Technische Universität Darmstadt
Fraunhofer-Institut für Sichere Informationstechnologie

Autorschaftsanalyse im Kontext der Attribution, Verifikation und intrinsischer Exploration

Master Thesis

ausgearbeitet von: *Oren Halvani*

Tag der Abgabe: 30. August 2012

Matrikelnummer: 1367234
Studiengang: Master Informatik

Aufgabensteller: Dr.-Ing. Martin Steinebach
Betreuer: Dr.-Ing. Huajian Liu
Prüfer: Prof. Dr. Michael Waidner

Fraunhofer SIT

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Master Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

$(Ort,\ Datum)$ $(Unterschrift)$

Danksagung

An dieser Stelle möchte ich all jenen danken, die durch ihre fachliche und persönliche Unterstützung zum Gelingen dieser Arbeit beigetragen haben. Ein ganz besonderer Dank gilt meinen beiden Betreuern Dr.-Ing. Martin Steinebach und Dr.-Ing. Huajian Liu, die mich durch ihre hilfreichen Anregungen unterstützt haben und großes Interesse an dieser Arbeit zeigten.

Weiterhin danke ich den folgenden Personen, die mich immer wieder aufs Neue motiviert und mir neue Horizonte, durch ihre „andere“ Denkweise verschafft haben:

- Christian Feier — Vielen Dank für die Hilfe beim Design der Grafiken und die unzähligen LaTeX-Tricks & Workarounds.
- Christos Votskos — Vielen Dank für die hilfreichen Gespräche, die mir öfters neue Anregungen gegeben haben.
- Christian Winter — Vielen Dank für die „aufklärenden Erkenntnisse“ zu einigen mathematischen Modellen.
- Steven Arzt — Vielen Dank für die nützlichen Optimierungstipps hinsichtlich einiger Algorithmen.

Mein ganz besonderer Dank gilt außerdem meinen Eltern, meiner Schwester Annet sowie meiner Freundin Meggie, die mir insbesondere in den vergangenen Monaten sehr viel Geduld, Verständnis und Unterstützung entgegengebracht hat.

... dedicated to my beloved son David

Zusammenfassung

In unserer heutigen Welt existieren unzählige Informationen, welche zumeist in elektronischer Form vorliegen. Ein Großteil dieser Informationen entsteht dabei im Internet und/oder wird dort vertrieben. Ein Problem, welches die meisten dieser Informationen anbelangt, ist die fehlende, anonyme oder nicht-eindeutige Autorschaft. In der Forschung haben sich dazu unterschiedliche Disziplinen etabliert, die versuchen diese Problematik entgegenzuwirken. Diese werden unter dem Begriff „Autorschaftsanalyse“ zusammengefasst und beruhen dabei zumeist auf stilometrischen Methoden, mit deren Hilfe Autorenstile automatisch durch quantifizierbare Stilmerkmale approximiert und anschließend diskriminiert werden können.

Die meisten Forschungsarbeiten in der Autorschaftsanalyse beschränken sich hauptsächlich auf die englische Sprache. Viele von ihnen konzentrieren sich zudem auf nicht zeitgenössische Domänen, wie beispielsweise Literatur aus der Renaissance (z.B. Shakespeare) oder bestimmte religiöse Werke aus dem alten und neuen Testament der Bibel. Hierbei sollte jedoch hervorgehoben werden, dass die Autorschaftsanalyse ein sprachabhängiges Problem darstellt und gleichzeitig nicht an spezifische Domänen gebunden ist. Des Weiteren existieren innerhalb der Disziplin der Autorschaftsanalyse weitere Herausforderungen, die in Forschungsarbeiten öfters nur am Rand oder überhaupt nicht betrachtet werden. Dazu zählen unter anderem das Finden von neuen und diskriminierenden Stilmerkmalen, der Umgang mit Stil-Einflüsse durch externe Faktoren (Internet/Software Lösungen wie z.B. Rechtschreibprogramme oder Synonym Datenbanken) oder das Tolerieren und Umgehen von Stil-Inkonsistenzen innerhalb von Dokumenten.

