darauf geachtet, ob ein geschriebenes Wort erkannt wird. Da TabScript bei der Erkennung die drei wahrscheinlichsten Ergebnisse zur Auswahl angibt (Abschnitt 2.8), wird hierbei auch beachtet, an welcher Stelle das Geschriebene erkannt wird. In Unterabschnitt 7.5.3 wird bei der Erkennungsrate über die Erkennung nur an erster Stelle, an den ersten beiden und an allen dreien zusammen gesprochen. Somit können Fehler teilweise abgefangen werden, die von der Erkennung gemacht wird, weil die Testperson mit einer schwer lesbaren oder stark von der Norm abweichenden Handschrift schreibt.

Der dritte Teil des Fragebogens soll die Zufriedenheit der Benutzer mit einzelnen Bereichen der Applikation (Unterabschnitt 7.5.4) erfassen. Vor dem Erstellen der genauen Fragen ist erörtert worden, zu welchen Dingen sich die Probanden äußern sollen. Hierbei ist eine Dreiteilung in die Funktionalität, die Bedienung und die Optik vorgenommen worden. Die Testpersonen bewerten die Funktionalität mit Noten von „1 = sehr gut“ bis „5 = gar nicht gut“, die Optik von „1 = übersichtlich“ bis „5 = unübersichtlich“ und die Bedienung bzw. Benutzerfreundlichkeit von „1 = sehr intuitiv“ bis „5 = gar nicht intuitiv“.

Im vierten Teil des Fragebogens werden Verbesserungsvorschläge und Anmerkungen der Testpersonen erörtert. Hierbei werden drei Fragen gestellt. Zuerst wird der Proband gefragt, ob ihm Fehler in der Applikation aufgefallen sind, die behoben werden müssen, danach, ob ihm bestimmte Funktionen oder Menüs verbesserungswürdig vorkommen. Neben diesen Verbesserungsvorschlägen ist auch positives Feedback erfasst worden, so lautet die dritte Frage, was der Testperson besonders gut gefallen hat.

Um den Testpersonen eine angenehme Dauer der Befragung zu gewähren, sind viele auf spezielle Dinge bezogene Fragen nicht in den Fragebogen aufgenommen worden. Zu diesen gehören Fragen zur Funktionalität einzelner Buttons und zu Menüfunktionen, sowie dem Markieren oder manuellen Verschieben von Aufgaben oder Listen (siehe Abschnitt 2.4 und Abschnitt 2.3). Stattdessen ist versucht worden die Fragen allgemein zu halten, jedoch keinen Bereich zu vernachlässigen. Da die Erkennung aber der größte Bereich ist, musste hierfür die meiste Zeit genutzt werden.

7.5.2. Testpersonen

Insgesamt wurden 45 Personen im Alter zwischen 19 und 48 Jahren befragt. Der Altersdurchschnitt lag bei 25 Jahren, wobei der Großteil der Befragten (89%) unter 28 Jahre alt war. Eine detaillierte Darstellung der Altersverteilung findet sich in Abbildung 7.7. Da die Umfrage an der Technischen Universität Dortmund durchgeführt wurde, ist unter den Befragten mit einer erhöhten Anzahl an Studenten zu rechnen. An der Umfrage nahmen insgesamt 12 Frauen und 33 Männer teil (vgl. Abbildung 7.8(a)). Somit

war der Anteil männlicher Personen mit 73% überrepräsentiert. Dies ist vermutlich dadurch zu erklären, dass die Befragung unter anderem am Institut für Roboterforschung durchgeführt wurde, welches durch Ihren technischen Schwerpunkt einen erhöhten Anteil männlicher Personen aufweist. Die Verteilung der Händigkeit (Rechtshänder: 91%, Linkshänder: 9%) entsprach der Verteilung in der Normalbevölkerung [Pri12] (vgl. Abbildung 7.8(b)).

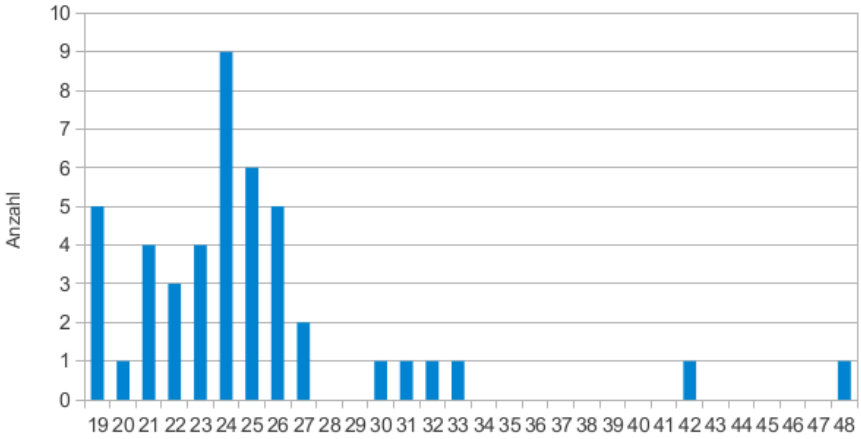
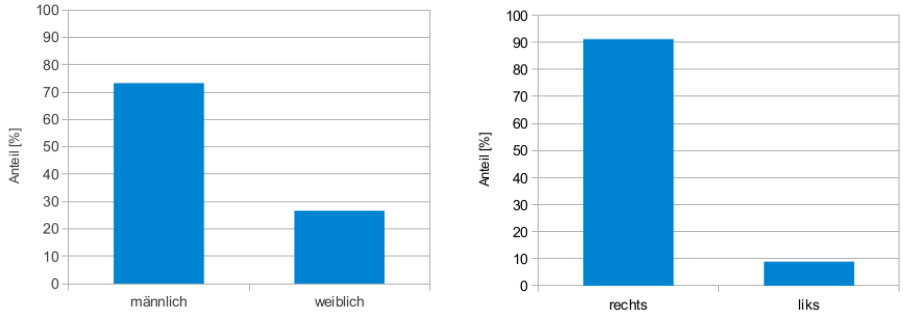


Abbildung 7.7. – Altersverteilung der Umfrageteilnehmer.



(a) (b)

Abbildung 7.8. – (a) Geschlecht und (b) Händigkeit der Umfrageteilnehmer.

7.5.3. Erkennungsrate

Die Auswertung der Umfrage ergab eine Gesamterkennungsrate von 68,44%. Innerhalb dieser Erkennungsrate lassen sich die Werte Top 1, Top 2 und Top 3 differenzieren. Top 1 gibt den Prozentsatz der Wörter wieder, welche an erster Stelle erkannt wurden, Top 2 und Top 3 entsprechend die Wörter, welche unter den ersten beiden bzw. den

ersten drei Vorschlägen aufgelistet wurden. Abbildung 7.9 stellt die Werte Top 1-3 für die Gesamtzahl aller in der Umfrage erfassten Wörter (n = 450) dar.

Während Top 1 bereits einen Wert von 60% aufweist, unterscheidet sich Top 2 nur um 6,44% von diesem, während die Differenz zwischen Top 2 und Top 3 mit 2% noch geringer ausfällt. Hieraus folgt, dass in den Top 3 erkannte Wörter in 88% der Fälle an erster Stelle gelistet werden. Dies hat für die Bedienung der Applikation einen hohen praktischen Nutzen, da direkt erkannte Wörter automatisch übernommen werden und nicht erst manuell aus der Liste aller vorgeschlagenen Wörter ausgewählt werden müssen.

Als nächstes werden die Top 3 Ergebnisse für die einzelnen Umfrageteilnehmer betrachtet. Abbildung 7.10 zeigt diese Werte absteigend sortiert. Es fällt auf, dass die Teilnehmer insgesamt sehr differente Erkennungsraten aufweisen. Diese reichen von 10-100%. Eine genauere Analyse zeigt, dass die Gesamterkennungsrate für den Großteil der Befragten (91%) im Bereich zwischen 40% und 100% liegt. Lediglich vier Personen weisen eine minimale Erkennungsrate von 10% auf. Gründe hierfür können vielfältig sein. Ein Vergleich mit dem Rest der Befragten, z.B. durch Analyse der Händigkeit und des Altersdurchschnitts ist in diesem Fall nicht indiziert, da eine Gruppe von vier Personen zu klein ist, um repräsentative Aussagen treffen zu können. Weitere Faktoren, welche nicht durch den Fragebogen erfasst wurden, können kulturelle Aspekte darstellen, so z.B. die Muttersprache des Nutzers, das Herkunftsland sowie erworbene Fremdsprachen. Denn Unterschiede von Sprachen und Schriften verschiedener ethnischer Gruppen können gegebenenfalls auch Einfluss auf die Handschrift nehmen [SWV06]. Ebenso verhält es sich mit subjektiven Zuständen des Nutzers, wie z.B. Stress und Müdigkeit [SLTWO96].

Abbildung 7.11 zeigt das Verhältnis zwischen Top 3 und dem Anteil nicht erkannter Wörter in Abhängigkeit zu der Wortlänge. Bei dieser Auswertung wurden nur Wörter mit 3-16 Zeichen berücksichtigt, da andere Längen mit weniger als fünf Wörtern keine ausreichend großen Stichproben lieferten (vgl. Abbildung 7.12). Bei Betrachtung der Abbildung fällt auf, dass Wörter umso besser erkannt werden, je mehr Zeichen sie aufweisen.

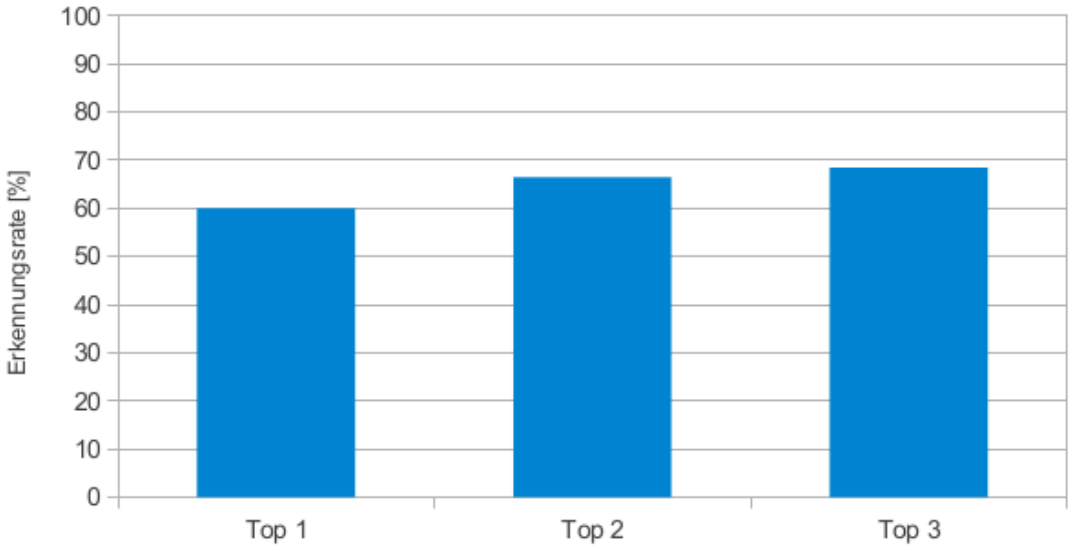


Abbildung 7.9. – Top 1-3 für die Gesamtzahl aller in der Umfrage erfassten Wörter (n = 450).

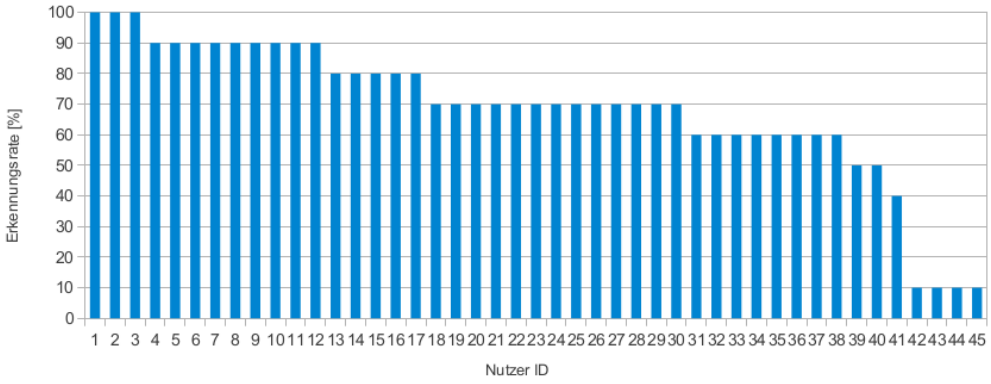


Abbildung 7.10. – Top 3 Ergebnisse der einzelnen Umfrageteilnehmer.

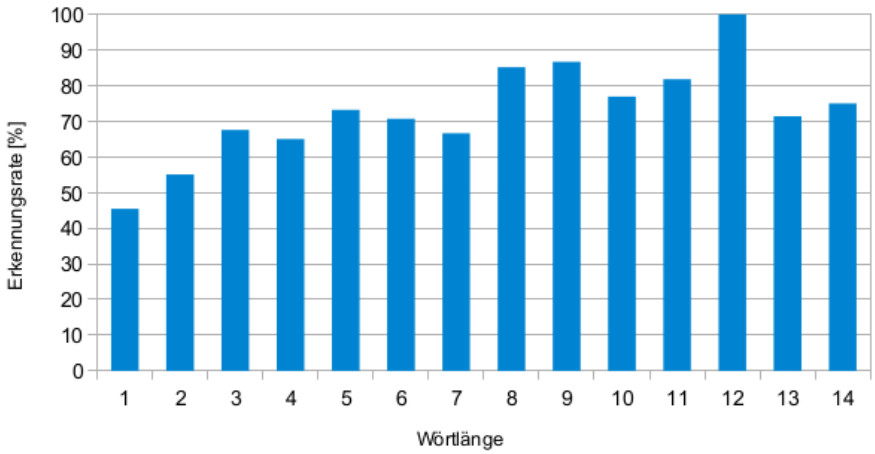


Abbildung 7.11. – Erkennungsrate in Abhängigkeit von der Wortlänge.

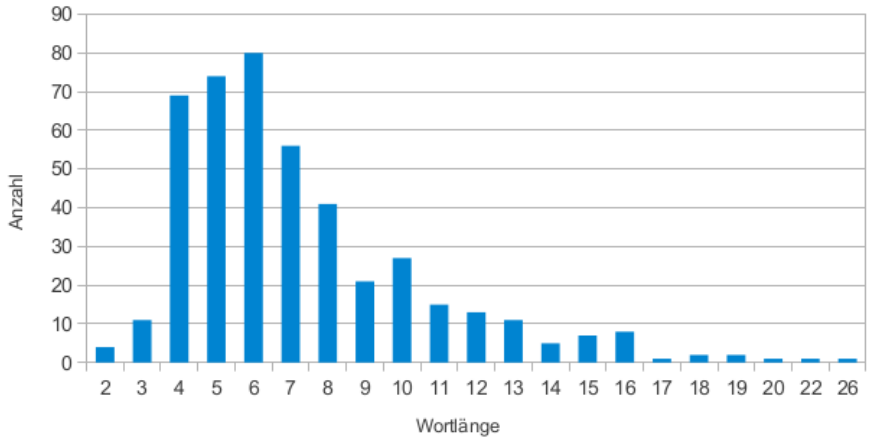


Abbildung 7.12. – Anzahl aller Wörter der Stichprobe einer bestimmten Länge.

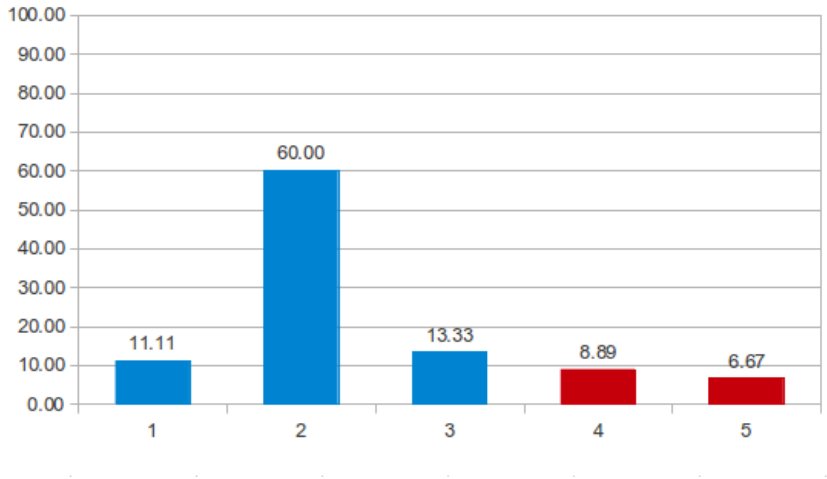


Abbildung 7.13. – Umfrageergebnisse zur Handschrifterkennung.

7.5.4. Benutzerzufriedenheit

Die Benutzerzufriedenheit ist in insgesamt fünf Fragen zu drei übergeordneten Themen der Applikation erfasst worden. Diese Bereiche sind die Erkennung (Abbildung 7.13), sowie die Bedienungsfreundlichkeit (Abbildung 7.14) und die Optik (Abbildung 7.15) der TabScript-Applikation. Alle Bereiche werden mit Noten von „1“ bis „5“ bewertet.

Wie in Abbildung 7.13 zu sehen ist, ist dieser die Handschrifterkennung der Applikation besonders oft mit gut beantwortet worden (60%). Insgesamt ist der Anteil an guten Noten (Noten „1“ und „2“) mit 71,11% sehr erfreulich.

Diese Frage ist jedoch auch gleichzeitig die mit den meisten negativen Bewertungen (Noten „4“ und „5“, insgesamt 15,56%). Hierbei ist aber zu beachten, dass die Beurteilung der Handschrifterkennung eine sehr subjektive ist. So vergeben Testpersonen, deren geschriebene Wörter nur selten erkannt wurden, schlechtere Noten, als jene, die eine überdurchschnittliche Erkennungsrate aufweisen. So hat nur eine der Testpersonen, die eine dieser beiden Noten vergeben haben, bei der eigenen Handschrift eine durchschnittliche Erkennungsrate erzielt, während die anderen darunter liegen. Daher ist hier der Notenschnitt mit 2,1 zwar ein guter, aber im Vergleich zu den anderen eher schlecht.

Bei der zweiten Frage zur Bedienungsfreundlichkeit ist eine deutlich bessere Bewertung zu erkennen als bei der Benotung der Erkennung, da sowohl die Anzahl an schlechten Noten niedriger (insgesamt 4,4%), als auch die der guten (insgesamt 80%) deutlich höher ist. Somit ist es nicht verwunderlich, dass der Notenschnitt mit 1,9 etwas besser ist. Die Bedienungsfreundlichkeit ist für die meisten Testpersonen angenehm, da die Applikation keine unnötigen Funktionen besitzt, sondern auf den Kern fokussiert ist. So sind auch die meisten Verbesserungsvorschläge eher als Zusatz gedacht.

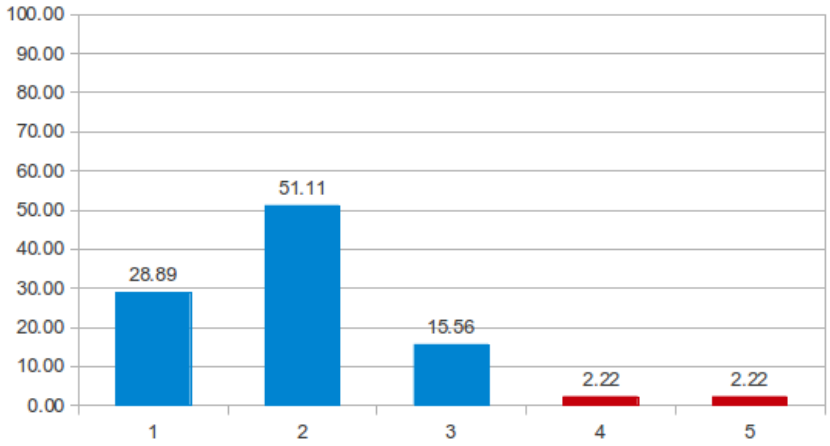
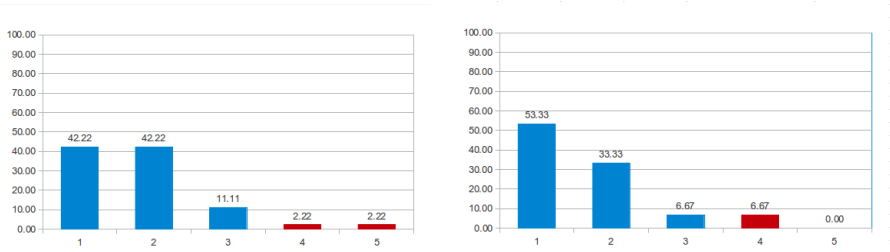
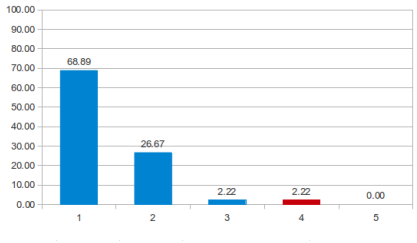


Abbildung 7.14. – Umfrageergebnisse zur Bedienungsfreundlichkeit.