Im Rahmen dieser Arbeit werden diese (und andere) Herausforderungen aufgegriffen und diesbezüglich Lösungen präsentiert. Dazu wird unter anderem ein Framework entwickelt, welches verschiedene automatisierte Autorschaftsanalyse Verfahren bereitstellt, die teilweise auf Beschreibungen aus thematisch verwandten Forschungsarbeiten basieren. Die entwickelten Verfahren, die sich in die drei Unterdisziplinen Autorschafts-Attribution, Autorschafts-Verifikation und die intrinsische Exploration einordnen lassen, benötigen vom Benutzer nur eine gewählte Parametrisierung. Die Analyse selbst erfolgt anschließend automatisch und interaktionsfrei. Des Weiteren werden einige theoretische Ansätze für die Autorschaftsanalyse vorgestellt, die mit dem heutigen Stand der natürlichen Sprachverarbeitung noch nicht umsetzbar sind, jedoch Anregung für zukünftige Forschungsarbeiten geben sollen. Zum Abschluss dieser Arbeit werden schließlich mehrere Experimente durchgeführt deren Zielsetzung ist, die implementierten Verfahren hinsichtlich ihrer Praxistauglichkeit bewerten zu können. Dazu werden eigens kompilierte Korpora vorgestellt, die sich sowohl von ihren (gegenwärtigen) Domänen als auch durch deren sprachlichen Varietäten voneinander unterscheiden.

Abstract

In our modern world, there is an abundance of information which is mostly stored electronically. The largest part of it originates from the Internet or is at least distributed online. However, for most of these documents the respective author is not known. The document may be anonymous or authorship information may be missing or ambiguous. In order to solve this problem, two different research disciplines have been established which are largely based on stylometric techniques and are summarized under the term authorship analysis. The goal is to automatically approximate the styles of different authors and then use this information for classifying texts.

Research in authorship analysis usually focuses on the English language. Additionally, much of the work does not concentrate on contemporary writings, but rather on e.g. Renaissance literature like Shakespeare writings or various religious works from the Old and New Testament. It is important to emphasize that authorship analysis is a language-dependent problem, but it is not bound to specific domains. Furthermore, there are additional challenges in the discipline of authorship analysis which are usually not directly or even not at all taken into consideration in research projects. These challenges include finding new and characteristic style features, handling external influences on an author's style (Internet and software solutions like spell checkers or synonym databases), and tolerating and circumventing stylistic inconsistencies in documents.

This thesis addresses the challenges discussed above (and other challenges) and presents solutions for them. We present a framework implementing a variety of techniques for automatic authorship analysis. Some of these techniques are based on theoretical descriptions from related fields of research. In general, the techniques can be classified into the three categories authorship attribution, authorship verification, and intrinsic exploration. For all methods, the user only needs to provide a selected set of parameters. The analysis itself is fully automatic and runs without interaction. Additionally, we present some theoretical approaches for authorship analysis which cannot be implemented given the current state of research in natural language processing, but nevertheless provide a direction for future research. Finally, we conduct several experiments in which the implemented methods are tested for suitability in realistic scenarios. We introduce specifically compiled corpora that differ from present domains as well as from each other in terms of language variability.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen Welt existieren unzählige Informationen, die entweder in elektronischer oder nicht-elektronischer Form vorliegen. Erstere werden dabei vor allem durch das Internet verbreitet oder entstehen dort. Die meisten dieser Informationen liegen in textueller Form vor, wobei ihre Quellen unterschiedlicher Natur sind. Dazu zählen unter anderem E-Mails, Weblogs, Foren, Wikis oder auch Kommentierungen und Nachrichten in sozialen Netzwerken.

Des Öfteren liegt der Fall vor, dass die Autorschaft dieser Texte nicht explizit gegeben ist, jedoch für viele Szenarien zwingend vorausgesetzt wird. Derartige Szenarien erstrecken sich über viele Gebiete bzw. Wissenschaften, wie beispielsweise Literaturwissenschaft, Rechtswissenschaft, Forensik oder aber auch Lehre und Forschung. Die Klärung der Autorschaft hat je nach Kontext unterschiedliche Ausprägungen. So können beispielsweise in der Lehre durch die Klärung der Autorschaft unter anderem Plagiatsvergehen aufgedeckt werden. In der Forensik dagegen, ist man vor allem an der Autorschaft brisanter Dokumente interessiert. Dazu zählen unter anderem Konversationen, Protokolle oder Mitschnitte aus den unterschiedlichsten Kommunikationsquellen (SMS, E-Mail, Instant-Messaging-Software, etc.). Neben Forensikern sind Juristen und Gutachter ebenfalls daran interessiert rechtsrelevante Dokumente hinsichtlich ihrer wahren Autorschaft zu überprüfen, um daraus verwertbare Indizien für Gerichtsprozesse zu gewinnen. Dazu zählen wiederum Geständnisse, Testamente, Erpresserbriefe oder gar Bekennerschreiben von Terroristen.

Alle diese Szenarien haben mindestens zwei Gemeinsamkeiten. Die erste ist, dass die Autorschaft der Texte auf Metadaten beruht, zu denen beispielsweise Dateinamen, Unterschriften oder auch eingebettete Dokumenteigenschaften (Benutzernamen, Titel, Organization, Schlüsselwörter, etc.) zählen. Derartige Metadaten bergen das Problem, dass sie entweder nicht vorliegen oder sich manipulieren lassen. Die zweite Gemeinsamkeit ist, dass die Analyse der Dokumente in der Regel händisch durch einen Experten durchgeführt werden muss, wodurch sehr viel Zeit für die Bearbeitung der Fälle beansprucht wird. Um dieser Problematik entgegenzuwirken, haben Forscher im Laufe der Jahre unterschiedliche Disziplinen konzipiert, die sich mit der Untersuchung fehlender oder vermeintlicher Autorschaften von Texten beschäftigen. In der Literatur werden diese Disziplinen unter dem Oberbegriff Autorschaftsanalyse zusammengefasst. Die meisten Unterdisziplinen der Autorschaftsanalyse verbindet oftmals der gleiche technische Kern, der den Einsatz stilometrischer Methoden einschließt. Bei diesen Methoden werden Stilmerkmale quantifiziert, um dadurch wiederum Autorenstile approximieren zu können. Dabei werden Stilmerkmale unmittelbar aus den Texten gewonnen, sodass hier keine Abhängigkeit bzgl. zwiespältiger Metadaten besteht. Innerhalb der Autorschaftsanalyse dominieren vor allem zwei Disziplinen: Die Autorschafts-Attribution sowie die Autorschafts-Verifikation. Die Erstere versucht, hinsichtlich eines gegebenen anonymen Textes, den wahrscheinlichsten Autor innerhalb einer bestehenden Trainingsmenge (die Beispieltexte von mehreren Autoren enthält) zu ermitteln. Die Autorschafts-Verifikation versucht dagegen, anhand eines Textes von einem angeblichen Autor $\mathcal{A}$ urteilen zu können, ob $\mathcal{A}$ tatsächlich als der wahre Autor des Textes betrachtet werden kann oder nicht. Die Autorschafts-Verifikation stellt ein Spezialfall der Autorschafts-Attribution dar, bei der die Trainingsmenge nur Dokumente eines einzigen Autors enthält.

In den letzten Jahrzehnten haben sich in der Forschung, vor allem in der Disziplin der Autorschafts-Attribution, zunehmend automatisierte Verfahren etabliert, um Experten bei der Bewältigung manueller Analysen zu assistieren. Diese spalten sich in instanz- und profilbasierten Verfahren auf. Beim Ersteren wird jeder Beispieltext innerhalb der Trainingsmenge einzeln als eine stilistische Instanz eines Autors betrachtet. Bei profilbasierten Verfahren werden dagegen alle Beispieltexte eines Autors zu einem großen Dokument (Autor-Profil) zusammengeführt. Dieser kann als eine Art *Stil-Repertoire* betrachtet werden, auf dem ein Autor zurückgreift, sobald dieser einen neuen Text verfasst. Aus den einzelnen Beispieltexten (bzw. Autor-Profilen), als auch aus dem anonymen Dokument, werden anschließend Stilmerkmale entnommen, um dadurch sogenannte Feature-Vektoren zu generieren. Ein