(a) (b)



(c)

Abbildung 7.15. – Umfrageergebnisse zum Aussehen der TabScript-Applikation (a) Bewertung der Benutzeroberfläche insgesamt. (b) Bewertung der Buttons und ihrer Anbringung. (c) Bewertung der Teilung des Bildschirms in Listen und Aufgaben.

Dementsprechend haben auch vereinzelt Probanden ihre Ideen (Unterabschnitt 7.5.5) als möglich, aber nicht unbedingt notwendig angesehen.

Die Optik der Applikation ist, wie in Unterabschnitt 7.5.1 beschrieben, in drei unterschiedliche Fragen aufgeteilt. Dabei beurteilen die Probanden das erst das gesamte Erscheinungsbild der GUI (Abbildung 7.15(a)), dann die Anbringung der Buttons, sowie die Intuitivität dieser (Abbildung 7.15(b)) und zuletzt die Teilung des Startbildschirms in Listen und Aufgaben (Abbildung 7.15(c)).

Die graphische Oberfläche schneidet bei der Umfrage mit einem Notenschnitt von 1,8 gut ab. Die über 80% guten Bewertungen, sowie die nicht mal 5% schlechten bestätigen den guten Eindruck, den die Optik der Applikation bei den Probanden hinterlassen hat. Bei dieser Frage haben einige Testpersonen die Farbwahl kritisiert und wünschen sich weitere Themes (Unterabschnitt 2.7.1) als die von uns gewählten. Jedoch bestätigen auch viele die Meinung unserer Entwicklergruppe, dass eine Notiz-App auch auf Notizzettel farbenem Hintergrund geschrieben werden sollte.

Auch die Teilung des Bildschirms und die Anbringung der Buttons wird von den Probanden gut angenommen. Das Splitten der Applikation in Listen und den dazugehörigen Aufgaben ist im Schnitt mit einer 1,7 bewertet, während die Intuitivität der Buttons mit einer 1,3 die beste Durchschnittsnote aller Fragen aufweist. Beide Fragen sind wie die allgemeine graphische Oberfläche von weit über 80% der Teilnehmer mit einer „1“ oder „2“ beantwortet worden, wobei die Buttons sogar über 90% „1“ und „2“ erhalten haben.

Allgemein kann gesagt werden, dass alle Teilbereiche der Applikation ein gutes Feedback erhalten haben. Sowohl die Funktionalität der Handschrifterkennung, als auch die Bedienungsfreundlichkeit und die Optik wurden mehrheitlich positiv bewertet. Jedoch zeigt Unterabschnitt 7.5.5, dass immer noch Verbesserungspotenzial in der Applikation steckt.

7.5.5. Verbesserungsvorschläge

Am Ende der Umfrage wurde den Nutzern die Möglichkeit gegeben, Kommentare in Form von Lob und Kritik bzw. Verbesserungsvorschlägen abzugeben. Die Ergebnisse werden in den folgenden Abschnitten vorgestellt.

Verbesserungsvorschläge

Ein großer Teil der Befragten (36%) kritisierte eine zu kurze Schreibdauer. Dabei kam es zu einem vorzeitigen Starten der Erkennung noch bevor das Wort vollständig aufgeschrieben wurde. Das zurückgelieferte Ergebnis war somit meistens fehlerhaft. Insbesondere vor dem Setzen von i-Punkten trat dieses Fehlverhalten gehäuft auf. Bei nachfolgenden Eingaben konnte es jedoch leicht umgangen werden, da die Zeit bis zum Starten der Erkennung flexibel eingestellt werden kann. Des Weiteren wurde das Design der Applikation bemängelt. Vor allem die Farbauswahl, welche sich an dem typischen Gelbton einer Haftnotitz orientiert erschien vielen Nutzern eher unschön. Es wurde der Wunsch nach einer moderneren Oberfläche mit mehr „Sexappeal“ geäußert.

Einige Benutzer zeigten Schwierigkeiten aus der Schreibansicht in die Listenansicht zurückzugelangen. Sie suchten instinktiv einen „Zurück"-Button neben den anderen Buttons der Ansicht. Der Wechsel zur Listenansicht gelang jedoch nur über die Navigationsleiste. Obwohl dies den offiziellen Android Design-Prinzipien [ADP] entspricht, könnte eine Abweichung von diesen Richtlinien die Benutzerfreundlichkeit steigern. Weiterhin wurde der Wunsch geäußert, eine Hilfsfunktion zu integrieren, welche den Benutzer bei der Bedienung der Applikation unterstützt. Unter anderem sollen durch diese spezielle Features, wie z.B. Tags oder der Menü-Button 1 der Aufgabenansicht (siehe Kapitel 2), welche nicht durch alle Benutzer sofort verstanden werden, gesondert erklärt werden. Einige Benutzer würden eine Einführung von Unteraufgaben als sinnvoll erachten. Mit diesen ließen sich bereits erstellte Aufgaben näher spezifizieren, z.B. könnte die Liste „Hausaufgaben" die Aufgaben „Mathe, Deutsch, Englisch" und „Deutsch" die Unteraufgaben „Zusammenfassung schreiben, Bescheinigung nachreichen". Hierbei ist jedoch der Informationsgewinn und eine damit verbundene mögliche Minderung der Übersichtlichkeit gegeneinander abzuwägen. Des weiteren wurde eine Zeichenfunktion vorgeschlagen, mit der sich einfache Grafiken erstellen lassen. Auf diese Weise können Informationen gespeichert werden, welche sich nicht in schriftlicher Form festhalten lassen. Beispielsweise können so Anfahrtsskizzen angefertigt werden. Weitere Verbesserungsvorschläge wurden nur vereinzelt geäußert, so z.B. ein direktes Aufschreiben ohne zusätzliches Öffnen der Schreibansicht, das Einführen eines „Weiter"-Buttons zur Eingabe mehrerer Aufgaben, das Einführen von Mengenangaben sowie größere und zum Teil aussagekräftigere Überschriften.

Lob

Auffällig häufig wurde im Rahmen der Umfrage „die Handschrifterkennung" gelobt, was für die Befragten sowohl die Erkennungsrate als auch die Erkennungsgeschwindigkeit umfasste. Die Idee, die Texteingabe über eine Handschrifterkennung zu realisieren, erschien vielen Umfrageteilnehmern nützlich und die Umsetzung als ToDo-Anwendung von besonderem praktischen Interesse. Weiterhin wurden die Übersichtlichkeit der Startansicht, die klare Aufteilung der Bereiche für Listen und Aufgaben sowie die schnelle Verständlichkeit der „Sort"-, „Hinzufügen"- und „Entfernen"-Buttons genannt. Das Klappmenü der Schreibansicht war für einige Benutzer eine wichtige Hilfestellung. Es zeigt bereits erstellte Listen bzw. Aufgaben und vermeidet somit ein unnötiges Zurückkehren zur Startansicht, falls der Nutzer diese nochmals betrachten möchte. Die Benutzung von Tags (siehe Kapitel 2) wurde als sinnvoll erachtet, da sie ein Kategorisieren von Listen ermöglichen. Auch fiel das standardkonforme Einstellungsmenü auf. Insbesondere Android-erfahrene Nutzer empfanden die Bedienung als gewohnt einfach.

8. Zusammenfassung

Mit TabScript wurde eine Applikation entwickelt, die es dem Benutzer ermöglicht, verschiedene Aufgaben und Notizen handschriftlich einzugeben und in Listen zu verwalten. Benutzereingaben werden dabei mit Hilfe von Hidden-Markov-Modellen klassifiziert und in Maschinenschrift umgewandelt. Die Verwendung von Handschrift soll eine intuitive Alternative zur üblichen Eingabe mittels Soft-Keyboard darstellen.

TabScript befähigt den Benutzer, Aufgabenlisten anzulegen, zu löschen oder anzuordnen. Diesen Aufgabenlisten können Aufgaben hinzugefügt werden, oder aus ihnen entfernt werden. Aufgaben können wahlweise als erledigt oder unerledigt markiert werden. Des Weiteren ist es möglich, Listen mit Tags zu versehen, und sie auf diese Weise zu sortieren. Die Applikation wurde für die Android-Plattform ab Version 3.0 entwickelt. Sie unterstützt sowohl Portrait- als auch Landscapemodus.

Wird eine handschriftliche Eingabe durch den Benutzer getätigt, startet die Erkennung automatisch. Nach Abschluss der Erkennung werden dem Benutzer die drei wahrscheinlichsten Wörter angezeigt. Das Erkennungssystem wurde mit Hilfe des *ESMERALDA*-Toolkits[1] realisiert. Es wird ein Schriftmodell auf Buchstabenebene verwendet, mit dem ganze Wörter durch zusätzliche Anwendung von Wörterbüchern erkannt werden können (vgl. Kapitel 6). Für das Training und die Schätzung der Modellparameter wurde eine Teilmenge der annotierten Trajektorien der Datenbank *Unipen*[2] verwendet. Der Anwendungskontext der Applikation kann durch verschiedene, jeweils ca. 2500 Wörtern umfassende Wörterbücher, festgelegt werden. Es stehen sowohl zwei Wörterbücher mit häufigen Begriffen in deutscher und englischer Sprache zur Auswahl, als auch ein deutsches Wörterbuch zum Thema Lebensmittel. Mit dem trainierten Modell konnte auf Testdaten eine Erkennungsrate von 79% erreicht werden, im Zuge einer repräsentativen Benutzerstudie waren 68% der geschriebenen Wörter unter den ersten drei Erkennungsergebnissen. Der Nutzerstudie lies sich zudem ein positives Feedback über die Hauptmerkmale von TabScript, wie die Integration der Handschrift und das Verwalten von Listen, entnehmen. Damit wurde das Hauptziel der Projektgruppe, Handschrifteingaben als Alternative zu einem Soft-Keyboard zu verwenden, erreicht.

[1] `http://sourceforge.net/projects/esmeralda/`, zuletzt aufgerufen am 23.10.2013
[2] `http://unipen.nici.com.nl/`, zuletzt aufgerufen am 31.10.2013

Literaturverzeichnis

[ADP] *Android Design Principles*. http://developer.android.com/design/get-started/principles.html, zuletzt abgerufen am 25.10.2013,

[Anda] ANDROID: *Layouts*. http://developer.android.com/guide/topics/ui/declaring-layout.html, zuletzt abgerufen am 31.10.2013,

[Andb] ANDROID: *Supporting Multiple Screens*. http://developer.android.com/guide/practices/screens_support.html#qualifiers, zuletzt abgerufen am 31.10.2013,

[AzH09] AIDA-ZADE, Kamil R. ; HASANOV, Jamaladdin Z.: Word base line detection in handwritten text recognition systems. 3 (2009), Nr. 4, S. 745 – 750. – ISSN 1307–6892

[Bau] BAUER, Carl: *DragSortListView*. https://github.com/bauerca/drag-sort-listview, zuletzt abgerufen am 31.10.2013, . – Android-Bibliothek

[Dud] *Duden*. http://www.duden.de/rechtschreibung/Merkmal, zuletzt abgerufen am 15.12.2013,

[Eul06] EULER, S.: *Grundkurs Spracherkennung: Vom Sprachsignal Zum Dialog - Grundlagen und Anwendungen Verstehen - Mit Praktischen Übungen*. Vieweg Verlag, Friedr, & Sohn Verlagsgesellschaft mbH, 2006

[Fei] FEINSTEIN, Jeremy: *SlidingMenu*. https://github.com/jfeinstein10/SlidingMenu, zuletzt abgerufen am 31.10.2013, . – Android-Bibliothek

[Fin08] FINK, Gernot A.: *Markov Models for Pattern Recognition: From Theory to Applications*. Springer London, Limited, 2008 (SpringerLink: Springer e-Books)

[GHJV95] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns: elements of reusable object-oriented software*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 1995. – ISBN 0–201–63361–2

[HYW98] HUERST, Wolfgang ; YANG, Jie ; WAIBEL, Alex: Interactive error repair for an online handwriting interface. In: *CHI 98 Conference Summary on Human Factors in Computing Systems*, 1998, S. 353–354

[JMRW01] JÄGER, Stefan ; MANKE, Stefan ; REICHERT, Jürgen ; WAIBEL, Alex: Online handwriting recognition: the NPen++ recognizer. In: *International Journal on Document Analysis and Recognition* 3 (2001), Nr. 3, S. 169–180

[JMW00] JÄGER, Stefan ; MANKE, Stefan ; WAIBEL, Alex: Npen++: An On-Line Handwriting Recognition System. In: *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, 2000, S. 249–260

[LB05a] LIWICKI, Marcus ; BUNKE, Horst: Handwriting Recognition of Whiteboard Notes. In: *Proceedings of the 12th Conference of the International Graphonomics Society*, 2005, S. 118–122

[LB05b] LIWICKI, Marcus ; BUNKE, Horst: IAM-OnDB - an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard. In: *Proceedings of the 8th International Conference on Document Analysis and Recognition*, 2005, S. 956–961

[Man98] MANKE, Stefan: *On-line Erkennung kursiver Handschrift bei großen Vokabularen*, Karlsruhe Institute of Technology, Diss., 1998

[MBS99] MILETZKI, Udo ; BAYER, Thomas ; SCHÄFER, Hartmut: Continuous Learning Systems: Postal Address Readers with Built-In Learning Capability. In: *Proceedings of the 5th Conference on Document Analysis and Recognition*, 1999, S. 329–332

[McQ67] MCQUEEN, J.: Some Methods for Classification and Analysis of Multivariate Observations. In: *Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), S. 281–296

[MS02] MACKENZIE, Scott ; SOUKOREFF, R. W.: Text Entry for Mobile Computing: Models and Methods, Theory and Practice. In: *Human Computer Interaction* 17 (2002), S. 147–198

[MZ97] MACKENZIE, Scott ; ZHANG, Shawn X.: The immediate usability of Graffit. In: *Proceedings of Graphics Interface*, 1997, S. 129–137

[Nie83] NIEMANN, Heinrich: *Klassifikation von Mustern*. Springer-Verlag, 1983

[Ots79] OTSU, Nobuyuki: A Threshold Selection Method from Gray-Level Histograms. In: *IEEE Transactions on Systems, Man and Cybernetics* 9 (1979), Nr. 1, S. 62–66

[PF09] PLÖTZ, Thomas ; FINK, Gernot A.: Markov models for offline handwriting recognition: a survey. In: *International Journal on Document Analysis and Recognition (IJDAR)* 12 (2009), S. 269–298

[Pri12] PRITZEL, Monika: Händigkeit. In: *Kognitive Neurowissenschaften*. Springer Berlin Heidelberg, 2012 (Springer-Lehrbuch), S. 705–710

[Rah09] RAHMANN, Sven: *Einführung in die Angewandte Bioinformatik: Proteinsequenz-Datenbanken*. `http://ls11-www.cs.uni-dortmund.de/people/rahmann/teaching/ss2009/angebio/ProteinsequenzDB.pdf`, zuletzt abgerufen am 23.10.2013, 2009

[Rah13] RAHMANN, Sven: *Algorithmen der Bioinformatik*. `https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxyYWhtYW5ubGFifGd4OjMzMGY2MDEwNmUwODk1YWQ`, zuletzt abgerufen am 31.10.2013, 2013

[Sca] SCACCO, Renzo: *Palm Handheld Computers: A Case Study In Innovation*

[Sch04] SCHAMBACH, Marc-Peter: *Automatische Modellierung gebundener Handschrift in einem HMM-basierten Erkennungssystem*, Fakultät für Informatik der Universität Ulm, Diss., 2004

[SLTWO96] SIMNER, Marvin L. (Hrsg.) ; LEEDHAM, C. G. C. G. (Hrsg.) ; THOMASSEN, A. J. W. M. (Hrsg.) ; WESTERN ONTARIO), International Graphonomics Society. C. o. (Hrsg.): *Handwriting and drawing research : basic and applied issues*. Amsterdam, Washington, DC : IOS Press Tokyo, Japan, 1996

[ST95] SCHUKAT-TALAMAZZINE, Ernst G.: *Automatische Spracherkennung. Statistische Verfahren der Musteranalyse*. Wiesbaden : Vieweg Verlag, 1995

[SWV06] SCHEIDAT, Tobias ; WOLF, Franziska ; VIELHAUER, Claus: Analyse biometrischer Handschriftverifikation im Kontext von Metadaten. In: *Sicherheit* Bd. 77, GI, 2006 (LNI), S. 54–65

[Uni] *Webseite der International Unipen Foundation*. `http://www.unipen.org/home.html`, zuletzt abgerufen am 15.12.2013,

[Wie03] WIENECKE, Markus: *Videobasierte Handschrifterkennung*, Bielefeld University, Diss., 2003

A. Handbuch

PG568: TabScript – Benutzerhandbuch

Autoren: Sulejman Begovic, Rebecca Doherty
Shinazi Faruki, Daria Filatova
Nina Hesse, Dennis Kesper
Julian Kürby, Niclas Raabe
Johann Straßburg, Christian Wieprecht

Inhaltsverzeichnis

1 Übersicht

Die Notiz-Applikation TabScript kann Aufgabenlisten mit jeweils mehreren Aufgaben verwalten. Zu diesem Zweck ist es dem Benutzer möglich, mit Hilfe der graphischen Oberfläche eine neue Aufgabenliste, auch *Taskliste* genannt, zu erstellen. Der Name dieser Taskliste kann dabei handschriftlich in einer speziellen Schreib-Ansicht eingegeben werden. Diese Eingabe wird schließlich erkannt und in Maschinenschrift abgespeichert.

Das vorliegende Benutzerhandbuch gibt zunächst einen Überblick über die Anwendung TabScript und stellt anschließend die graphische Oberfläche sowie alle Funktionalitäten im Detail vor. Abbildung 1.1 zeigt die Startansicht der Anwendung.

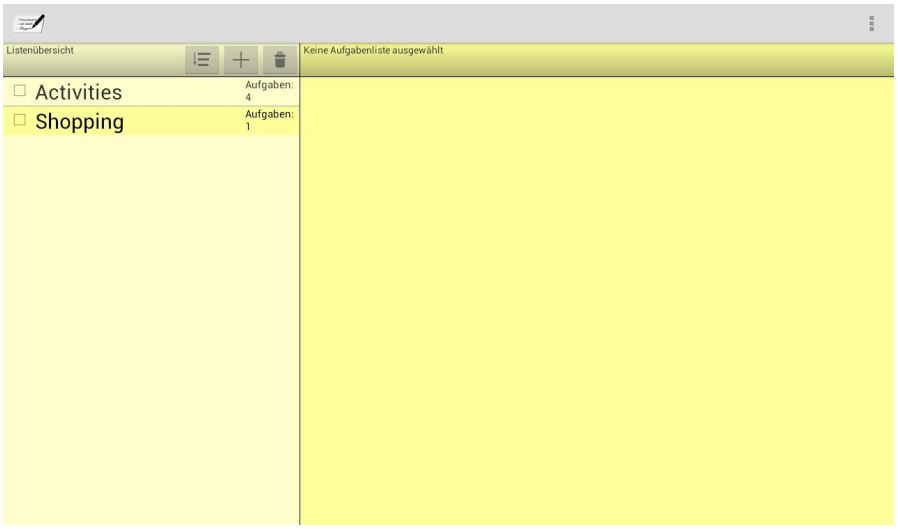


Abbildung 1.1: Startansicht der Android-Anwendung TabScript.

Eine weitere zentrale Funktion der Anwendung ist die Verwendung sogenannter *Tags* für Aufgabenlisten. Tags sind bestimmte Auszeichner, die einer Aufgabenliste zugewiesen werden können und weitere Informationen enthalten. Derselbe Tag lässt sich mehreren Listen zuordnen. Es besteht dann die Möglichkeit, nach einem bestimmten Tag zu filtern und sich damit nur genau diejenigen Aufgabenliste anzeigen zu lassen, welche mit dem entsprechenden Tag versehen sind.

Der Aufbau des Handbuches ist gemäß den drei zentralen Funktionen der Anwendung TabScript gestaltet: Verwaltung von Listen, Aufgaben und Tags. Zunächst wird das Anlegen und die Verwaltung von Aufgabenlisten thematisiert (siehe Kapitel 2). Erstellte Aufgabenlisten lassen sich anschließend mit Aufgaben befüllen, deren Verwaltung in Kapitel 3 erläutert wird. Außerdem können Aufgabenlisten durch bestimmte Tags ausgezeichnet werden. Eine genaue Beschreibung des Tag-Managements findet sich in Kapitel 4. Abschließend wird das Einstellungs-Menü der Applikation erklärt (siehe Kapitel 5).

Zur Visualisierung der verschiedenen Funktionalitäten der Applikation TabScript werden im Folgenden häufig Screenshots verwendet, auf denen Buttons mit roten Nummern, Bildschirmbereiche mit roten Umrandungen sowie weitere im Beschreibungs-Fokus stehende Oberflächen-Elemente mit roten Ellipsen gekennzeichnet sind.

2 Verwaltung von Listen

Die zentrale Aufgabe der Anwendung TabScript besteht in der Verwaltung von verschiedenen Aufgabenlisten, die aus einer Reihe von Aufgaben bestehen können. Dieses Kapitel gibt einen Überblick über alle Funktionen, die im Zusammenhang mit dem Listen-Management stehen.