Feature-Vektor repräsentiert somit den Autorenstil eines Dokuments bzw. Autor-Profils. Die Attribution hinsichtlich des wahrscheinlichsten Autors erfolgt bei beiden Verfahren anhand eines Attributions-Modells. Bei diesem kann es sich wahlweise um einen Klassifikator oder eine Metrik (oder eine Kombination aus beiden) handeln. Ein Klassifikator stellt hierbei ein Verfahren des maschinellen Lernens dar, welches die Aufgabe hat aus gegebenen bereits klassifizierten Daten (die wiederum als Feature-Vektoren vorliegen) neue unbekannte Daten zu klassifizieren. Bei einer Metrik handelt es sich dagegen, um eine einfache Distanz- oder Ähnlichkeitsfunktion, mit der Abweichungen bzw. Ähnlichkeiten zwischen je zwei Feature-Vektoren bestimmt werden können.

Beide Ansätze haben ihre Vor- und Nachteile. So sagt die Literatur instanzbasierten Verfahren beispielsweise nach, leicht mit der Kombinierung verschiedener Stilmerkmale umgehen zu können, was bei profilbasierten Verfahren nur schwer zu realisieren ist. Dagegen ist die Trainingsphase (das Erlernen eines Stil-Modells) bei instanzbasierten Verfahren sehr hoch, während diese bei der profilbasierten Variante praktisch nicht existiert, [99, Seite: 20].

Trotz zahlreicher Forschungsarbeiten innerhalb der Autorschaftsanalyse sind bis heute noch viele Fragen offen. Es konnte beispielsweise nicht genau geklärt werden, ob und in welchem Umfang das Register sowie Genre eines Textes Einfluß auf die Analyse-Verfahren haben (sowohl für die Attribution als auch Verifikation). Weiterhin ist die Frage offen, wie groß ein Text mindestens sein sollte, um eine erfolgreiche Analyse durchzuführen. Auch die Frage wie Verfahren mit Stil-Inkonsistenzen innerhalb der Texte umgehen können, wurde in der Literatur nicht beantwortet. Mit anderen Worten, können Texte, die unterscheidliche Stile von einem Autoren beinhalten, denoch bzgl. ihrer Autorschaft zugeordnet werden? Eine andere Fragestellung, die bisher in der Forschung kaum Beachtung gefunden hat, ist die, ob und wie eine multiple Autorschafts-Attribution ermöglicht werden kann. Diese und andere Fragestellungen werden innerhalb dieser Arbeit aufgegriffen und diesbezüglich (teilweise) Lösungen präsentiert.

1.2 Zielsetzung

Die vorliegende Arbeit hat unterschiedliche Zielsetzungen, wobei das primäre Ziel die Entwicklung eines Autorschaftsanalyse-Systems mit Deutsch als Ausgangssprache darstellt. Dabei sollen Verfahren für die Autorschafts-Attribution, Autorschafts-Verifikation und die intrinsische Exploration prototypisch implementiert und in einer Experimentierreihe evaluiert werden.

Weiterhin soll die Basis der Autorschaftsanalyse genauer erkundet werden, welche wiederum die Stilmerkmale darstellen. Hierbei sollen bestehende, als auch neue Stilmerkmale beschrieben, kategorisiert, angewendet und schließlich bewertet werden. Vom besonderen Interessse ist dabei die Praxistauglichkeit der jeweiligen Stilmerkmale. Diese sollen vor allem in der Lage sein domänenübergreifend verlässliche Erkennungsgenauigkeiten liefern zu können, wobei der Fokus hier auf Autorschafts-Attributions Verfahren liegen soll. Darüber hinaus soll erläutert werden, welche Schritte notwendig sind, um Stilmerkmale aus Textdokumenten extrahieren zu können. Hierfür sollen Natural Language Processing Werkzeuge eingesetzt werden.