2.1 Die Listenübersicht

Abbildung 2.1 zeigt die Ansicht, die beim Start der Anwendung erscheint. Die (hier noch leere) Listenübersicht ist dabei rot markiert. Im Wesentlichen besteht sie aus vier Menü-Buttons und einer Anzeige auf der linken Bildschirmhälfte, die die Namen aller bereits erstellten Listen zeigt. Da beim erstmaligen Start der Anwendung noch keine Liste erstellt wurde, ist sie zunächst leer.

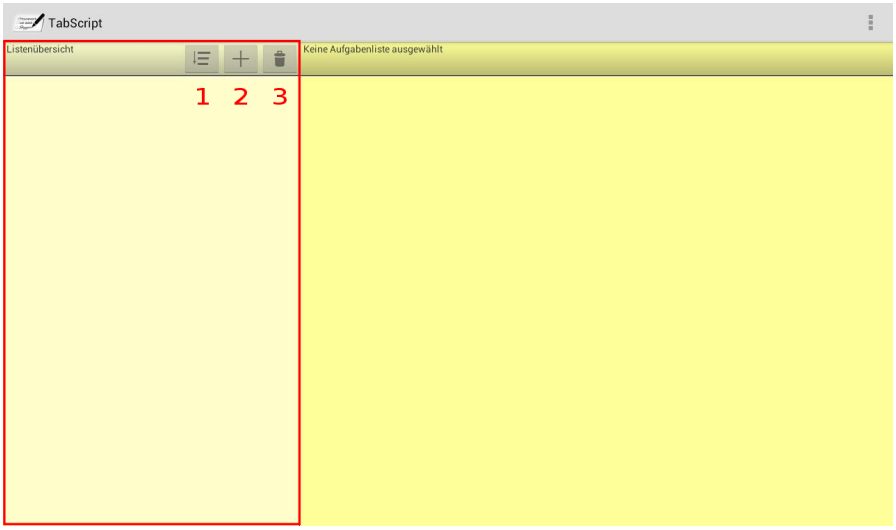


Abbildung 2.1: Die Listenübersicht der Anwendung TabScript.

Die Menü-Buttons dieser Ansicht sind hier in rot nummeriert, um sie im Text besser referenzieren zu können. Der in der Abbildung rot markierte Button 1 dient zur Sortierung der Aufgabenlisten. Mittels Button 2 kann das Menü zum

Erstellen einer neuen Aufgabenliste aufgerufen werden (siehe Abschnitt 2.2). Button 3 dient zum Aufrufen des Löschmodus für eine oder mehrere Listen.

Abbildung 2.2: Schreib-Ansicht für Listen.

2.2 Hinzufügen neuer Listen

In diesem Abschnitt geht es um das Hinzufügen einer neuen Liste zur Listenübersicht. Dazu öffnet sich nach Betätigung des Menü-Buttons 2 aus Abbildung 2.1 die Schreib-Ansicht der Applikation (siehe Abbildung 2.2). Eine genaue Beschreibung dieser Ansicht findet sich in Abschnitt 2.3. Dort kann der gewünschte Listenname nun per Handschrift eingegeben und anschließend mit dem Menüeintrag „Fertig“ bestätigt werden. Neben dem Listennamen erscheint die Anzahl der Aufgaben, die in einer Liste enthalten sind (siehe Abbildung 2.3).

2.3 Die Schreib-Ansicht

Die Schreib-Ansicht ist in Abbildung 2.2 im Kontext der Listenerzeugung dargestellt. Die gleiche Ansicht wird von der Applikation aber auch bei der Erstellung neuer Aufgaben (vergleiche Kapitel 3) oder Tags (vergleiche Kapitel 4) verwendet. Die Linien in der Ansicht sind Hilfslinien, die dazu dienen, die handschriftliche Eingabe zu erleichtern. Die Eingabe muss nicht zwingen exakt an den Hilfslinien ausgerichtet sein.

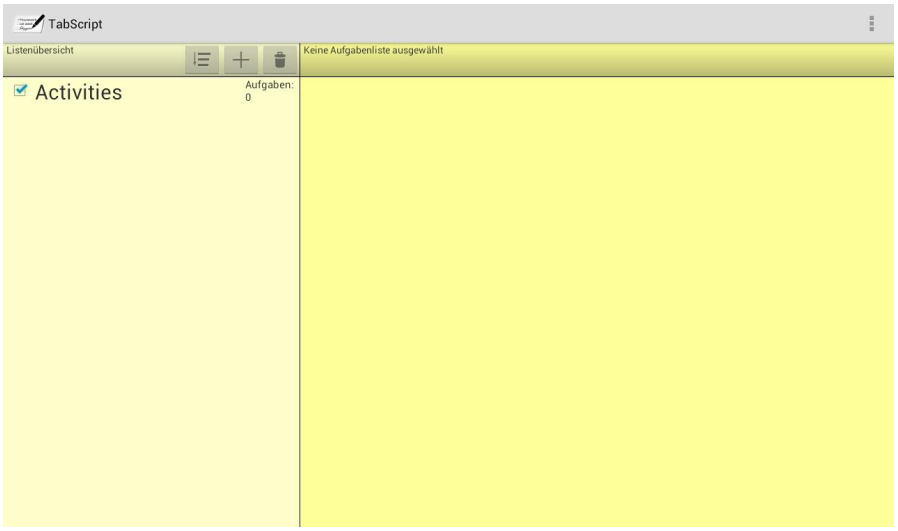


Abbildung 2.3: Die Anzahl der enthaltenden Aufgaben einer Liste befindet sich in der Übersicht rechts neben ihrem Namen.

2.3.1 Eingabe mit Hilfe der Schreib-Ansicht

Die Text-Eingabe kann im gesamten gelben Eingabefenster durchgeführt werden. Sollen mehrere Begriffe auf einmal geschrieben werden, so erfolgt die Eingabe wortweise. Sobald das erste Wort geschrieben wurde, erscheint direkt unter der oberen Menüleiste ein Auswahlmenü, was sich durch einen einfachen Klick ausklappen lässt (siehe Abbildung 2.4). Hier werden die drei plausibelsten Erkennungsergebnisse angezeigt. Diese können durch Klicken ausgewählt werden.

2.3.2 Korrektur und Löschen von Wörtern

Geschriebene Wörter können editiert werden, indem auf das Auswahlmenü des entsprechenden Wortes geklickt und dort der Eintrag „Editieren“ ausgewählt wird (siehe Abbildung 2.5). Wie in Abbildung 2.6 zu sehen ist, wird das entsprechende Wort dann im Auswahlmenü rot hinterlegt. Nun kann dieses Wort im gelben Eingabefenster neu geschrieben werden.

Möchte der Nutzer ein Wort komplett löschen, so kann er ebenfalls das Auswahlmenü dieses Wortes aufrufen und dort mittels des Eintrages „Löschen“ (siehe Abbildung 2.7) dafür sorgen, dass das Wort aus der Leiste mit den erkannten Wörtern entfernt wird.

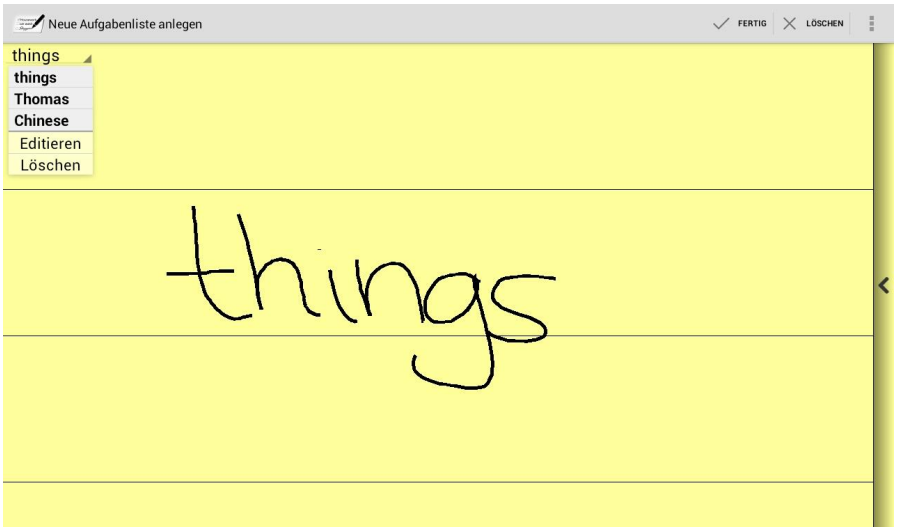


Abbildung 2.4: Auswahlmenü für weitere Wortvorschläge bei der Eingabe eines neuen Wortes.

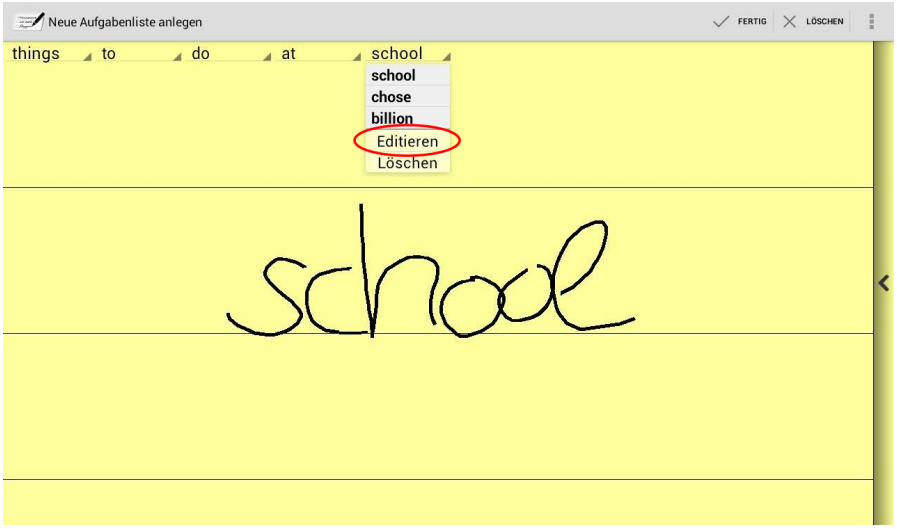


Abbildung 2.5: Editieren eines Wortes in der Schreib-Ansicht.

2.3.3 Zurücksetzen der gesamten Schreib-Ansicht

Als weitere Option stellt TabScript dem Benutzer eine Möglichkeit zur Verfügung, alle bereits erkannten Wörter und Eingaben zu löschen. In Abbildung 2.8 wird demonstriert, wie dies mittels des Menü-Eintrages „Löschen“ in der oberen Menüleiste durchgeführt werden kann.

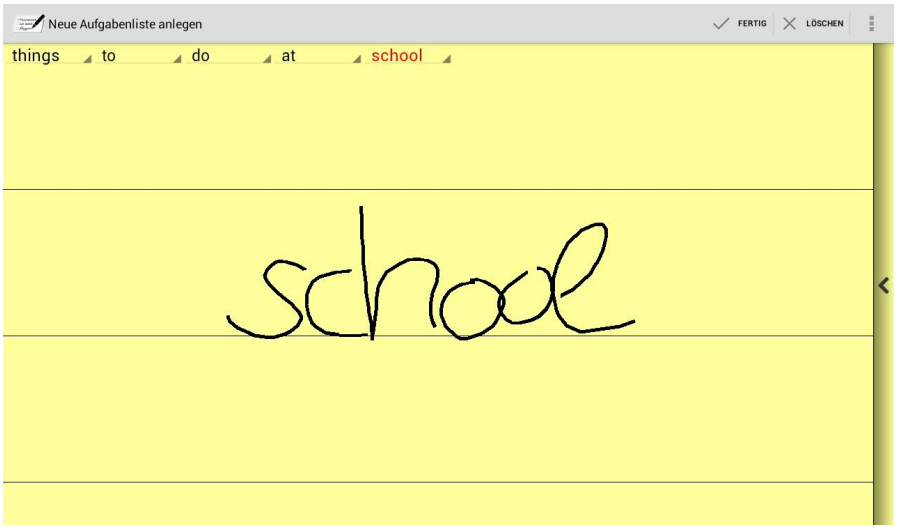


Abbildung 2.6: Das zu editierende Wort wird rot hinterlegt.

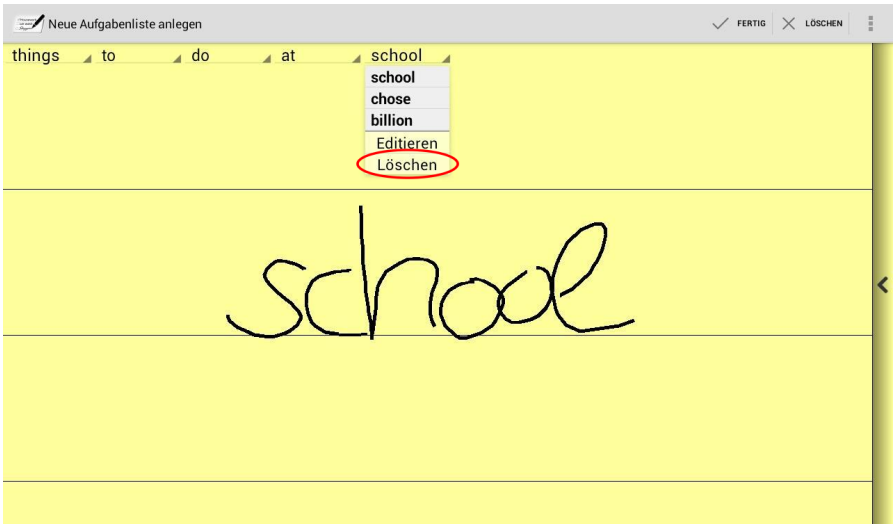


Abbildung 2.7: Löschen eines einzelnen Wortes in der Schreib-Ansicht.

2.3.4 Ausklappbare Seitenansicht

Um das Anlegen von neuen Listen, Aufgaben oder Tags zu vereinfachen, lässt sich über den Pfeil am rechten Bildschirmrand der Schreib-Ansicht eine Seitenansicht ausklappen, die je nach Kontext entweder alle bereits erstellten Listen, Aufgaben einer Liste oder Tags anzeigt (siehe Abbildung 2.9).
Zunächst wird auf den Pfeil geklickt und dieser Klick gehalten. Nun lässt sich die Randleiste mit gehaltenem Klick nach links ziehen, bis sie einrastet (siehe

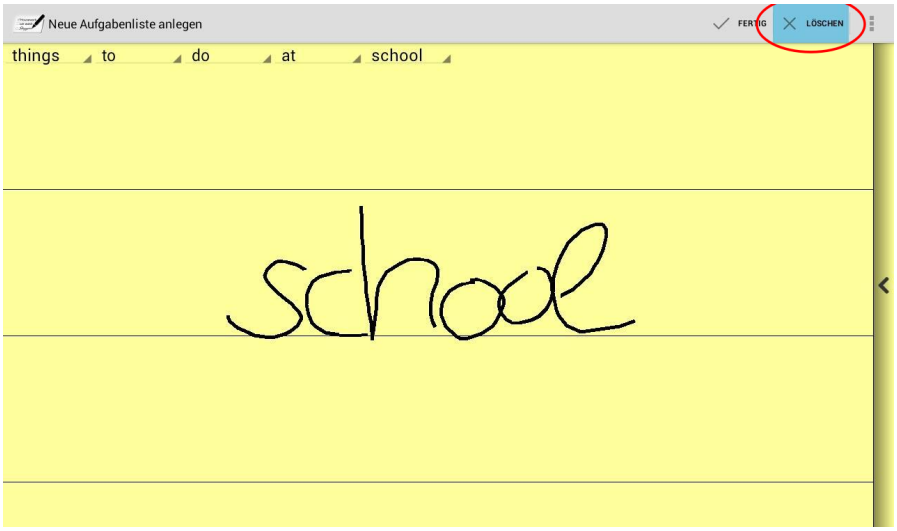


Abbildung 2.8: Zurücksetzen des gesamten Schreib-Ansicht.

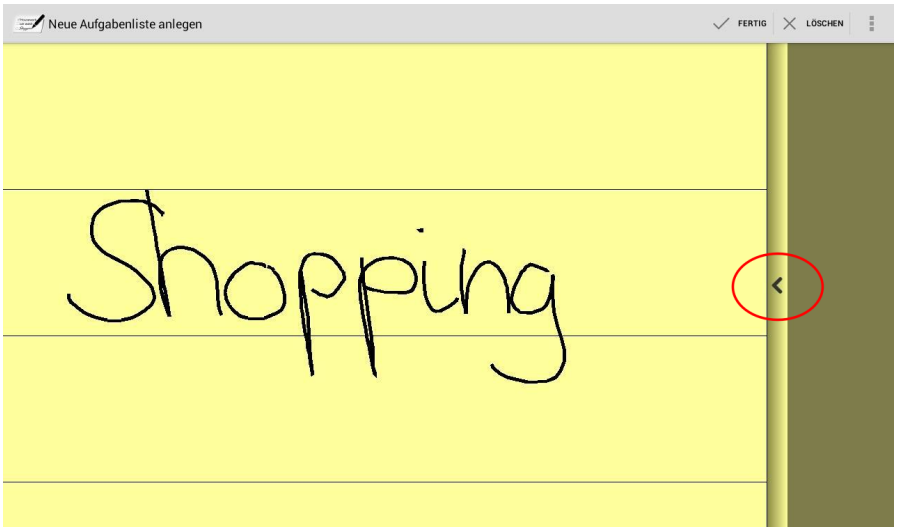


Abbildung 2.9: Über den Pfeil am rechten Bildschirmrand lässt sich die Seitenansicht ausfahren.

Abbildung 2.10). Um diese Ansicht wieder einzuklappen wird der Vorgang auf die gleiche Weise in die andere Richtung durchgeführt (siehe Abbildung 2.11).

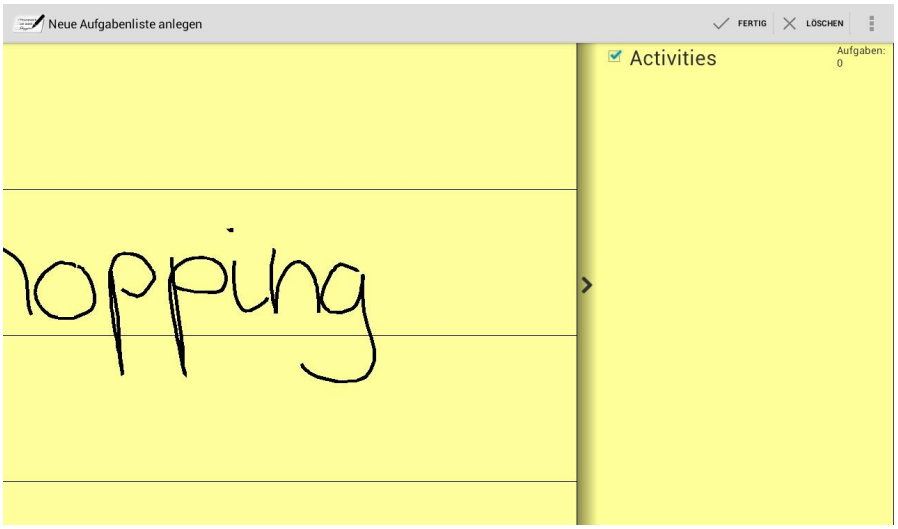


Abbildung 2.10: Die Seitenansicht zeigt je nach Kontext alle bereits erstellten Listen, Aufgaben oder Tags an.

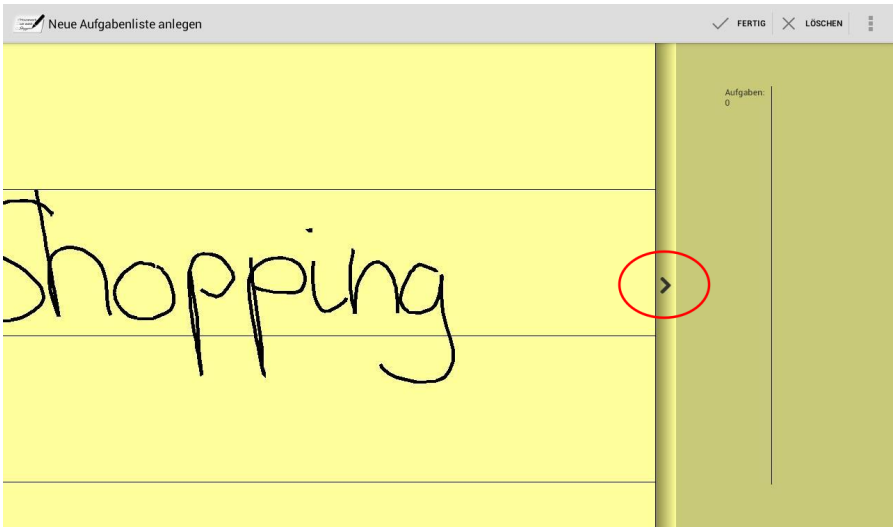


Abbildung 2.11: Um die Seitenansicht wieder einzufahren, wird der nach rechts zeigende Pfeil verwendet.

2.3.5 Hinzufügen von Wörtern zum internen Wörterbuch

Über das Android-Menü lässt sich innerhalb der Schreib-Ansicht der Eintrag „Neues Wort zum Wörterbuch hinzufügen“ auswählen. Ein Klick auf diesen Button öffnet ein Fenster, in dem ein neues Wort unter Verwendung der Software-Tastatur von Android eingegeben werden kann (siehe Abbil-

dung 2.12). Das Wort wird in das interne Wörterbuch von TabScript aufgenommen und durch die Bestätigung mit dem Button „Hinzufügen“ auch in der Wörter-Leiste der Schreib-Ansicht als letzte Eingabe angezeigt. Abbrechen lässt sich der Vorgang durch einen Klick auf den gleichnamigen Button. Mit Hilfe dieser Funktionalität ist es möglich, auch Wörter aufzunehmen, die wiederholt nicht von der Handschrift-Erkennung erkannt wurden.

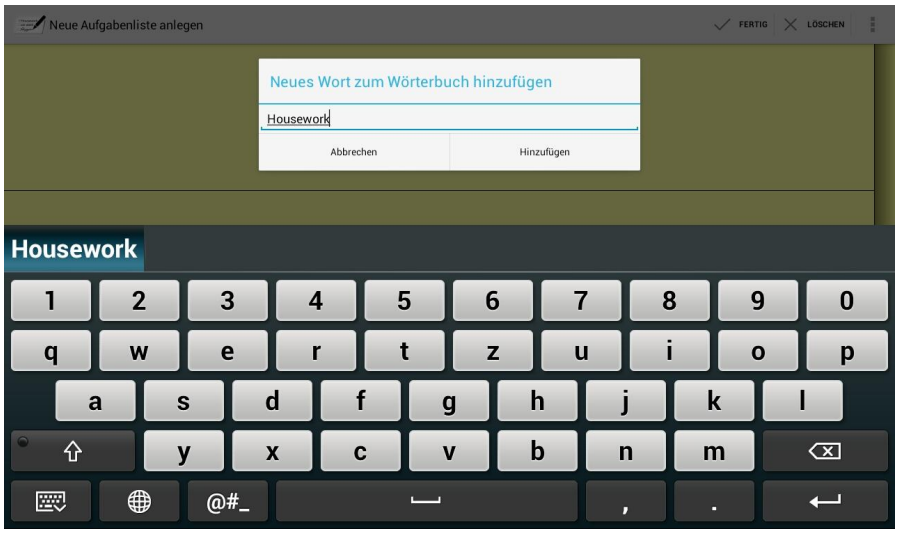


Abbildung 2.12: Hinzufügen von Wörtern zum Wörterbuch der Anwendung über die Software-Tastatur von Android.

2.4 Löschen von Listen

Um eine oder mehrere Aufgabenlisten gleichzeitig zu entfernen muss der Menü-Button 3 angeklickt werden. Auf diese Weise wird der Löschmodus aktiv. Nun lassen sich per Klick auf die jeweiligen Listennamen alle Listen auswählen, die gelöscht werden sollen. Durch das Anklicken werden die Listennamen rot hinterlegt (siehe Abbildung 2.13).

Die Bestätigung des Löschvorganges erfolgt dann über die Auswahl des Eintrages „Löschen“ in der oberen Menüleiste (siehe Abbildung 2.14). Um den Vorgang ohne das Löschen von Listen abzubrechen und den Löschmodus zu verlassen, lässt sich in der Menüleiste der Eintrag „Abbrechen“ wählen.

Es ist zu beachten, dass beim Löschen von ganzen Listen auch alle enthaltenen Aufgaben der ausgewählten Listen entfernt werden. Der Löschmodus wird in dieser Form ebenfalls für das Löschen von Aufgaben und Tags verwendet, was in Kapitel 3 bzw. 4 beschrieben wird.

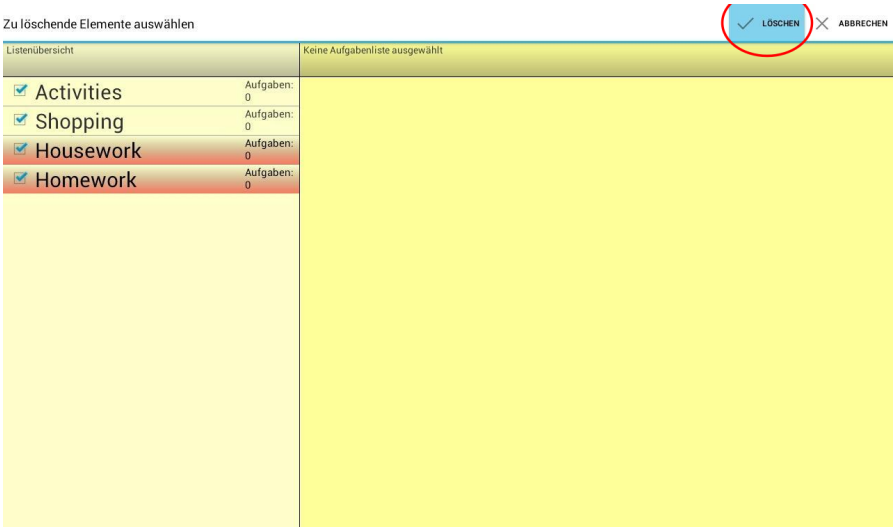


Abbildung 2.13: Auswahl mehrerer Listen im Löschmodus der Anwendung.

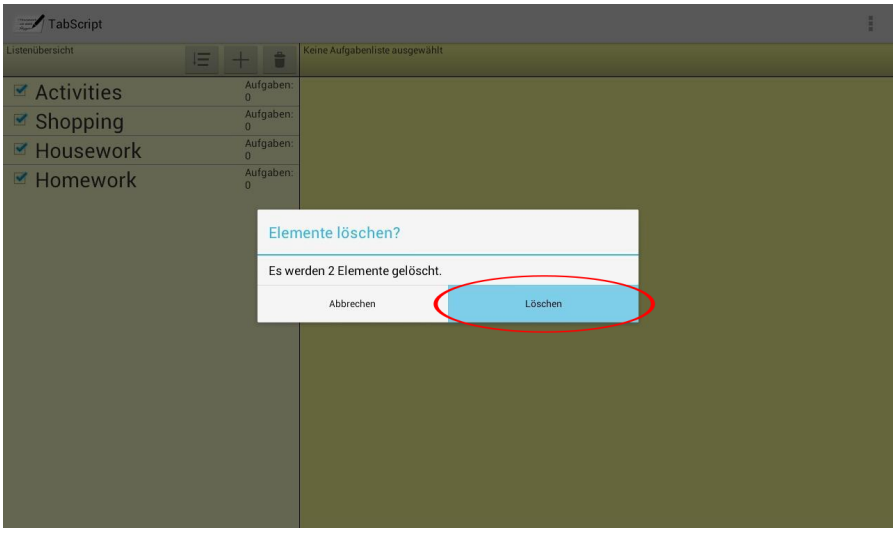


Abbildung 2.14: Bestätigung des Löschvorganges der ausgewählten Listen.

2.5 Sortierung von Listen

TabScript bietet zwei verschiedene Möglichkeiten zur Sortierung von Aufgabenlisten in der Listenansicht, welche in diesem Abschnitt beschrieben werden.

2.5.1 Automatische Sortierung

Bei der automatischen Sortierung kann zwischen einer Sortierung nach „Name“, „Erstellungsdatum“ oder „Anzahl“ gewählt werden. Abbildung 2.15 zeigt, wie sich das Sortiermenü über das Klicken auf den Menü-Button 1 aufrufen lässt. Nun kann der Vorgang durch Klicken auf die gewünschte Sortierreihenfolge ausgeführt werden.

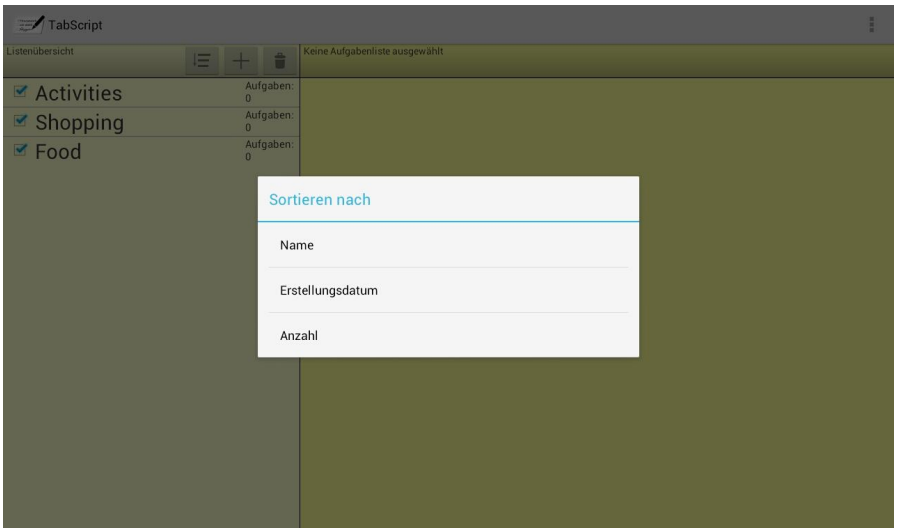


Abbildung 2.15: Sortieren der Listen in der Listenübersicht nach „Name“, „Erstellungsdatum“ oder „Anzahl“.

2.5.2 Manuelle Sortierung

Als Alternative zur automatischen Sortierung steht die manuelle Sortierung zur Verfügung. Dabei lassen sich einzelne Listen an eine andere Stelle in der Listenübersicht schieben. Zu diesem Zweck wird der Listenname der zu verschiebenden Liste angeklickt und der Klick gehalten. Nun lässt sich das Element an eine andere Position in der Liste ziehen (Abbildung 2.16) und rastet dort ein.

Die Möglichkeiten zum Sortieren werden in derselben Form ebenfalls zum Sortieren von Aufgaben und Tags verwendet, was in Kapitel 3 bzw. 4 beschrieben wird.

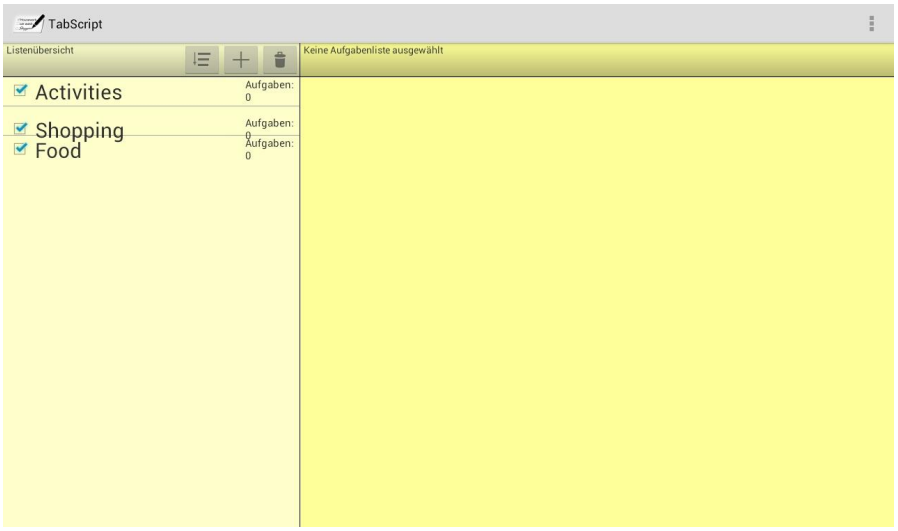


Abbildung 2.16: Manuelle Sortierung in der Listenübersicht.

3 Verwaltung von Aufgaben

Sobald eine Aufgabenliste erstellt wurde, lässt sich diese mit zugehörigen Aufgaben füllen. In diesem Kapitel werden die Funktionalitäten zum Erstellen, Anzeigen, Sortieren und Löschen solcher Listen-Einträge vorgestellt.

3.1 Anzeigen von Aufgaben

In der Startansicht der Anwendung TabScript befindet sich auf der rechten Bildschirmhälfte die Aufgabenübersicht (siehe Abbildung 3.1). Diese Ansicht zeigt in der Titelleiste den Namen der ausgewählten Liste an. Am rechten Rand dieser Leiste befinden sich vier Menü-Buttons. In der Abbildung sind diese in rot nummeriert, um sie im Text besser referenzieren zu können.

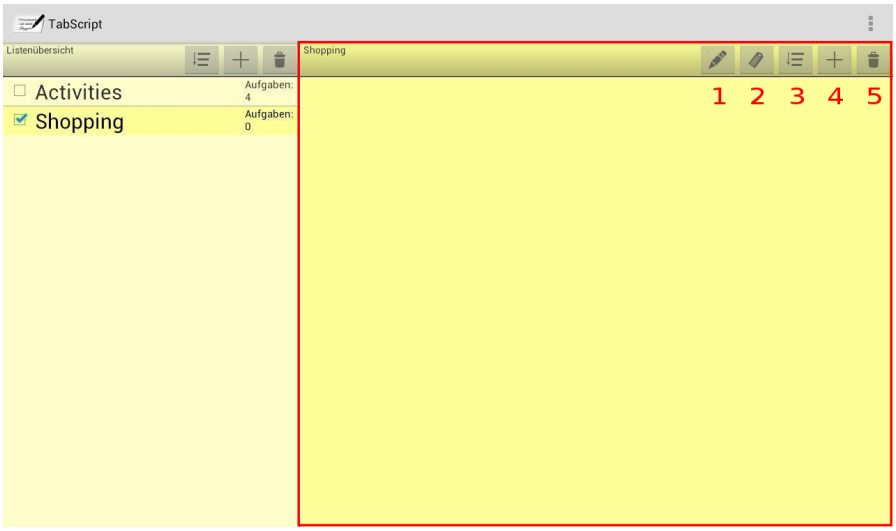


Abbildung 3.1: Die Aufgabenübersicht der Anwendung TabScript.

Mittels Menü-Button 1 kann der Listenname einer ausgewählten Liste editiert werden. Der Button 2 dient zum Zuweisen eines Tags zu einer Aufgabenliste. Diese Funktionalität wird in Kapitel 4 genauer erläutert. Die Sortierung der Aufgaben innerhalb einer Liste kann mittels Button 3 durchgeführt werden.

Über Button 4 lässt sich das Menü zum Erstellen einer neuen Aufgabe aufrufen. Button 5 dient zum Aufrufen des Löschmodus für eine oder mehrere Aufgaben.

Die folgenden Abschnitte erläutern, wie sich die Aufgaben und die Zusatzinformationen einer Aufgabe anzeigen lassen.

3.1.1 Anzeigen von Aufgaben einer Liste

Ein einfacher Klick auf eine Aufgabenliste führt dazu, dass in der Aufgabenübersicht alle Namen der enthaltenen Aufgaben dieser Liste angezeigt werden. Abbildung 3.2 gibt einen Überblick über diese Ansicht. Enthält eine Liste noch keine Aufgaben, so erscheint eine leere Aufgabenübersicht.

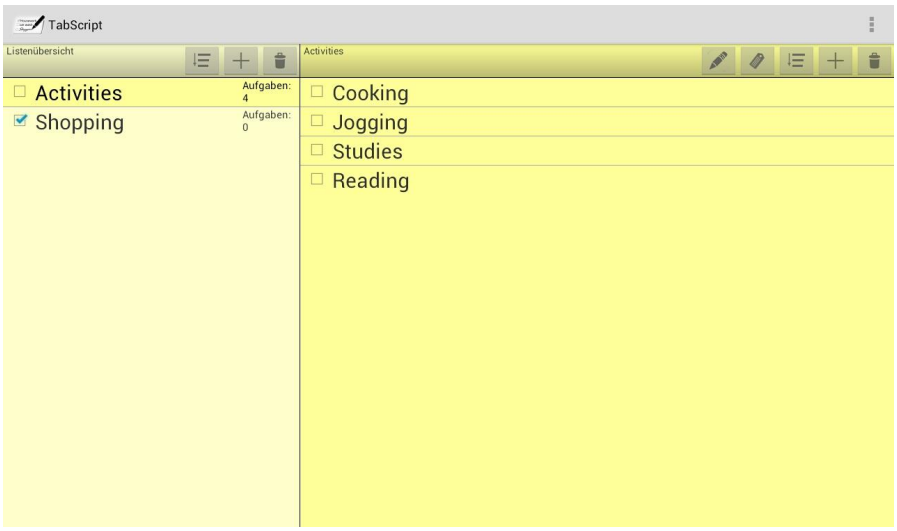


Abbildung 3.2: Anzeigen der Aufgaben, die zu einer Liste gehören in der Aufgabenübersicht.

Links neben dem Namen einer Aufgabe wird ein Kästchen angezeigt, welches zunächst leer ist. Ein solches leeres Kästchen neben dem Aufgabentitel symbolisiert, dass eine Aufgabe noch nicht erledigt wurde. Um diese als „erledigt“ zu markieren, ist ein Klick auf das Kästchen erforderlich. Anschließend erscheint ein Haken in dem Kästchen, wie in Abbildung 3.3 dargestellt ist. Aufheben lässt sich die Markierung durch einen weiteren Klick auf das Kästchen.

3.1.2 Anzeigen der Informationen einer Aufgabe

Um sich zusätzliche Informationen, die pro Aufgabe von der Anwendung gespeichert werden, anzeigen zu lassen, wird der Name der konkreten Aufgabe

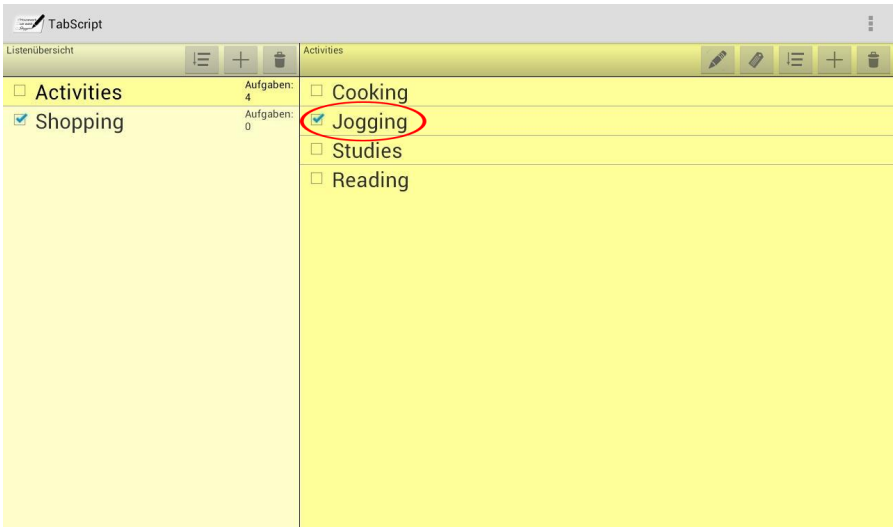


Abbildung 3.3: Markieren einer Aufgabe als „erledigt“.

in der Aufgabenübersicht angeklickt. Abbildung 3.4 gibt einen Überblick über das Übersichts-Fenster mit den Informationen für eine Beispiel-Aufgabe, welches dann angezeigt wird.

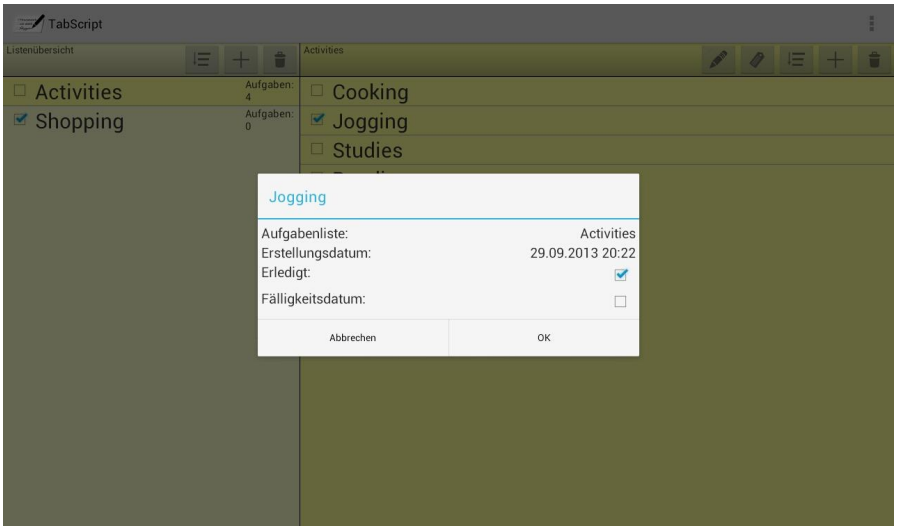


Abbildung 3.4: Zusatzinformationen einer Aufgabe anzeigen lassen.

Wie Abbildung 3.4 visualisiert, wird in diesem Übersichts-Fenster für jede Aufgabe die zugehörige Aufgabenliste und das Erstellungsdatum der Aufgabe gespeichert. Weiterhin wird der Status einer Aufgabe („erledigt“ oder „nicht erledigt“) sowie ein eventuell ausgewähltes Fälligkeitsdatum angezeigt.