Neben der Implementierung des Frameworks sollen zudem die wichtigsten Herausforderungen der Autorschaftsanalyse detailiert beschrieben und (zumindest für ein Teil davon) Lösungen erarbeitet bzw. vorgeschlagen werden. Hierbei sollen auch theoretische Ansätze diskutiert werden, die mit dem heutigen Stand der Technik nicht bzw. nicht im vollem Umfang umgesetzt werden können. Das Ziel der theoretischen Ansätze soll es sein Anregungen für zukünftige Forschungsarbeiten zu geben.

1.3 Aufbau der Arbeit

Zu Beginn der Arbeit werden zunächst allgemeine Grundlagen erläutert. Darin werden insbesondere eine formale Notation und linguistische Begriffe eingeführt, die innerhalb der gesamten Arbeit verwendet werden. Im nächsten Grundlagenkapitel werden computerlinguistische Werkzeuge beschrieben,

mit deren Hilfe Stilmerkmale aus den sprachlichen Ebenen der Textes extrahiert werden können. Das letzte Grundlagenkapitel umfasst Begriffe und Konzepte des maschinellen Lernens. Der Kern dieses Kapitels befasst sich mit der Beschreibung von Metriken, Klassifikatoren und Clustering-Verfahren, die die Basis einiger Autorschaftsanalyse-Verfahren innerhalb dieser Arbeit darstellen. Darüber hinaus werden in diesem Kapitel Evaluierungsmöglichkeiten vorgestellt, mit deren Hilfe die implementierten Verfahren hinsichtlich ihrer Güte bewertet werden können.

Im fünften Kapitel wird detailiert auf die Thematik der Features eingegangen, die den wichtigsten Bestandteil der Autorschaftsanalyse darstellen. Der Kern dieses Kapitels befasst sich mit dem Prozess der Feature-Vektor Generierung. Hierfür wird der umfangreiche Prozess in Teilschritte aufgeteilt, wobei jeder Schritt in einem Unterkapitel genauer erläutert wird.

Das sechste Kapitel führt in die Autorschaftsanalyse ein. Hierbei werden zunächst generelle Modelle für Autorschaftsanalyse-Verfahren beschrieben und schematisch dargestellt, wobei für jede Unterdisziplin, die benötigten Komponenten ausführlich erläutert werden. Des Weiteren werden die wichtigsten Herausforderungen für die Autorschaftsanalyse im Detail beschrieben. Zum Ende des Kapitels werden schließlich einige Anwendungsgebiete für die Autorschaftsanalyse-Verfahren genannt.

Im siebten Kapitel werden die eigenen Verfahren vorgestellt, die im Rahmen des Frameworks implementiert wurden. Des Weiteren werden mehrere theoretische Ansätze vorgeschlagen, deren Umsetzung aufgrund technischer Machbarkeit (noch) nicht realisierbar ist.

Im achten Kapitel wird eine kurze Einführung in das Framework gegeben. Dabei werden die einzelnen Komponenten des Frameworks hinsichtlich ihrer Aufgaben zusammengefasst, wobei hier auch externe Werkzeuge genannt werden, ohne die die Realisierung des Frameworks nicht möglich gewesen wäre.

Im neunten Kapitel werden sämtliche Korpora vorgestellt, die neben den Features und den implementierten Verfahren, die Basis der Experimente darstellen. Hierbei werden unter anderem Statistiken für die Korpora aufgeführt, um dadurch deren innere Struktur wiederzuspiegeln.

Im letzten Kapitel wird schließlich eine umfangreiche Experimentierreihe durchgeführt. Der Fokus liegt hierbei vor allem auf die Verfahren für die Autorschafts-Attribution, wobei jedoch auch Experimente hinsichtlich der Autorschafts-Verifikation und intrinsischen Exploration durchgeführt werden. Im Anschluß an die Experimente werden schließlich die Erkenntnisse zusammengefasst und des Weiteren ein Ausblick gegeben, der unter anderem beschreibt, wie die Verfahren erweitert werden können.