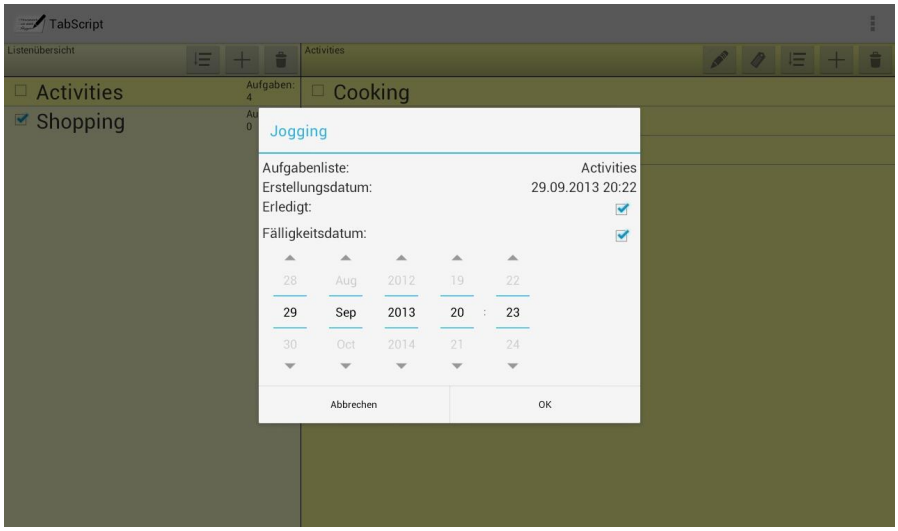


Abbildung 3.5: Einstellen eines Fälligkeitsdatums für eine Aufgabe.

In dieser Übersicht kann der Status einer Aufgabe durch Klick auf das Kästchen des Eintrages „Erledigt“, wie zuvor beschrieben, verändert werden. Ist ein Fälligkeitsdatum für eine Aufgabe gewünscht, so lässt sich das Kästchen neben dem gleichnamigen Eintrag durch einen Klick anwählen. Es erscheint die Datums- und Uhrzeit-Auswahl aus Abbildung 3.5. Der gewünschte Termin für die Aufgabe kann hier eingestellt werden. Entfernt wird das Fälligkeitsdatum ebenfalls wieder mit einem Klick auf das zugehörige Kästchen.

Um die Informations-Ansicht einer Aufgabe zu verlassen und Änderungen zu speichern, wird der Vorgang mit einem Klick auf den Button „OK“ bestätigt. Sollen die Änderungen verworfen werden, so wird dies mit einem Klick auf den Button „Abbrechen“ erreicht.

3.2 Umbenennen von Listen

Das Ändern des Namens einer bereits erstellten Liste ist ebenfalls über die Menü-Buttons der Listenübersicht möglich. Hierzu muss die entsprechende Liste zunächst ausgewählt werden und anschließend der Menü-Button 1 aus Abbildung 3.7 betätigt werden. In der Schreib-Ansicht, die sich daraufhin öffnet, kann nun der gewünschte neue Listenname eingegeben werden. Abschnitt 2.3 beschreibt den Prozess der Handschrifteingabe.

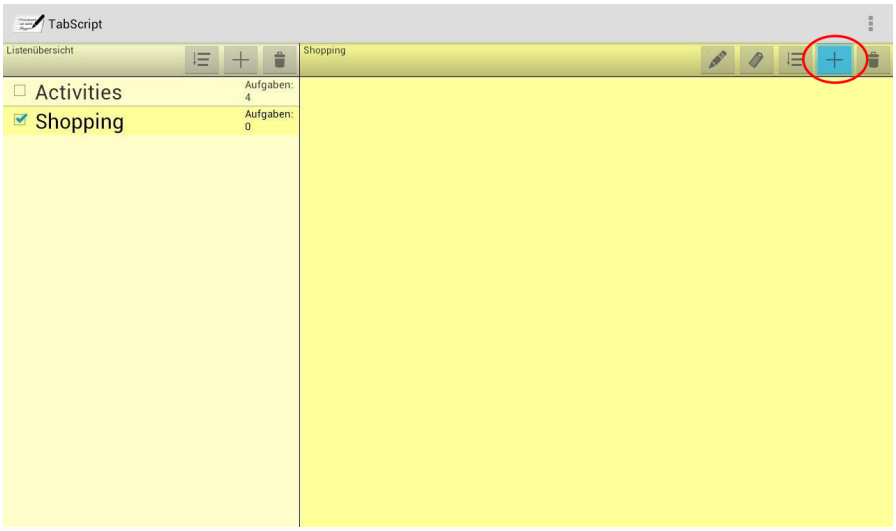


Abbildung 3.6: Menü-Button zum Hinzufügen einer neuen Aufgabe zu einer Liste.

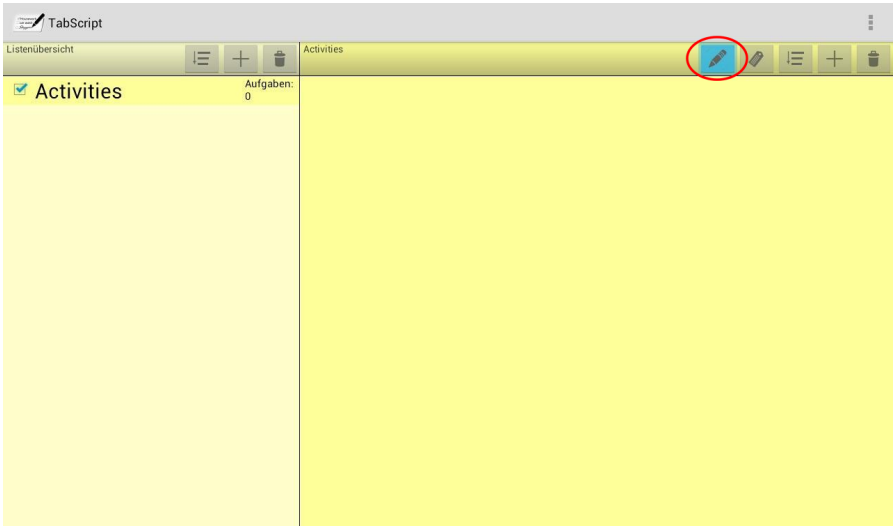


Abbildung 3.7: Umbenennen von Listen.

3.3 Hinzufügen von Aufgaben zu einer Liste

Um eine neue Aufgabe zu einer vorhandenen Liste hinzuzufügen, muss die gewünschte Aufgabenliste zunächst in der Listenansicht ausgewählt werden. Anschließend wird Button 4 aus der Titelleiste der Aufgabenübersicht betätigt (siehe Abbildung 3.6). In der Schreib-Ansicht kann nun der Name der neuen

Aufgabe eingegeben und bestätigt werden. Wie die Handschrifteingabe im Detail durchzuführen ist, wird in Abschnitt 2.3 erklärt. Die neu erzeugte Aufgabe erscheint nun zusammen mit den anderen Aufgaben der Liste in der Aufgabenübersicht (siehe Abbildung 3.8).

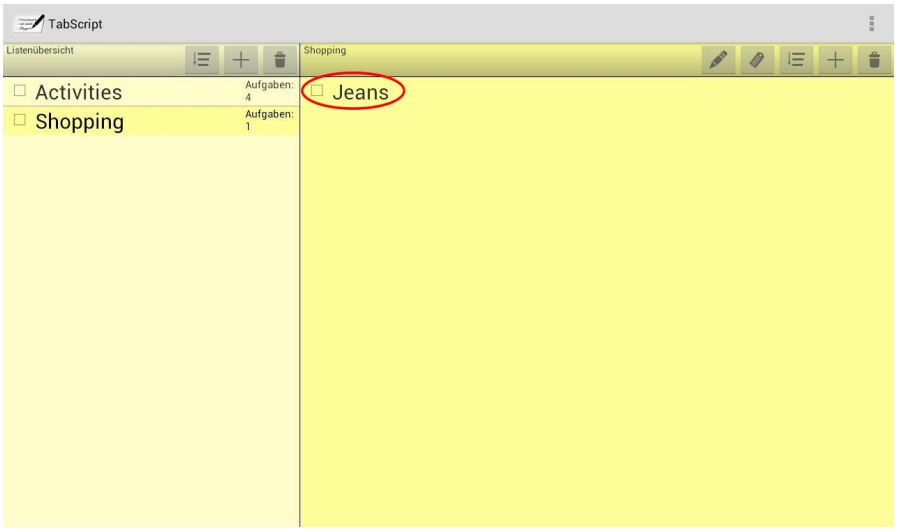


Abbildung 3.8: Eine neue Aufgabe wurde zu einer Liste hinzugefügt und erscheint nun auch in der Aufgabenansicht.

3.4 Sortierung von Aufgaben

Um eine automatische Sortierung aller Aufgaben einer bestimmten Liste vorzunehmen, wird diese zunächst in der Listenübersicht ausgewählt. Im Anschluss muss der Button mit der 3 in der Titelleiste der Aufgabenübersicht betätigt werden. Nun erscheint das Menü aus Abbildung 3.9 und es besteht die Möglichkeit, eine Sortierung nach „Name“, „Erstellungsdatum“, „erledigte Aufgaben“ oder „Fälligkeitsdatum“ durchzuführen. Die Sortierung erfolgt analog zu dem Verfahren, das in Abschnitt 2.5.1 für Aufgabenlisten beschrieben wird. Auch eine manuelle Sortierung ist für Aufgaben möglich. Diese erfolgt ebenfalls analog zu dem Verfahren für Listen aus Abschnitt 2.5.2.

3.5 Löschen von Aufgaben

Das Löschen von einer oder mehreren Aufgaben aus einer Liste funktioniert auf folgende Weise: Zunächst wird die entsprechende Liste in der Listenübersicht

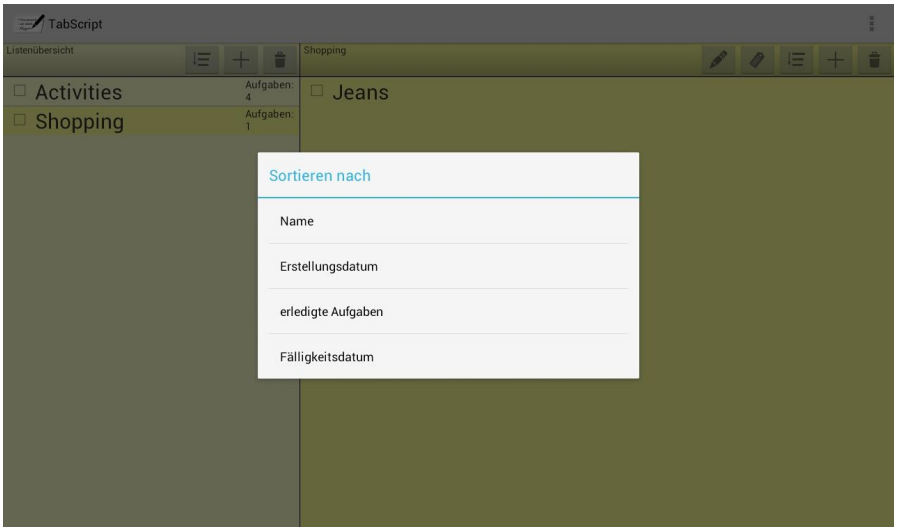


Abbildung 3.9: Sortierung von Aufgaben nach „Name“, „Erstellungsdatum“, „erledigte Aufgaben“ oder „Fälligkeitsdatum“ in der Aufgabenübersicht.

ausgewählt. Nun wird mit Hilfe des Buttons 5 aus der Titelleiste der Aufgabenübersicht der Löschmodus aktiviert. Die Bedienung erfolgt analog zu der Vorgehensweise bei ganzen Listen, die in Abschnitt 2.4 beschrieben wird. Die zu löschenden Aufgaben werden per Klick ausgewählt, anschließend muss der Vorgang mit Klick auf den Button „Löschen“ bestätigt werden (siehe Abbildung 3.10).

3.6 Benachrichtigungen für Aufgaben

In Abschnitt 3.4 wird beschrieben, wie einer Aufgabe ein Fälligkeitsdatum zugewiesen werde kann. Besitzt eine Aufgabe ein solches Ablaufdatum, so erscheint in der Benachrichtigungsleiste von Android das Programm-Icon (siehe Abbildung 3.11) der Anwendung, sobald die Erinnerung für diese Aufgabe aktiv wird. Dies geschieht auch, wenn TabScript zu dem Zeitpunkt gar nicht geöffnet ist. Der Erinnerungszeitpunkt lässt sich in den Einstellungen festlegen. Eine genaue Beschreibung dieser Option findet sich in Abschnitt 5.2. Ein Klick auf das Benachrichtigungs-Symbol öffnet TabScript und dort insbesondere die Informationen der jeweiligen Aufgabe.

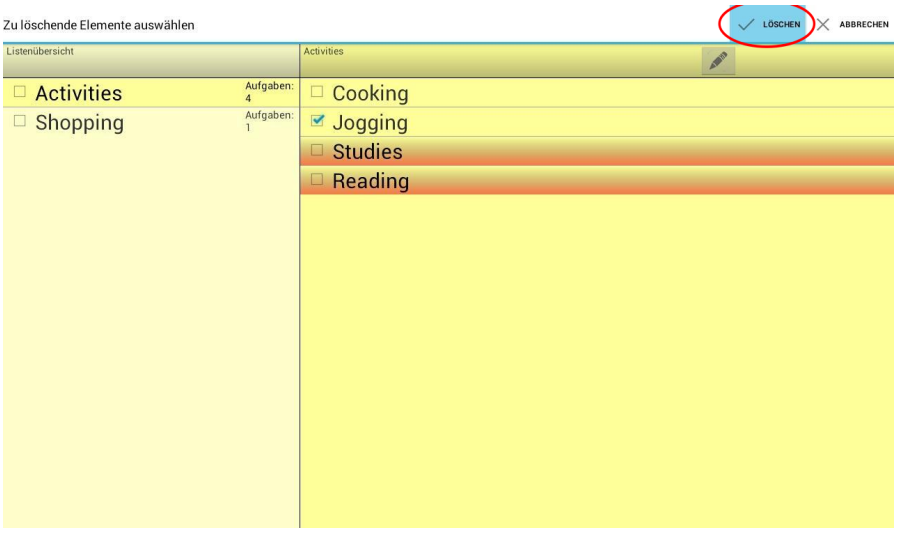


Abbildung 3.10: Löschen von einer oder mehrerer Aufgaben aus der Aufgabenübersicht.

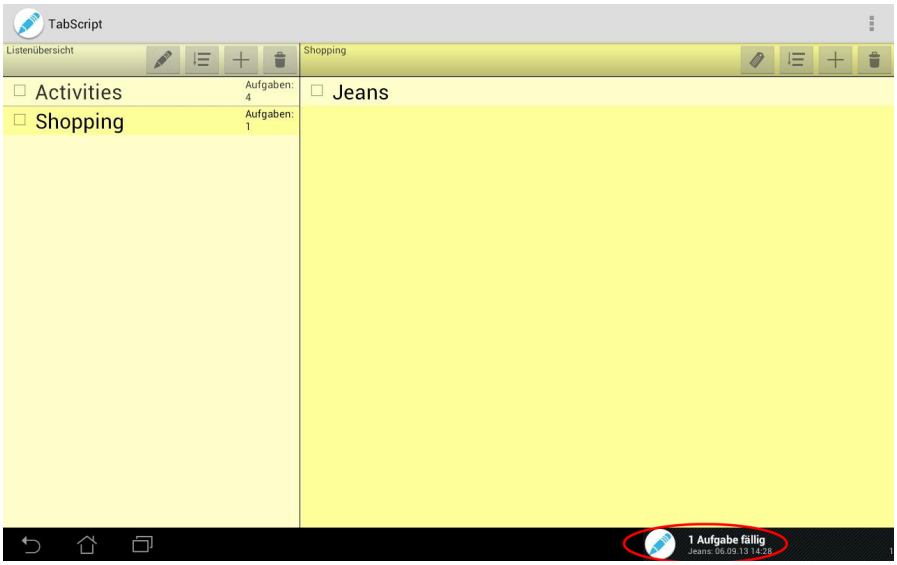


Abbildung 3.11: Symbol für den Fälligkeitszeitpunkt einer Aufgabe in der Benachrichtigungsleiste von Android.

4 Verwaltung von Tags

In TabScript ist es möglich, bestehende Aufgabenlisten auszuzeichnen und sie somit mit bestimmten Zusatzinformationen zu versehen, sogenannten *Tags*. Wie solche Tags erstellt, sortiert oder gelöscht werden, wird in den folgenden Abschnitten erläutert. Zudem wird erklärt, wie die Auszeichnung von einer Liste mit einem Tag erfolgt und die Ansicht nach einem solchen gefiltert werden kann.

4.1 Die Tag-Übersicht

Um in die Tag-Übersicht zu gelangen, wird im Android-Menü der Anwendung der Eintrag „Tags verwalten“ per Klick ausgewählt (siehe Abbildung 4.1). Abbildung 4.2 zeigt diese Übersicht mit den Namen aller bereits erstellten Tags und den drei zugehörigen Menü-Buttons in der Titelleiste. Diese sind hier zur Vereinfachung der Beschreibung mit roten Zahlen nummeriert.

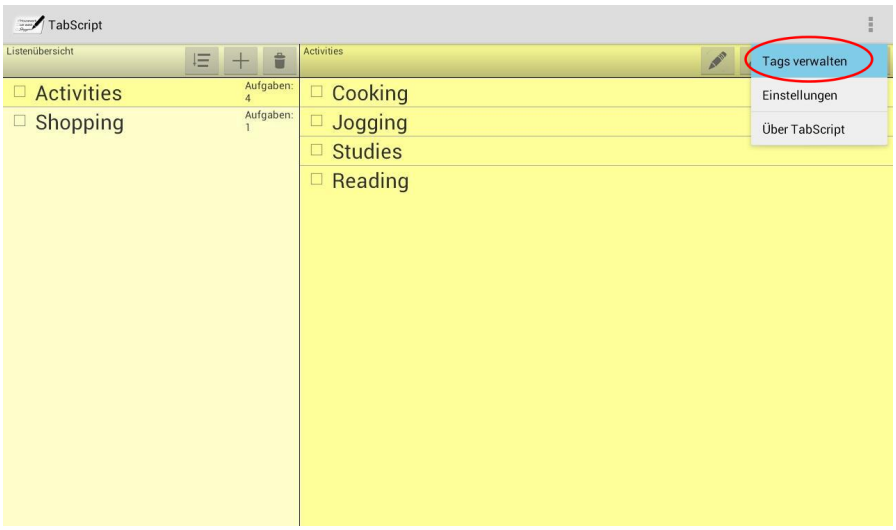


Abbildung 4.1: Aufrufen der Tag-Ansicht in TabScript.

Der Menü-Button 1 ermöglicht das Hinzufügen eines neuen Tags mit Hilfe der Schreib-Ansicht, deren Bedienung in Abschnitt 2.3 beschrieben wird. Button 2

aktiviert den Löschmodus zum Löschen von einem oder mehreren Tags. Mit Hilfe des Buttons 3 können die Tags sortiert werden.

Abbildung 4.2: Die Tag-Ansicht gibt einen Überblick über alle bereits erstellten Tags. Da noch keine Tags erstellt wurden, ist sie zunächst leer.

4.2 Hinzufügen von Tags

Über die Betätigung des Buttons 1 lässt sich in der Tag-Übersicht ein neuer Tag hinzufügen. Über die Auswahl dieses Buttons wird die Schreib-Ansicht geöffnet, welche in Abschnitt 2.3 genauer erläutert wird. In dieser lässt sich nun der Name des neu zu erstellenden Tags eingeben und bestätigen. Anschließend wird der Name dieses Tags zusammen mit den anderen Tags in der Übersicht angezeigt (siehe Abbildung 4.3).

Abbildung 4.3: Tag-Ansicht, nachdem drei Beispiel-Tags hinzugefügt wurden.

4.3 Sortierung von Tags

Um eine Sortierung aller bereits erstellten Tags vorzunehmen, muss in der Tag-Übersicht der Button 3 betätigt werden. Nun erscheint eine Auswahlmöglichkeit, die anbietet, entweder nach „Name“ oder nach „Erstellungsdatum“ zu sortieren (siehe Abbildung 4.4). Durch einen Klick auf den jeweiligen Button kann die Auswahl bestätigt werden. Die Sortierung erfolgt analog zu dem Verfahren, das in Abschnitt 2.5.1 für Aufgabenlisten beschrieben wird. Auch eine manuelle Sortierung ist für Tags möglich. Diese erfolgt ebenfalls analog zu dem Verfahren für Listen aus Abschnitt 2.5.2.

4.4 Löschen von Tags

Zum Löschen von einem oder mehreren Tags wird in der Tag-Übersicht der Button 2 angeklickt und auf diese Weise der bereits aus Abschnitt 2.4 bekannte Löschmodus aktiviert. Die zu löschenden Tags können nun per Klick ausgewählt werden (siehe Abbildung 4.5). Schließlich kann der Löschvorgang mit einem Klick auf den Button „Löschen“ bestätigt werden.

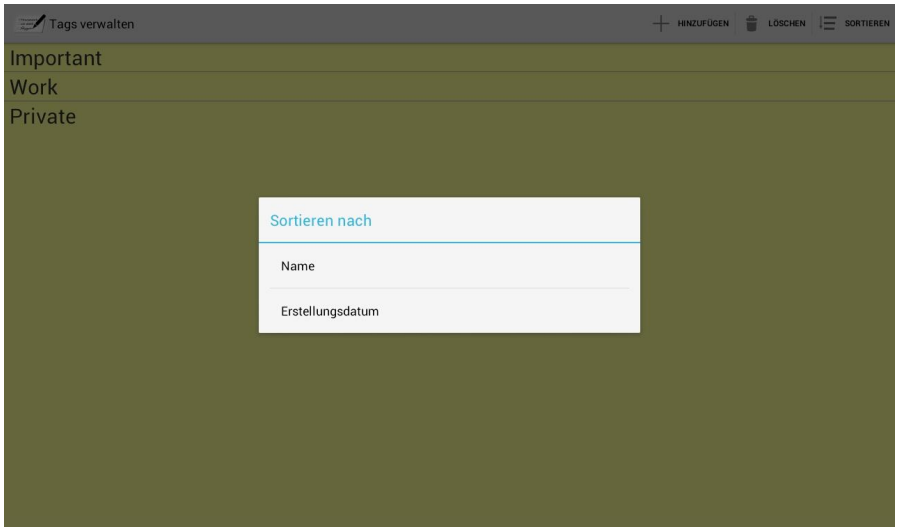


Abbildung 4.4: Sortierung von Tags nach „Name“ oder nach „Erstellungsdatum“ in der Tag-Ansicht.

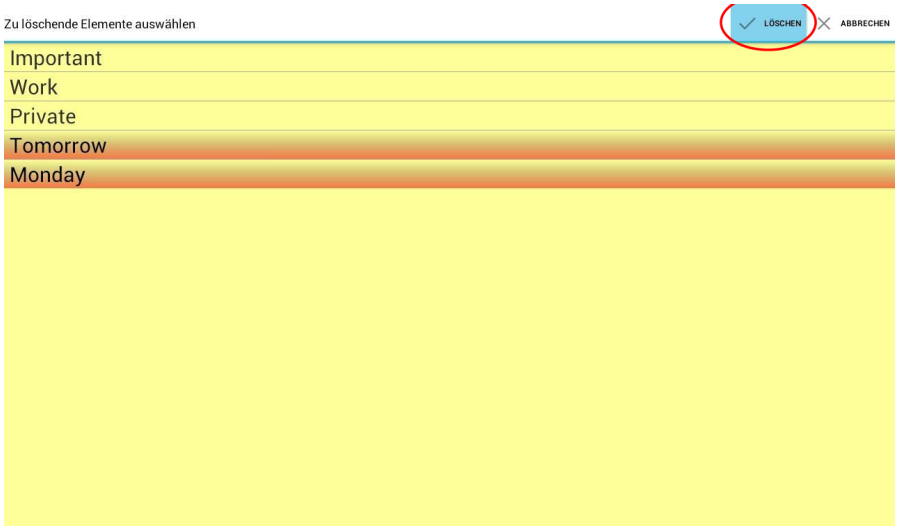


Abbildung 4.5: Löschen von einem oder mehreren Tags mit dem Löschmodus von TabScript.

4.5 Versehen von Listen mit Tags

Um eine Liste mit einem bestimmten Tag zu versehen, muss die Aufgabenliste zunächst in der Listenübersicht durch einen Klick markiert werden. Nun lässt sich der Menü-Button 1 in der Titelleiste der Aufgabenübersicht auswählen und

es öffnet sich eine Dialogbox (siehe Abbildung 4.6). Diese zeigt bereits erstellte Tags an und bietet die Möglichkeit einen oder mehrere Tags durch Klick auf die nebenstehenden Kästchen anzuwählen.

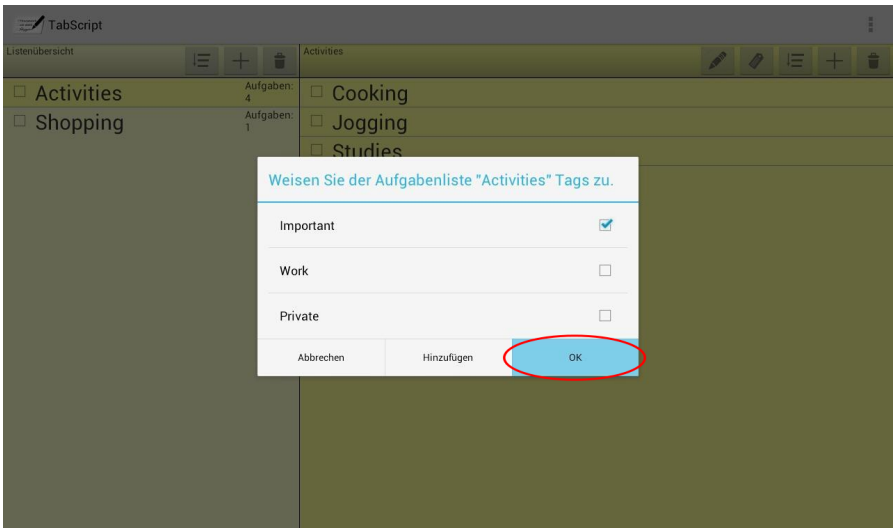


Abbildung 4.6: Versehen einer bestimmten Liste mit einem oder mehreren Tags.

Im unteren Bereich des Dialog-Fensters befinden sich Buttons zum Abbrechen des Vorgangs, zum Hinzufügen neuer Tags und zur Bestätigung der getroffenen Auswahl an Tags.

4.6 Listenübersicht nach einem Tag filtern

Um sich in der Listenübersicht ausschließlich diejenigen Listen anzeigen zu lassen, die mit einem bestimmten Tag versehen sind, muss in der Tag-Übersicht auf den Namen des entsprechenden Tags geklickt werden. Dann erscheint die Listenübersicht aus der Startansicht in einer gefilterten Variante.

Die Android-Menü-Leiste der Anwendung zeigt dann einen Button mit der Aufschrift „Filter löschen“ an, über die der Filtervorgang wieder aufgehoben werden kann. Das bedeutet, dass beim Klick auf diesen Button in der Listenübersicht wieder alle erstellten Listen angezeigt werden und nicht nur solche, denen ein bestimmter Tag zugewiesen wurde.

5 Einstellungen

Die Anwendung TabScript lässt sich in einigen Bereichen an die Benutzerwünsche anpassen. Hierzu gehören das äußere Erscheinungsbild der Anwendung, Einstellungen für Benachrichtigungen bei Fälligkeit eines Tasks sowie Optionen zur Handschrifterkennung für Experten.

Das Einstellungs-Menü kann über die Android-Menü-Leiste und dort über den Button „Einstellungen“ erreicht werden (siehe Abbildung 5.1). Die folgenden Abschnitte geben einen Überblick über die möglichen Einstellungen von TabScript.

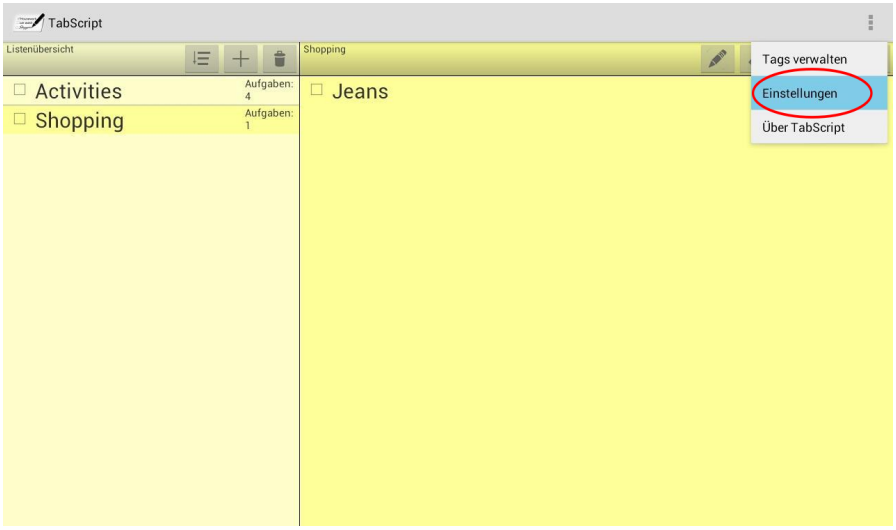


Abbildung 5.1: Um in das Einstellungs-Menü der Anwendung zu gelangen, muss der Button „Einstellungen“ der Android-Menü-Leiste betätigt werden.

5.1 Erscheinungsbild

Die Einstellungen zum Aussehen der Anwendung enthalten den Eintrag „Theme“, unter dem sich per Klick zwei verschiedene Motive für die Anwendung

auswählen lassen (siehe Abbildung 5.2). Beibehalten werden kann das ursprüngliche Motiv entweder durch die Betätigung des Abbrechen-Buttons oder durch erneuten Klick auf diese Auswahl.

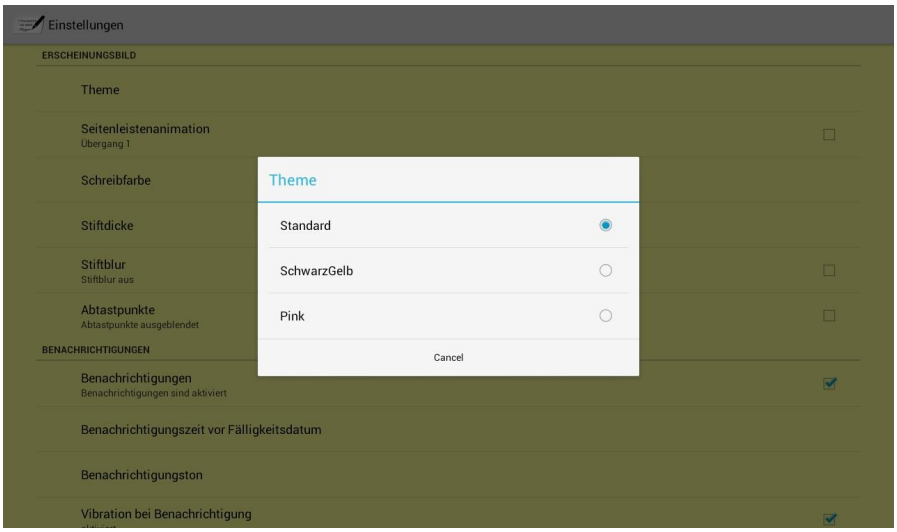


Abbildung 5.2: Auswahl der verschiedenen Themes für die Anwendung.

Weiterhin kann der Animations-Übergang für die ausklappbare Seitenansicht, die sich aus der Schreib-Ansicht (siehe Abschnitt 2.3) aufrufen lässt, eingestellt werden. Standardmäßig ist „Übergang 1“ ausgewählt. Um zwischen den beiden zur Verfügung stehenden Übergängen zu wechseln, wird das nebenstehende Options-Kästchen angeklickt.

Der Eintrag „Schreibfarbe“ dient zur Auswahl der Trajektorien-Farbe in der Schreib-Ansicht. Auch die Dicke des Stiftes kann unter „Stiftdicke“ eingestellt werden. Ist die Option „Stiftblur“ aktiviert, so wird die Handschrift-Trajektorie mit einem Unschärfe-Effekt versehen. Außerdem lässt sich unter „Abtastpunkte“ auswählen, ob die Trajektorienpunkte in der Schreib-Ansicht angezeigt werden sollen.

5.2 Benachrichtigungen

Abbildung 5.3 gibt einen Überblick über die möglichen Einstellungen zu den Aufgaben-Benachrichtigungen der Anwendung. Über den Eintrag „Benachrichtigungen“ lässt sich einstellen, ob die Funktion der Benachrichtigungen überhaupt genutzt werden sollen. Dies geschieht durch einen Klick auf das zugehörige Kästchen am rechten Bildschirmrand.

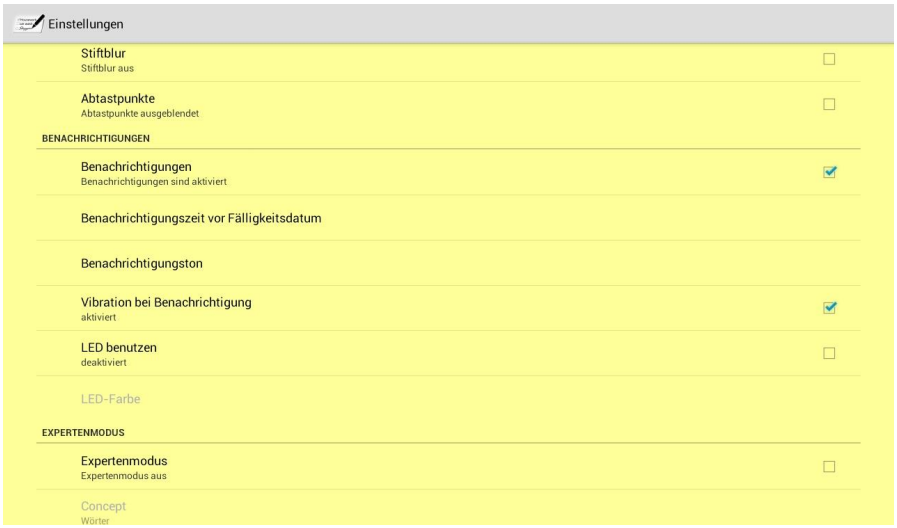


Abbildung 5.3: Konfigurationsmöglichkeiten für die Benachrichtigungen bei Fälligkeitszeitpunkt von Aufgaben.

Durch Klick auf den Eintrag „Benachrichtigungszeit vor Fälligkeitsdatum“ öffnet sich ein Auswahlfenster, in dem sich wählen lässt, wie viele Minuten vor dem Fälligkeitszeitpunkt einer Aufgabe eine Erinnerung erscheinen soll (siehe Abbildung 5.4).

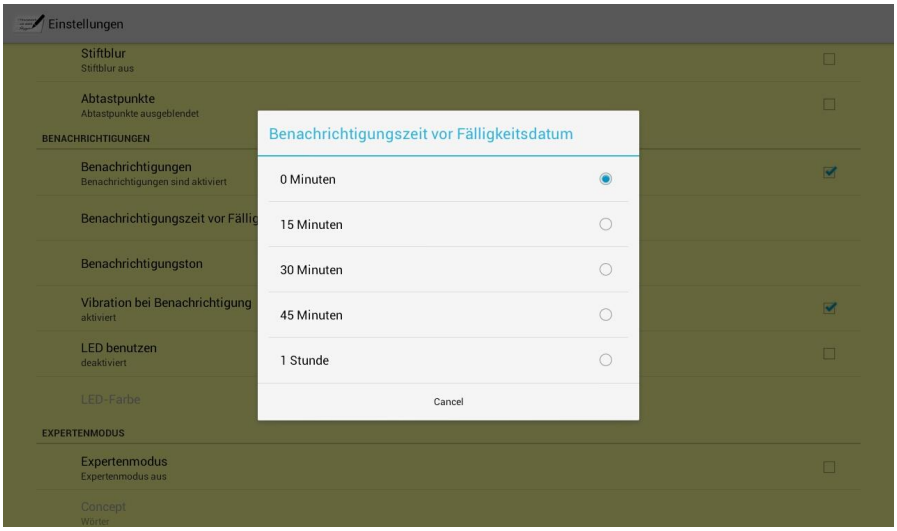


Abbildung 5.4: Einstellung des Erinnerungszeitpunktes für Aufgaben mit Fälligkeitszeitpunkt.

Der Ton für die Erinnerung kann unter dem Eintrag „Benachrichtigungston“

durch einen Klick auf den jeweiligen Ton und eine anschließende Bestätigung mit dem Button „OK“ festgelegt werden. Über einen Abbrechen-Button lässt sich die ursprüngliche Auswahl beibehalten (siehe Abbildung 5.5).

Über den Eintrag „Vibration bei Benachrichtigung“ lässt sich festlegen, ob mit dem Erscheinen einer Benachrichtigung auch ein kurzer Vibrationsalarm ausgeführt werden soll. Durch Klick auf das nebenstehende Kästchen am rechten Bildschirmrand lässt sich diese Funktionalität sowohl aktivieren.

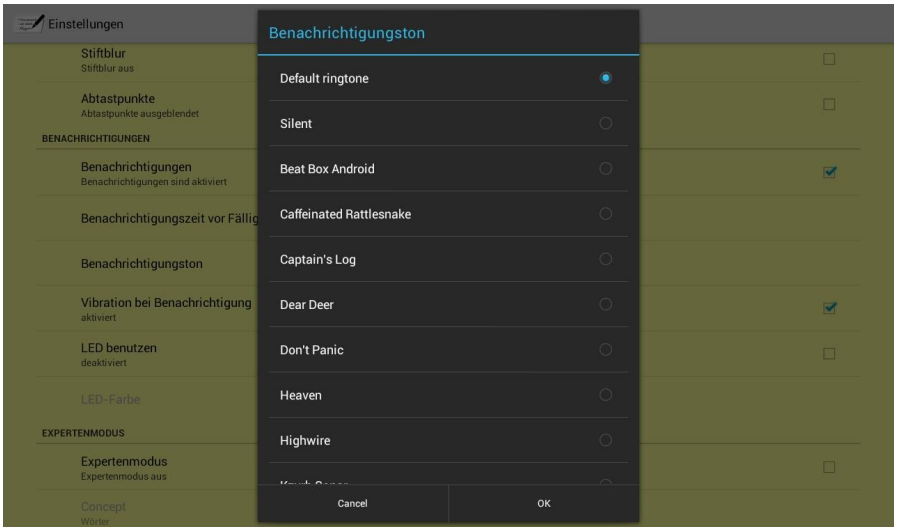


Abbildung 5.5: Auswahl des Benachrichtigungstones für Aufgaben mit Fälligkeitszeitpunkt.

Weiterhin lässt sich einstellen, dass beim Erscheinen einer Benachrichtigung auch die eingebaute LED eines Tablets leuchtet. Aktiviert wird diese Funktion über einen Klick auf das nebenstehende Kästchen am rechten Bildschirmrand. Ist diese Funktion aktiv, so erscheint ein neuer Eintrag mit dem Titel „LED-Farbe“. Per Klick auf diesen öffnet sich ein Auswahlfenster, in dem sich die gewünschte Farbe der leuchtenden LED festlegen lässt. Der Abbrechen-Button dient dazu, die zuletzt getroffene Auswahl zu erhalten.

5.3 Expertenmodus

Im Expertenmodus können Einstellungen vorgenommen werden, die nur für erfahrene Benutzer sinnvoll sind, die sich mit dem Thema Handschrifterkennung auskennen. Der Modus wird durch einen Klick auf das Kästchen neben dem Eintrag „Expertenmodus“ aktiviert.

Eine weitere Funktionalität, die im Expertenmodus durchgeführt werden kann, ist die automatische Konfiguration der Anwendung mit festgelegten Listen, Aufgaben und Tags. Diese Option wird vor allem für die Evaluation von TabScript verwendet. Zusätzlich gibt es auch eine Möglichkeit, alle vordefinierten Listen, Aufgaben und Tags der Anwendung zu löschen.

B. Fragebogen

Fragebogen zur Benutzerfreundlichkeit der Android-Anwendung TabScript

Liebe Teilnehmerin, lieber Teilnehmer,

wir, die Projektgruppe TabScript des Fachbereiches Informatik von der TU-Dortmund, haben im Rahmen einer zwei-semestrigen Veranstaltung eine Android-Applikation entwickelt, welche eine ToDo-Liste für Smartphones und Tablets realisiert. Diese Anwendung ermöglicht es dem Benutzer, Aufgaben per Handschrift auf dem Medium zu erstellen, die schließlich erkannt und als Computerschrift gespeichert werden.

Für die abschließende Evaluation unserer Arbeit, würden wir Ihnen gerne ein paar Fragen zur Benutzerfreundlichkeit der grafischen Oberfläche und zur Erkennung der Handschrift stellen. Die gesammelten Daten dienen ausschließlich universitären Zwecken und werden vertraulich behandelt. Die Beantwortung der Fragen erfolgt selbstverständlich anonym.
Zur Erleichterung der Bedienung, wird an dieser Stelle ein kurzer Überblick über die Funktionen der Anwendung
gegeben. Mit TabScript lassen sich Listen erstellen, betrachten und löschen. Zu jeder dieser Listen können Aufgaben hinzugefügt werden. Neben dem Betrachten und Löschen von Aufgaben ist es auch möglich, Aufgaben als erledigt zu markieren. Des Weiteren lassen sich Listen mit einem sogenannten "Tag“ versehen. Mit Hilfe dieser Markierung kann der Benutzer aus einer Menge von Listen bestimmte Listen herausfiltern, um nur diese anzeigen zu lassen. Sowohl Listen als auch die Aufgaben einer Liste können mit TabScript nach bestimmten Kriterien sortiert werden. Die Startansicht der Applikation zeigt dem Benutzer bereits alle erstellten Listen an.

Vielen Dank, dass Sie uns durch Ihre Teilnahme bei der Auswertung von TabScript unterstützen!

Angaben zum Benutzer

1. **Alter:**

   .....

2. **Geschlecht:**
   Wählen Sie alle zutreffenden Antworten aus.

   - [ ] weiblich
   - [ ] männlich

3. **Zum Schreiben verwendete Hand:**
   Wählen Sie alle zutreffenden Antworten aus.

   - [ ] linke Hand
   - [ ] rechte Hand

Erkennung

4. **Erstes geschriebenes Wort:**

.....

.....

.....

.....

.....

5. **Das erste Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

6. **Zweites geschriebenes Wort:**

.....

.....

.....

.....

.....

7. **Das zweite Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

8. **Drittes geschriebenes Wort:**

.....

.....

.....

.....

.....

9. **Das dritte Wort wurde erkannt an...**
   Markieren Sie nur ein Oval.

   - ◯ erster Stelle
   - ◯ zweiter Stelle
   - ◯ dritter Stelle
   - ◯ keiner Stelle

10. **Viertes geschriebenes Wort:**

    .....