2 Allgemeine Grundlagen

In diesem Kapitel werden die grundlegenden Begrifflichkeiten und Konzepte der Autorschaftsanalyse definiert und erläutert. Das Ziel dabei ist ein Fundament an Vokabular zu schaffen, auf dem sich spätere Kapitel stützen können. Dazu wird zunächst eine symbolische Notation eingeführt, die innerhalb dieser Arbeit an zahlreichen Stellen wiederverwendet wird. Anschließend werden linguistische Begriffe definiert, die vor allem für das Verständnis der Kapitel 3 und 5 benötigt werden. In den nächsten Abschnitten werden neben der Begriffe Stil und Stilmerkmale, die sogenannten „sprachlichen Ebenen“ beschrieben, aus denen Stilmerkmale entnommen werden können, um dadurch Autorenstile zu approximieren. Der letzte Abschnitt dieses Kapitels beschäftigt sich mit der Thematik der Text Normalisierung, deren Zielsetzung es ist, Texte auf eine einheitliche Form zu bringen, um diese später weiterverarbeiten zu können.

2.1 Notation

Im Verlauf dieser Arbeit werden zahlreiche Begriffe des Öfteren in unterschiedlichen Kontexten wiederverwendet. Aus diesem Grunde wurde entschieden die häufigsten Begriffe zu formalisieren, um dadurch zum einen Wortwiederholungen zu vermeiden und zum anderen sprachliche Mehrdeutigkeiten aufzulösen. Die folgende Notation wird in Abbildungen, Pseudocode-Beispiele, Definitionen oder Tabellen wiederverwendet.

Zahlenmengen:

- $\mathbb{B}$ beschreibt die Menge der binären Zahlen.
- $\mathbb{N}$ beschreibt die Menge der natürlichen Zahlen.
- $\mathbb{R}$ beschreibt die Menge der reelen Zahlen.

Autoren:

- $\mathcal{A}$ bezeichnet einen bekannten Autor.
- ε bezeichnet einen unbekannten Autor.
- $\mathbb{A} = \{ \mathcal{A}_1, \mathcal{A}_2, \dots \}$ bezeichnet eine Menge von (bekannten) Autoren.

Dokumente:

- $\mathcal{D}$ bezeichnet allgemein ein Dokument (mit oder ohne einer bekannten Autorschaft).
- $\mathcal{D}_{\mathcal{A}}$ bezeichnet ein Dokument, welches von einem Autor $\mathcal{A}$ verfasst wurde. Wird Bezug auf ein spezifisches Dokument $\mathcal{D}_i$ eines Autors $\mathcal{A}_j$ genommen, so wird die Notation $\mathcal{D}_{i\mathcal{A}_j}$ benutzt.
- $\mathcal{D}_{\varepsilon}$ beschreibt ein Dokument, dessen Autorschaft unbekannt ist. Anders ausgedrückt: ε ist der Autor von $\mathcal{D}$.
- $\mathbb{D}_{\mathcal{A}} = \{ \mathcal{D}_{1\mathcal{A}}, \mathcal{D}_{2\mathcal{A}}, \dots \}$ beschreibt die Menge aller Dokumente die von einem Autor $\mathcal{A}$ verfasst wurden. Wird Bezug auf einen spezifischen Autor $\mathcal{A}_j$ genommen, so wird die Notation $\mathbb{D}_{\mathcal{A}j}$ benutzt.
- $\mathbb{D}$ beschreibt die Menge aller Dokumente von allen Autoren. Es gilt hierbei:

$$\mathbb{D} = \bigcup_{\mathcal{A} \in \mathbb{A}} \mathbb{D}_{\mathcal{A}} = \{ \mathcal{D}_{1\mathcal{A}_1}, \mathcal{D}_{2\mathcal{A}_1}, \dots, \mathcal{D}_{1\mathcal{A}_2}, \mathcal{D}_{2\mathcal{A}_2}, \dots, \mathcal{D}_{r-1\mathcal{A}_m}, \mathcal{D}_{r\mathcal{A}_m} \}, \text{ mit } m = |\mathbb{A}| \text{ und } r > m$$

- $\mathcal{D}_{BIG\mathcal{A}}$ beschreibt ein Autor-Profil mit $\mathcal{A}$ als dessen Autor. Ein Autor-Profil stellt eine Konkatenation aller $\mathcal{D}_{i\mathcal{A}} \in \mathbb{D}_{\mathcal{A}}$ dar. Es gilt somit $\mathcal{D}_{BIG\mathcal{A}_j} = \mathcal{D}_{1\mathcal{A}} \circ \mathcal{D}_{2\mathcal{A}} \circ \dots \circ \mathcal{D}_{\ell\mathcal{A}_j}$ mit $\ell = |\mathbb{D}_{\mathcal{A}}|$.