    .....

    .....

    .....

    .....

11. **Das vierte Wort wurde erkannt an...**
    Markieren Sie nur ein Oval.

    - ◯ erster Stelle
    - ◯ zweiter Stelle
    - ◯ dritter Stelle
    - ◯ keiner Stelle

12. **Fünftes geschriebenes Wort:**

    .....

    .....

    .....

    .....

    .....

13. **Das fünfte Wort wurde erkannt an...**
    Markieren Sie nur ein Oval.

    - ◯ erster Stelle
    - ◯ zweiter Stelle
    - ◯ dritter Stelle
    - ◯ keiner Stelle

14. **Sechstes geschriebenes Wort:**

.....

.....

.....

.....

.....

15. **Das sechste Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

16. **Siebtes geschriebenes Wort:**

.....

.....

.....

.....

.....

17. **Das siebte Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

18. **Achtes geschriebenes Wort:**

.....

.....

.....

.....

.....

19. **Das achte Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

20. **Neuntes geschriebenes Wort:**

.....

.....

.....

.....

.....

21. **Das neunte Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

22. **Zehntes geschriebenes Wort:**

.....

.....

.....

.....

.....

23. **Das zehnte Wort wurde erkannt an...**
Markieren Sie nur ein Oval.

- ◯ erster Stelle
- ◯ zweiter Stelle
- ◯ dritter Stelle
- ◯ keiner Stelle

24. **Die Handschrifterkennungs-Funktion der Anwendung funktioniert...**
Markieren Sie nur ein Oval.

	1	2	3	4	5	
sehr gut	◯	◯	◯	◯	◯	gar nicht

Optik

25. **Die grafische Oberfläche der gesamten Startansicht bewerte ich zusammenfassend als...**
Markieren Sie nur ein Oval.

	1	2	3	4	5	
übersichtlich	◯	◯	◯	◯	◯	unübersichtlich

26. **Die Anbringung der einzelnen Buttons bewerte ich zusammenfassend als...**
Markieren Sie nur ein Oval.

	1	2	3	4	5	
übersichtlich	◯	◯	◯	◯	◯	unübersichtlich

27. **Die Teilung der grafischen Oberfläche bewerte ich zusammenfassend als...**
Markieren Sie nur ein Oval.

	1	2	3	4	5	
übersichtlich	◯	◯	◯	◯	◯	unübersichtlich

Bedienung

28. **Die Bedienung der Applikation bewerte ich als...**
Markieren Sie nur ein Oval.

	1	2	3	4	5	
sehr intuitiv	◯	◯	◯	◯	◯	gar nicht intuitiv

Anmerkungen und Verbesserungsvorschläge

29. **Folgende Fehler habe ich in der Anwendung feststellen können:**

.....

.....

.....

.....

.....

30. **Die Umsetzung der folgenden Option/ des folgenden Menüs erscheint mir besonders verbesserungswürdig:**

.....

.....

.....

.....

.....

31. **Die Umsetzung der folgenden Option/ des folgenden Menüs erscheint mir besonders gut:**

.....

.....

.....

.....

.....

C. Pflichtenheft

Projektgruppe: Tabscript

Pflichtenheft

TU Dortmund

Sulejman Begovic
Rebecca Doherty
Shinazi Faruki
Daria Filatova
Nina Hesse
Dennis Kesper
Julian Kürby
Niclas Raabe
Johann Straßburg
Christian Wieprecht

1. November 2013

Inhaltsverzeichnis

1 Einleitung

Die zu entwickelnde Software soll eine ToDo Anwendung für Tablets realisieren, die das Erstellen und Planen von zu erledigenden Aufgaben mittels Listen ermöglicht. Die Aufgaben können dabei mit einem Erfüllungstermin versehen, nach Tags (Auszeichnungen) gegliedert oder als erledigt markiert werden.

Durch die immer größere Verbreitung von Tablets und ihren mobilen Einsatz hat die Entwicklung tabletbasierter Applikationen an zunehmender Bedeutung gewonnen. Insbesondere die räumliche Flexibilität stellt für die geplante Anwendung einen wesentlichen Nutzen dar. Auf diese Weise können ToDo Listen jederzeit auch unterwegs, z.B. auf Reisen oder Konferenzen erstellt und bearbeitet werden.

In der Regel wird der berührungssensitive Bildschirm des Tablets dazu genutzt, Texteingaben über eine virtuelle Tastatur zu realisieren. Allerdings hat das Einblenden der Tastatur in den Bildschirm den Nachteil, dass dieser teilweise verdeckt wird. Wird die Verdeckung minimiert, müssen jedoch auch die Tasten der Bildschirmtastatur kleiner dargestellt werden, so dass die Eingabe zunehmend erschwert wird. Eine Lösung bietet daher die Integration einer Handschrifterkennung. Diese stellt gleichzeitig eine besonders natürliche Form der Eingabe dar und ermöglicht dem Benutzer das schnelle und einfache Schreiben von Texten. Zielgruppe der zu entwickelnden Anwendung sind daher Benutzer welche eine besonders intuitive Eingabemethode wünschen. Die Handhabung des Tablets soll dann der eines realen Notizblockes ähneln.

2 Zielbestimmungen

Im Folgenden werden die zu erreichenden Ziele durch die Projektgruppe zu entwickelnden Anwendung festgelegt. *Musskriterien* und *Sollkriterien* sind dabei auf jeden Fall zu erreichen. Allerdings sind nur *Musskriterien* für die Funktionsfähigkeit der Anwendung essentiell. *Kannkriterien* sind wünschenswerte Erweiterungen der Anwendung, über deren Umsetzung jedoch erst im Laufe der Entwicklung entschieden wird. *Abgrenzungskriterien* beschreiben Funktionen, die von der Anwendung *nicht* zu erfüllen sind und die somit eine Abgrenzung zu ähnlichen Anwendungen darstellen.

2.1 Musskriterien

- Listen von Aufgaben anlegen/löschen/anordnen
- Aufgaben in einer Liste hinzufügen/löschen/anordnen und als unerledigt/erledigt markieren
- Listen löschen (einschließlich Aufgaben)
- jede Aufgabe wird per Handschrift eingegeben, die Trajektorie wird durch ein Handschrifterkennungssystem in Maschinenschrift umgewandelt
- Handschrifterkennung per Schrift- und Sprachmodell (Nutzung von Hidden Markov Modellen und n-Gram-Modellen)
- Modellbildung und -training mit Hilfe des *ESMERALDA-Toolkits*[1]
- Korrektur von ganzen Worten durch Neueingabe (ebenfalls per Handschrift)
- geeignete Abwägung zwischen benötigtem Speicherplatz des Modells und Erkennungsgüte des Handschrift-Erkenners
- bei Eingabe einer Aufgabe werden Wortvorschläge angezeigt
- Markierung einer Liste als erledigt, falls alle Aufgaben erledigt sind
- Unterstützung von Portrait/Landscape (außer nach Beginn des Schreibvorgangs)

2.2 Sollkriterien

- Aufgaben können sortiert werden (etwa Alphabetisch oder nach Datum)
- Listen können Tags gegeben werden (Tags sind Kategorien, die vom Nutzer verwaltet werden können)
- Die Anzeige der Listen kann nach Tags gefiltert werden
- Anzeige der aktuellen Liste bei Eingabe einer Aufgabe

[1] http://sourceforge.net/projects/esmeralda/

- Erstellungs- und Fälligkeitsdatum von Aufgaben
- Lösch-Modus zum gleichzeitigen Löschen mehrerer Aufgaben
- Optionales Einschalten des Bildschirmtastatur

2.3 Kannkriterien

- Eine Liste, die alle Aufgaben anzeigt. In dieser Liste sollen keine Aufgaben erstellt werden können.
- Benachrichtigung/Erinnerung an Fälligkeitsdatum bei aktiver Anwendung (bzw. beim Start der Anwendung)
- Synchronisation mit Google Account (Google Tasks)
- Verbesserung/Erweiterung der Handschrifterkennung durch Hinzunahme von Web-Kontexten

2.4 Abgrenzungskriterien

- Die Trajektorie der Handschrift wird nach der Worterkennung nicht mehr angezeigt
- Kein Abhaken ganzer Listen
- Keine Verwendung von Multimedia-Daten in Aufgaben (bspw. Fotos)
- Keine Erinnerungsfunktion, die einen Hintergrund-Prozess der Anwendung voraussetzen würde, z. B. durch die Kalenderfunktion von Android
- Keine Bereitstellung eines Widgets für den Homescreen
- Kein Mehrbenutzer-Betrieb

3 Produkteinsatz

Das folgende Kapitel beschreibt die näheren Bedingungen und Umstände, unter denen die Anwendung zum Einsatz kommt.

3.1 Anwendungsbereiche

Nutzer verwenden die Anwendung zum Verwalten von Listen von Aufgaben. Besonders soll hierbei auf die Verwendung der Bildschirmtastatur verzichtet werden und mit der Handschrifterkennung eine alternative Eingabemethode eingeführt werden. Ein Hauptziel der Anwendung ist somit die Evaluierung von Online-Handschrifterkennung auf Android-Geräten.

3.2 Zielgruppen

Die Zielgruppe ist charakterisiert durch private Endnutzer, die Aufgaben-Listen in ihrem Tablet verwalten wollen und darüber hinaus die Texteingabe per Handschrift der herkömmlichen Eingabe per Bildschirmtastatur bevorzugen.

3.3 Betriebsbedingungen

Das System soll unter alltäglichen Android-Gerät Umgebungseigenschaften arbeiten. Die Beseitigung möglicher äußerer Störfaktoren, z.B. durch Sonneneinstrahlung liegt dabei in der Verantwortung des Benutzers. Weiterhin ist die tägliche Betriebszeit variabel, und hängt von der Aktivität des Benutzers ab. Passive Funktionen, die nicht explizit vom Nutzer angestossen werden, sind lediglich das geräteinterne Speichern der Aufgabenlisten sowie die Synchronisierung der Daten mit dem verbundenen Google-Account bei Internet-Verfügbarkeit.

4 Produktumgebung

Das Produkt stellt einige Anforderungen an die Umgebung, in der es ausgeführt wird. Im Folgenden werden Bedingungen an benötigte Software-Installationen definiert, die zu benutzende Hardware spezifiziert und bedingte Kommunikationsmöglichkeiten des Produkts mit ausgelagerten Diensten beschrieben.

4.1 Software

Das Produkt ist für den Einsatz auf mobilen Geräten vorgesehen, die das Android-Betriebssystem aufweisen. Einige GUI-Funktionalitäten wie die Action Bar erfordern die Version 3.0 des Android-Betriebssystems, sodass dies die Mindestanforderung darstellt.

4.2 Hardware

Unerlässlich für die Bedienbarkeit des Produkts ist eine berührungssensitive Anzeige des mobilen Geräts; die GUI-Navigation sowie die Schrifteingabe ist per Finger oder zugehörigem Stift zu führen. Da eine große Vielfalt von Hardwarekonfigurationen mit dem Android-Betriebssystem ausgeliefert wird, erfolgt eine Formulierung der Mindestanforderungen an Prozessor und Arbeitsspeicher eines vorliegenden Tablet-Computers. Die CPU weist eine Taktrate von mindestens 1 GHz auf und das Gerät verfügt über 1 GB RAM. Ziel der Implementierung ist es an dieser Stelle, die Performanz auf einem Referenzgerät[2] zu optimieren, welches die Mindesthardwareanforderungen genau erfüllt. Der größte Rechenaufwand in Form der Handschrifterkennung wird das zugrunde liegende Gerät nur soweit auslasten, dass die interaktive Benutzbarkeit aufrecht erhalten bleibt. Zur Gewährleistung einer gut benutzbaren Schreibfläche ist ein Gerät mit ausreichend großer Bilddiagonale empfehlenswert, um die Eingabe von mehreren (längeren) Wörtern in einer Textzeile zu ermöglichen. Kleinere Anzeigen ermöglichen zwar grundsätzlich die Benutzung des Produkts, verhindern jedoch eine komfortable Schriftführung. Optimale Benutzbarkeit gewährt ein Bildschirm ab 20 cm Diagonale [1] was in etwa der Benutzung eines DIN A5 Blattes (ca. 25 cm Diagonale) entspricht. Das Programm speichert im Verlauf des Produktgebrauchs die generierten Nutzerinhalte auf dem lokalen Speichermedium des zugrunde liegenden Geräts.

4.3 Orgware

Der eigentliche Betrieb des Produkts ist ohne eine Internetanbindung möglich. Im Falle einer optionalen Synchronisation mit einer Online-Datenbank ist jedoch ein Internetzugang erforderlich. Konkret wird gegen die Google Tasks API[3] implementiert. Der Abgleich mit Online-Daten erfolgt automatisch bei aufrechter Internetverbindung, sodass keine permanente Internetverbindung nötig ist.

[2] z.B. Samsung Galaxy Tab 10.1

[1] ,

[3] `https://developers.google.com/google-apps/tasks/v1/reference/?hl=de`

5 Produktfunktionen

/F.../ weißt einer Produktfuntion eine Nummer zu. Bei einem Zusatz S bzw. K wird dabei auf die Soll bzw. Kann Kriterien aus den Zielbestimmungen verwiesen. Ohne Zusatz beschreiben die Funktionen ein Musskriterium.

5.1 Aufgabeneintrag

Der Benutzer kann im Eingabefenster seine Aufgabe handschriftlich verfassen.

/F010/ *Aufgabe eintragen*: Eine neue Aufgabe wird in dem Eingabefenster erstellt. Der Benutzer trägt seine Aufgabe in dem dafür vorgesehenen Feld, mittels eines Fingers oder Stifts, als Handschrift ein. Das System erkennt diese und zeigt eine mögliche Interpretation der Handschrift als Maschinenschrift an.

/F020/ *Eintrag korrigieren*: Der interpretierte Eintrag kann durch Auswahl von Alternativen im Ausklappmenü editiert werden.

/F030/ *Eintrag bestätigen*: Ein Eintrag kann über die "Häkchen"-Taste bestätigt und somit in die zuvor geöffnete Liste eingetragen werden, woraufhin die Aufgabenliste aufgerufen wird.

/F040/ *Eintrag bestätigen und nächsten eintragen*: Ein Eintrag kann über die "Häkchen+"-Taste bestätigt und somit in die zuvor geöffnete Liste eingetragen werden, woraufhin ein neues Aufgabe-Eintragen Fenster für das Eintragen einer weiteren Aufgabe geöffnet wird.

/F050/ *Eintrag abbrechen*: Ein Eintrag kann über die Zurück-Taste des Systems widerrufen werden, woraufhin das Eingabefenster geschlossen und die geöffnete Aufgabenliste wieder angezeigt wird. Ebenso kann der Eintrag über die Abbruch-Taste gelöscht werden, ohne das Eingabefenster zu verlassen.

/F060S/ *Aufgabenfälligkeitsdatum eintragen*: Das Fälligkeitsdatum für die Aufgabe, kann über das Kontextmenü mithilfe eines Datum-Auswahl-Fensters eingetragen werden .

/F070S/ *Aufgabenliste einblenden*: Die geöffnete Aufgabenliste kann über eine Einblende-Taste an der linken Seite ausgeklappt werden. Währenddessen ist ein Bearbeiten der Aufgabe nicht möglich.

/F080S/ *Aufgabenliste ausblenden*: Bei aufgeklappter Aufgabenliste, wird diese wieder über die Einklapp-Taste an der rechten Seite der Liste eingeklappt.

/F090S/ *Bildschirmtastatur einblenden*: Zum Eintragen über eine Tastatur, kann die Bildschirmtastatur über das Kontextmenü eingeblendet werden. Die Handschrifterkennung wird dabei deaktiviert.

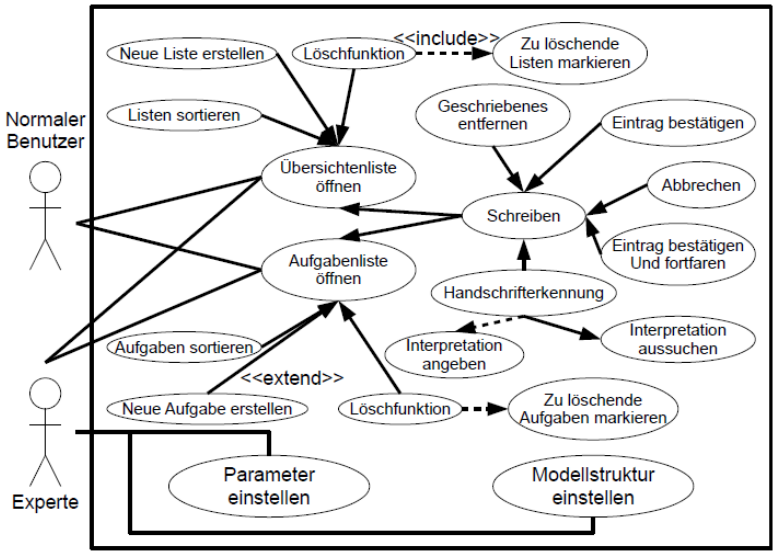


Abbildung 1: Use Case Diagramm des Programms. Der normale Benutzer hat Zugang zu allen Funktionen, die die Aufgabenverwaltung betreffen. Zusätzlich hat der Experte die Möglichkeit Parameter und Modellstruktur einzustellen, die ihm ermöglichen die Funktionen der Handschrifterkennung direkt zu beeinflussen.

5.2 Aufgabenliste

Die Aufgabenliste bietet Möglichkeiten zur Übersicht und Verwaltung von Aufgaben.

/F100/ *Aufgabe hinzufügen*: Das Aufgabe-Erstellen Fenster wird über die "+"-Taste in einer Aufgabenliste geöffnet.

/F110S/ *Löschmodus aktivieren/deaktivieren*: Der Löschmodus kann über das zugehörige "Papierkorb"-Symbol in der oberen Leiste aktiviert bzw. deaktiviert werden.

/F120/ *Aufgaben entfernen*: Aufgaben können auf unterschiedliche Weise entfernt werden:

- Klicken des ×-Symbols eines Aufgaben-Items in einer Aufgabenliste auf der rechten Seite, bei eingeschaltetem Löschmodus (**F110S**), entfernt den gewählten Eintrag.
- Das Löschen einer Aufgabenliste (**F250**) löscht gleichzeitig alle Aufgaben innerhalb der Liste.

/F130/ *Aufgabe ändern*: Eine Aufgabe kann über eine entsprechende Optionstaste durch Neueintragung der Aufgabe geändert werden.

/F140/ *Aufgabe als "erledigt" markieren*: Der Benutzer kann eine Aufgabe als erledigt markieren, indem er das Auswahlkästchen an der linken Seite eines Aufgaben-Elements anklickt.

/F150/ *Markierung von als "erledigt" markierten Aufgaben aufheben*: Der Benutzer kann die markierung einer Aufgabe aufheben, indem er das Auswahlkästchen an der linken Seite eines Aufgaben-Elements anklickt.

/F160S/ *Aufgabenliste sortieren*: Die Sortierfunktion für Aufgaben kann in jeder Aufgabenliste über das Kontextmenü gewählt werden.

/F170/ *Aufgabenliste manuell sortieren*: Die Aufgaben-Elemente einer Aufgabenliste können manuell per Drag-and-Drop innerhalb der Liste verschoben werden.

/F180S/ *Aufgabenliste taggen*: Der Aufgabenliste kann über ein Kontextmenü, erreichbar über das Menü an der oberen Leiste, ein Tag zugewiesen werden.

/F200S/ *Zwischen Listen wechseln*: Der Benutzer kann zwischen Listen wechseln, indem er mit einer horizontalen Wischbewegung nach links oder rechts eine Liste auswählt.

/F210S/ *Fälligkeitsdatum für Aufgaben eintragen*: Das Fälligkeitsdatum für eine Aufgabe, kann über ein Optionsmenü an einem Aufgaben-Item eingestellt werden.

5.3 Übersichtsmenü

Im Übersichtsmenü können die Aufgabenlisten verwaltet werden.

/F220/ *Aufgabenliste anlegen*: Der Benutzer kann im Übersichtsmenü eine neue Aufgaben-Liste mithilfe der "+"-Taste erstellen, sowie einen Namen handschriftlich vergeben.

/F230/ *Aufgabenliste öffnen*: Eine Aufgabenliste kann über einen Klick auf das Aufgabenlisten-Element geöffnet werden.

/F240/ *Zwischen Aufgabenlisten scrollen*: Um zu einer bestimmten Aufgabenliste zu gelangen, kann zur gewünschten Aufgabenliste vertikal gescrollt werden.

/F250/ *Aufgabenliste löschen*: Eine Aufgabenliste kann vom Benutzer im Listenübersichtsmenü über ein Löschkontextmenü gelöscht werden. Das Löschen der Liste muss bestätigt werden und bewirkt ein Löschen der Liste, mitsamt der beinhaltenden Aufgaben.

/F260S/ *Aufgabenlisten sortieren*: Die Sortierfunktion für Aufgabenlisten kann über das Kontextmenü, erreichbar über das Menü an der oberen Leiste, gewählt werden.

/F270S/ *Aufgabenlisten nach Tags filtern*: Die Aufgaben können über das Kontextmenü nach ausgewählten Tags gefiltert werden.

/F280K/ *Alle-Aufgaben-Liste*: Die Alle-Aufgaben-Liste kann im Übersichtsmenü aufgerufen werden. Sie enthällt alle Aufgaben. Aufgaben können gefiltert, sortiert und markiert/demarkiert werden. Weitere Änderungen sind nicht möglich.

5.4 Optionen

/F290K/ *Aufgabenliste mit Googletasks synchronisieren*: Der Benutzer kann seine Aufgabenliste mit Google Tasks über die Option im Optionsmenü synchronisieren und einrichten.

/F300S/ *Listentags verwalten*: Der Benutzer kann neue Tags erstellen und bestehende Tags löschen. Beim Löschen der Tags, verlieren die entsprechenden Aufgabenlisten lediglich diese Tags, bleiben ansonsten jedoch unverändert.

/F310/ *Expertenmodus*: Der Benutzer kann im Optionsmenü zwischen Expertenmodus und normalem Benutzer wählen 5.1.

- **normaler Benutzer**: Im Modus "normaler Benutzer" befinden sich alle Parameter und Modellstrukturen im Urzustand.
- **Experte**: Im Modus "Experte" werden Optionsmöglichkeiten angezeigt, Parameter und Modellstrukturen zu modifizieren. Die Veränderungen der Parameter im Expertenmodus verändern die Arbeitsweise des Programms und können die Effektivität unter Umständen enorm beeinträchtigen. Der Expertenmodus sollte daher nur von einem fortgeschrittenem Nutzer verwendet werden.

6 Produktdaten

Jeder Punkt **/D???/** stellt einen Datensatz dar.

/D010/ *Benutzerdaten für Google Tasks:* Für die Anbindung an Google Tasks müssen Anmeldeinformationen für Google Tasks sowie Synchronisation gespeichert werden:

- Kennung
  - **Benutzername** *(eindeutig)*
  - **Passwort** *(verschlüsselt)*
- Sonstige Daten
  - **letzte Synchronisation** *(Datum)*
  - **letzte Änderung** *(Datum)*

/D110/ *Listenelemente:* Die einzelnen Listeneinträge

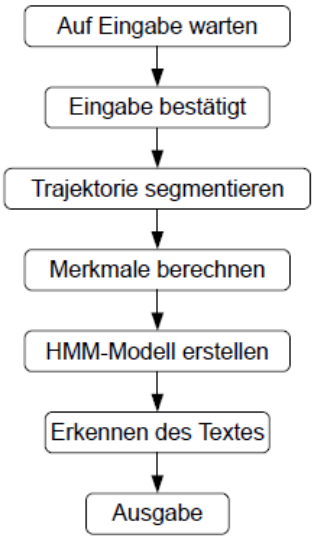
- **Name: Beschreibung der Aufgabe**
- **AufgabenId** *(eindeutig)*
- **Status des Listeneintrags** *(checked, unchecked)*
- **Zeitstempel** *(Datum)*
- **Fälligkeitsdatum** *(optional, Datum)*

/D120/ *Listen:* Die Daten, die zu den einzelnen Listen gehören.

- **Listenname**
- **ListenId** *(eindeutig)*
- **Listentag** *(optional)*
- **Zeitstempel***(Datum)*
- **Listeneinträge**

7 Produktleistungen

Als wesentlicher Bestandteil des Produkts muss die Handschrifterkennung einer gewissen Erkennungsrate genügen. Schließlich soll der Benutzer seine native Schreibart nicht grundlegend ändern, um erfolgreich mit dem Programm arbeiten zu können.
Bei internet-basierten Synchronisierungsschritten transferiert das Programm nicht erheblich mehr Datenvolumen, als die eigentliche Traglast der Daten erfordert. Zusätzlich geschieht dies auch nur zu definierten Zeitpunkten, damit der Nutzer explizit Kontrolle über seine Daten behält.
Das ausgelieferte Programm selbst genügt stets der Anforderung einer interaktiven Benutzbarkeit; es treten bei rechenbedingten Leistungsengpässen entsprechende Hinweise wie Fortschrittsanzeigen auf, damit der Nutzer über den Status seiner Anwendung informiert ist.
Die nutzergenerierten Inhalte der Anwendung bleiben stets konsistent. Die Veränderung von Nutzerinhalten durch das Programm (z.B. Löschen, Verschieben von Listenpositionen) werden entweder durch den Nutzer initiiert oder von diesem bestätigt beziehungsweise akzeptiert. Insbesondere bleiben bei unvorhergesehenen Programmabbrüchen, Verbindungsstörungen oder Benutzerfehlbedienungen die vor der betreffenden Sitzung auf dem Gerät vorhandenen Daten der Anwendung erhalten, sofern der Nutzer nicht explizit von der Anwendung über deren möglichen Verlust aufmerksam gemacht wurde.

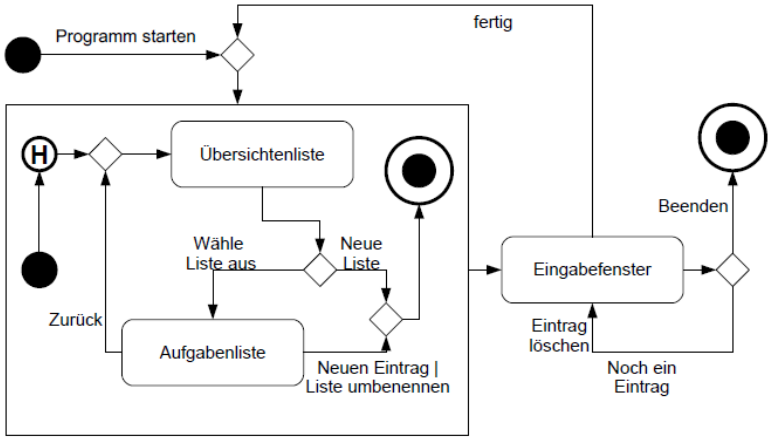


8 Benutzungsoberfläche

Im Folgenden wird der Aufbau und das Aussehen der Benutzungsoberfläche der zu implementierenden Handschrifterkennungs-Anwendung präsentiert.

8.1 Dialogstruktur

Dieser Abschnitt gibt einen Überblick über die grobe Dialogstruktur der zu erstellenden Android-Anwendung. Die Bedienung der Applikation erfolgt stets mit einem speziellen Tablet-Stift oder den Fingern. Abbildung 8.1 visualisiert die einzelnen Aktivitäten, die bei der Verwendung der Handschrifterkennungs-Anwendung ausgeführt werden können, sowie ihren Gesamtzusammenhang.



8.1.1 Hauptseite

/B01/ Hauptseite: siehe Abbildung 2

- ◇ Die Hauptseite besteht aus einer Übersicht über alle angelegten Listen, der sogenannten `ListActivity`.
- ◇ Die Hauptseite erscheint beim Neustart der Anwendung und solange noch keine Liste durch den Benutzer angelegt worden ist.
- ◇ Die Hauptseite umfasst im unteren Bereich eine Auflistung der Namen aller gespeicherten Listen.
- ◇ Standardmäßig wird beim ersten Programmstart auf der Hauptseite eine leere Liste angezeigt.
- ◇ Die zugehörige Menüleiste befindet sich am oberen Bildschirmrand und erstreckt sich über die gesamte Bildschirmbreite (`Activity Bar`).
- ◇ Die Menüleiste beinhaltet (von links nach rechts betrachtet) folgende Elemente:
  - ∗ ein Symbol für das Programm-Menü (Liste),

- den Fenstertitel **Listen**,
- ein Symbol für den Aufruf des Löschmenüs (Papierkorb),
- ein Symbol für den Aufruf des Menüs zum Erstellen einer neuen Liste (Plus-Zeichen).

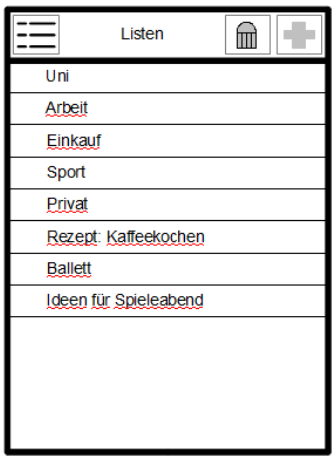


Abbildung 2: Hauptseite der Anwendung.

8.1.2 Aufgabenliste

/B02/ Aufgabenliste: siehe Abbildung 3

- ◇ Wurde vom Benutzer bereits mindestens eine Liste angelegt und die Anwendung nicht erst neugestartet, so erscheint beim Weiterverwenden der Anwendung stets die Aufgabenliste (`ItemActivity`).
- ◇ Die Aufgabenliste zeigt die zuletzt verwendete Liste zusammen mit ihren Einträgen (untereinander angeordnet).
- ◇ Für jeden Listeneintrag ist in der gleichen Zeile auch eine zugehörige Checkbox vorhanden.
- ◇ Die zugehörige Menüleiste befindet sich am oberen Bildschirmrand und erstreckt sich über die gesamte Bildschirmbreite (`Activity Bar`).
- ◇ Die Menüleiste beinhaltet (von links nach rechts betrachtet) folgende Elemente:
  - ∗ ein Symbol für das Programm-Menü (Liste),
  - ∗ den Fenstertitel <**Listenname**>,
  - ∗ ein Symbol für den Aufruf des Löschmenüs (Papierkorb),
  - ∗ ein Symbol für den Aufruf des Menüs zum Erstellen eines neuen Listeneintrags (Plus-Zeichen).

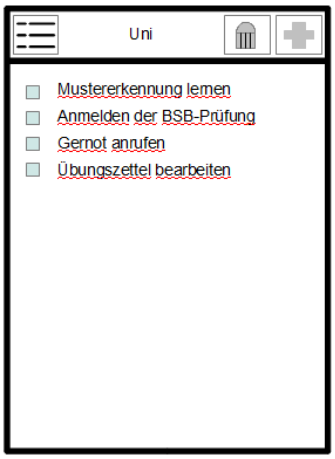


Abbildung 3: Aufgabenliste der Anwendung.

8.1.3 Handschrift-Ansicht

/B03/ Handschrift-Ansicht: siehe Abbildung 4

- ◇ Der untere Bereich dieser Ansicht dient als Möglichkeit zur Worteingabe per Handschrift.
- ◇ Der Hintergrund dieser Ansicht soll mit horizontalen Linien hinterlegt sein.
- ◇ In der rechten oberen Ecke des Bereiches befindet sich eine „x“-Taste zum zurücksetzen des Handschrift-Bereiches.
- ◇ In der Leiste über dem Handschrift-Bereich werden pro geschriebenem Wort die wahrscheinlichsten Wortvorschläge in Form eines Ausklapp-Menüs angezeigt.
- ◇ Die wahrscheinlichsten Wörter sind dabei automatisch ausgewählt.
- ◇ Am linken Rand des Handschrift-Bereiches befindet sich eine ein- und ausklappbare Liste, die alle bisherigen Listeneinträge der jeweiligen Liste zeigt.
- ◇ In der rechten unteren Ecke des Handschrift-Bereiches befinden sich zwei Knöpfe. Einer dient zur Bestätigung, dass ein Listeneintrag abgeschlossen ist und vorerst kein weiterer Eintrag erfolgen soll (symbolisiert durch einen Haken). Der andere dient ebenfalls zur Bestätigung, dass ein Listeneintrag abgeschlossen ist, ermöglicht es jedoch in dieser Ansicht zu bleiben und einen weiteren Eintrag hinzuzufügen (symbolisiert durch einen Haken mit einem Plus-Zeichen).

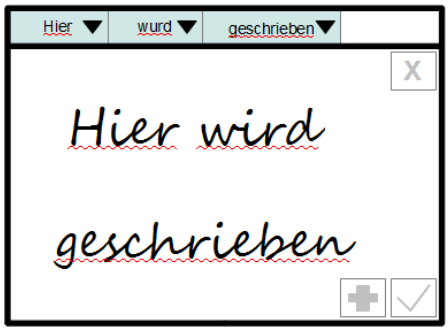


Abbildung 4: Handschriftansicht der Anwendung.

9 Qualitätszielbestimmungen

	sehr wichtig	wichtig	weniger wichtig	unwichtig
Robustheit	**X**			
Präzision	**X**			
Benutzungsfreundlichkeit		**X**		
Effizienz			**X**	
Portierbarkeit				**X**
Kompatibilität			**X**	
Erweiterbarkeit				**X**
Design			**X**	

10 Globale Testszenarien und Testfälle

Jede Produktfunktion */F????/* wird anhand von konkreten Testfällen */T????/* getestet.

10.1 Aufgabeneintrag

/T010/ *Aufgabe eintragen*: Die Testperson schreibt einen Testsatz in das Textfenster und überprüft den ausgegebenen Satz auf Fleichheit.

/T020/ *Eintrag korrigieren*: Die Testperson korrigiert einen Eintrag durch Auswahl eines anderen Textes über das Ausklappmenü.

/T030/ *Eintrag bestätigen*: Die Testperson schreibt eine Aufgabe ins Eingabefeld und bestätigt den Eintrag über die "Häkchen"-Taste. Anschließend wird in der Aufgabenliste nach dem soeben erstelltem Eintrag gesucht.

/T040/ *Eintrag bestätigen und nächsten schreiben*: Die Testperson schreibt eine Aufgabe ins Eingabefenster und bestätigt den Eintrag über die "Häkchen-Plus"-Taste. Anschließend wird der Test **T030** durchgeführt.

/T050/ *Eintrag abbrechen*: Die Testperson schreibt einen Eintrag ins Aufgabenfenster und bricht den Eintrag ab. Anschließend vergewissert sich die Testperson, dass keine neue Aufgabe in die Aufgabenliste eingetragen wurde.

/T060S/ *Aufgabenfälligkeitsdatum eintragen*: Der Tester trägt das Fälligkeitsdatum für die Aufgabe über das Kontextmenü ein und überprüft, ob die Aufgabe daraufhin mit diesem Fälligkeitsdatum versehen ist.

/T070S/ *Aufgabenliste einblenden*: Die Testperson blendet die Aufgabenliste über die Einblende-Taste an der linken Seite auf und überprüft diese auf Richtigkeit. Sie überprüft, ob auf dem Schreibfeld weiterhin geschrieben werden kann.

/T080S/ *Aufgabenliste ausblenden*: Die Testperson blendet die ausgeklappte Aufgabenliste wieder aus und schreibt eine Aufgabe in die Liste.

T/090S/ *Bildschirmtastatur einblenden*: Der Tester wählt über das Kontextmenü die Bildschirmtastatur und trägt so eine Aufgabe in die Aufgabenliste ein.

10.2 Aufgabenliste

/T100/ *Aufgabe hinzufügen*: Der Tester klickt auf das "+" Symbol um das Eingabefenster zu öffnen.

/T110S/ *Löschmodus aktivieren/deaktivieren*: Der Testnutzer aktiviert bzw. deaktiviert den Löschmodus über das zugehörige Symbol in der Menüleiste.

/T120/ *Aufgaben entfernen*: Die Testperson klickt auf das $\times$-Symbol eines Aufgaben-Elements in einer Aufgabenliste, bei eingeschaltetem/ausgeschaltetem Löschmodus und überprüft, ob die Aufgabe daraufhin noch verfügbar ist.

/T130/ *Aufgabe ändern*: Der Testnutzer klickt auf die Option zum ändern einer Aufgabe und ersetzt diese durch einen neuen Eintrag.

/T140/ *Aufgabe als "erledigt" markieren*: Die Testperson klickt in der Aufgabenliste auf das Auswahlkästchen einer Aufgabe, um sie als erledigt zu markieren.

T150 *Markierung von als "erledigt" markierten Aufgaben aufheben*: Der Testnutzer hebt die Markierung einer zuvor als "erledigt" markierten Aufgabe auf, indem er auf das entsprechende Auswahlkästchen klickt.

/T160S/ *Aufgabenliste sortieren*: Der Tester wählt die Sortierfunktion im Optionsmenü um die Aufgabenliste zu sortieren. Anschließend überprüft er ob die Aufgaben richtig sortiert sind.

/T170/ *Aufgabenliste manuell sortieren*: Der Tester sortiert die Liste neu, durch das Ziehen der gewählten Aufgaben-Elemente an gewünschte Positionen.

/T180S/ *Aufgabenliste taggen*: Die Testperson weist einer Liste einen Tag über das Kontextmenü zu.

/T190/ *Durch Listen scrollen*: Der Tester scrollt durch die Aufgabenliste.

/T200S/ *Zwischen Listen wechseln*: Der Testnutzer wischt durch die Aufgabenlisten.

/T210S/ *Fälligkeitsdatum für Aufgaben eintragen*: Der Tester ändert das Fälligkeitsdatum einer Aufgabe über das Aufgaben-Element Optionsmenü.

10.3 Übersichtsmenü

/T220/ *Aufgabenliste anlegen*: Der Testnutzer klickt die "+"-Taste, um eine neue Aufgabenliste zu erstellen. Er gibt einen Namen für die Liste und überprüft, ob sie wählbar ist.

/T230/ *Aufgabenliste öffnen*: Der Tester klickt auf eine Aufgabenliste im Auswahlmenü, um sie zu öffnen.

/T240/ *Zwischen Aufgabenlisten scrollen*: Die Testperson scrollt zwischen den Aufgabenlisten.

/T250/ *Aufgabenliste löschen*: Der Tester löscht eine Aufgabenliste über das Löschkontextmenü, bestätigt dies und überprüft, ob die Liste tatsächlich gelöscht ist.

/T260W/ *Aufgabenlisten sortieren*: Die Testperson wählt die Sortierfunktion im Optionsmenü um die Aufgabenlisten zu sortieren. Anschließend überprüft sie ob die Aufgabenlisten richtig sortiert sind.

/T270W/ *Aufgabenlisten nach Tags sortieren*: Die Testperson wählt im Kontextmenü die Filtern-Funktion, um nach gewählten Tags zu filtern. Gegebenfalls weist die Testperson mehreren Listen unterschiedliche Tags zu (**T180S**).

/T280K/ *Alle-Aufgaben-Liste*: Die Testperson öffnet die Alle-Aufgaben-Liste, überprüft sie auf Vollständigkeit und Editierbarkeit.

10.4 Optionen

/T290K/ *Aufgabenliste mit Googletasks synchronisieren*: Der Testnutzer synchronisiert seine Aufgabenlisten mit Google Tasks und überprüft online und auf dem Android Gerät, ob die Aufgaben synchron sind.

/T300W/ *Listentags verwalten*: Die Testperson erstellt neue Tags und überprüft, ob sie daraufhin aufrufbar sind. Ebenso löscht er Tags, um nachher zu überprüfen, ob sie tatsächlich gelöscht wurden.

/T310/ *Expertenmodus*: Die Testperson überprüft im Optionsmenü, ob durch Wahl des normalen Benutzers bzw. des Expertenmodus, Parameter verändert werden können.

11 Entwicklungsumgebung

11.1 Software

- Entwicklungssystem: Windows / Linux / Mac OSX
- Versionskontrolle: Git
- Programmiersprache: Java, C/C++
- Verwendete Bibliotheken: Android SDK, OpenCV, Android NDK, ESMERALDA, IAM-On-Datenbank
- IDE: Eclipse
- Diagramme:
- Dokumenterstellung: LaTeX

11.2 Hardware

- Tablet mit Android-OS ab 3.0 (API 11)

11.3 Orgware

- Internet
- Redmine
- Pflichtenheft

12 Ergänzungen

12.1 Sprachmodul

Das System verfügt über ein austauschbares Sprachmodul, indem alle Texte zum Dialog mit den Benutzern vorliegen und jeweils über eine Identifikationsnummer angesprochen werden. Um das System um eine Sprache erweitern zu können, muss das Sprachmodul mit der neuen Sprache analog zum bereits existierenen Sprachmodul erstellt werden.

13 Evaluation

Dieses Kapitel beschreibt das geplante Vorgehen bei der Evaluation der zu entwickelnden Applikation. Diese soll anhand einer repräsentativen Kontrollgruppe erfolgen, welche die Qualität der Anwendung im Rahmen einer schriftlichen Befragung beurteilt.

Ziel der Evaluation ist es insbesondere die Güte der Handschrifterkennung zu bewerten und in wiefern diese den Anforderungen des Benutzers entspricht. Ferner soll die Bedienung der Anwendung beurteilt werden.

Die Qualitätsbeurteilung soll anhand folgender Kriterien erfolgen:

1. **Erkennungsgüte**: Der geschriebene Text wird von der Anwendung weitesgehend korrekt erkannt. Einzelne falsch erkannte Wörter lassen sich leicht verbessern.
2. **Vorteile** gegenüber einer Bildschirmtastatur: Eine Eingabe durch Handschrift erleichtert die Benutzbarkeit gegenüber der Verwendung einer Bildschirmtastatur.
3. **Bedienkomfort**: Die Bedienung der Applikation sollte intuitiv verständlich sein. Der Benutzer findet sich schnell zurecht und empfindet die Bedienung als angenehm.
4. **Fehler**: Das System arbeitet überwiegend fehlerfrei. Auftretende Fehler führen nicht zum Systemabsturtz. Schwere Fehler treten nicht auf.

Die Befragung der Kontrollgruppe erfolgt mittels eines Fragebogens. Hierzu wird der Benutzer gebeten, eine beliebige Anzahl an Tasks und ToDo-Listen zu erstellen und seine Zufriedenheit anschließend in einer überreichten Checkliste zu vermerken.