Korpora:

- $\mathcal{K}$ bezeichnet einen Korpus und damit eine Kollektion von Dokumenten. In den meisten Fällen kann $\mathcal{K}$ mit $\mathbb{D}$ gleichgesetzt werden.

Dokument-Elemente:

- ω beschreibt ein Wort innerhalb eines Dokuments, sodass $\omega \in \mathcal{D}$ gilt.
- σ beschreibt ein Satz innerhalb eines Dokuments, sodass $\sigma \in \mathcal{D}$ gilt.
- ξ beschreibt ein Slice (Text-Fragment) innerhalb eines Dokuments, sodass $\xi \in \mathcal{D}$ gilt.

Features:

- f bezeichnet ein einzelnes Feature (Stilmerkmal) wie beispielsweise $f_1 =$ `Wörter mit mindestens drei und höchstens sieben Vokalen`.
- $f(\mathcal{D})$ bezeichnet eine relative Häufigkeit, die aus der Anwendung eines Features f auf ein Dokument $\mathcal{D}$ resultiert. Es gilt $f(\mathcal{D}) \in (\mathbb{R}^+ \cup \{0\})$.
- $F = \{f_1, f_2, \dots\}$ bezeichnet eine Kategorie von Features die semantisch zusammengehören, wie etwa Wortarten: $f_1 =$ `Adjektive`, $f_2 =$ `Verben`, $f_3 =$ `Nomen`, ...
- $\mathbb{F} = F_1 \cup F_2 \cup \dots$ bezeichnet einen Feature-Raum, welcher die Gesamtheit aller Features[1] darstellt.
- $\mathcal{F}_{i\mathcal{A}_j} = (f_1(\mathcal{D}_{1\mathcal{A}_j}), f_2(\mathcal{D}_{1\mathcal{A}_j}), \dots, f_n(\mathcal{D}_{1\mathcal{A}_j}))$ bezeichnet ein Feature-Vektor, dessen n Elemente Features darstellen, die auf $\mathcal{D}_{1\mathcal{A}_j}$ angewendet wurden. Feature-Vektoren repräsentieren innerhalb dieser Arbeit Autorenstile und sind daher von zentraler Bedeutung.
- $\mathcal{I}(f_k) = (f_1(\mathcal{D}_{\mathcal{A}1}), f_1(\mathcal{D}_{\mathcal{A}2}), \dots, f_1(\mathcal{D}_{\mathcal{A}m}))$ bezeichnet ein Feature-Instanz Vektor, dessen Elemente die Anwendung eines Features f_1 auf $\mathbb{D}$ repräsentieren.
- F_{matrix} bezeichnet eine Feature-Matrix für einen beliebigen Korpus $\mathcal{K}$, die sich aus $r = |\mathbb{D}|$ Zeilen und $n \in \{1, 2, \dots, |\mathbb{F}|\}$ Spalten zusammensetzt. Schematisch kann F_{matrix} wie folgt dargestellt werden:

	f_1	f_2	...	f_k	...	f_n
	$\mathcal{I}(f_1)$	$\mathcal{I}(f_2)$	...	$\mathcal{I}(f_k)$	...	$\mathcal{I}(f_n)$
$\mathcal{F}_{1\mathcal{A}_1}$	...	...	...	$f_k(\mathcal{D}_{1\mathcal{A}_1})$	...	...
$\mathcal{F}_{2\mathcal{A}_1}$	...	...	...	$f_k(\mathcal{D}_{2\mathcal{A}_1})$	...	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathcal{F}_{1\mathcal{A}_2}$	...	...	...	$f_k(\mathcal{D}_{1\mathcal{A}_2})$	...	...
$\mathcal{F}_{2\mathcal{A}_2}$	$f_1(\mathcal{D}_{2\mathcal{A}_2})$	$f_2(\mathcal{D}_{2\mathcal{A}_2})$	...	$f_k(\mathcal{D}_{2\mathcal{A}_2})$	...	$f_n(\mathcal{D}_{2\mathcal{A}_2})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathcal{F}_{1\mathcal{A}_m}$	...	...	...	$f_k(\mathcal{D}_{1\mathcal{A}_m})$	...	...
$\mathcal{F}_{2\mathcal{A}_m}$	...	...	...	$f_k(\mathcal{D}_{2\mathcal{A}_m})$	...	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\mathcal{F}_{r\mathcal{A}_m}$	...	...	...	$f_k(\mathcal{D}_{r\mathcal{A}_m})$	...	...

Trainingsmenge/Testmenge:

- $\mathbb{D}_{train}$ bezeichnet eine Dokumenten-Trainingsmenge.
- $\mathbb{D}_{test}$ bezeichnet eine Dokumenten-Testmenge.
- **Hinweis:** Da beide Mengen vom jeweiligen Kontext abhängen bleiben sie an dieser Stelle unterspezifiziert.

[1] Damit sind sämtliche Features gemeint, die in dieser Arbeit zur Verfügung stehen.

Funktionen:

- $\boxed{\textit{Action}} \rightsquigarrow \textit{Object}$ bezeichnet die Anwendung einer Aktion *Action* auf einem Objekt *Object*. Dabei kann es sich bei *Action* beispielsweise um ein Algorithmus, Programm oder ein System handeln, während *Object* ein Textdokument, Textfragment, Satz, Wort oder Ähnliches sein kann.
- $\textit{length}(\textit{Object})$ beschreibt eine Funktion, welche die (Zeichen-)Länge eines Objekts *Object* ermittelt.
- $\#(o, \textit{Object})$ beschreibt eine Funktion, die die Anzahl aller Elemente o innerhalb *Object* liefert. Mit anderen Worten: $\#(o, \textit{Object})$ repräsentiert die absolute Häufigkeit.
- Die Funktion $\text{Mean}(X)$ beschreibt das arithmetische Mittel (bzw. den Durchschnitt) aller n Messwerte einer Folge $X \in \mathbb{R}^n$. Es gilt:

$$\text{Mean}(X) = \frac{1}{n} \sum_{i=1}^{n} X_i$$

- Der Median bezeichnet ein statistisches Lagemaß, welches eine (geordnete) Folge $X \in \mathbb{R}^n$ mit n Messwerten in zwei Hälften aufteilt. $\text{Median}(X)$ beschreibt dabei diejenige Zahl, die sich zwischen beiden Hälften befindet, dabei gilt:

$$\text{Median}(X) = \begin{cases} X_{\frac{n+1}{2}}, & \text{falls } n \text{ ungerade,} \\ \frac{1}{2}\Big(X_{\frac{n}{2}} + X_{\frac{n}{2}+1}\Big), & \text{falls } n \text{ gerade.} \end{cases}$$

- Die Funktion $\text{Variance}(X)$ beschreibt die Varianz einer Folge $X \in \mathbb{R}^n$. Es gilt:

$$\text{Variance}(X) = \frac{1}{n} \sum_{i=1}^{n} \Big(X_i - \text{Mean}(X)\Big)^2$$

- Die Funktion $\text{MedVariance}(X)$ beschreibt eine abgewandelte Form der Varianz einer Folge $X \in \mathbb{R}^n$, bei der anstelle des arithmetischen Mittels der Median verwendet wird. Es gilt:

$$\text{MedVariance}(X) = \frac{1}{n} \sum_{i=1}^{n} \Big(X_i - \text{Median}(X)\Big)^2$$

  Hinweis: Innerhalb dieser Arbeit wurde diese Form der Varianz verwendet da sich der Median, Ausreißern gegenüber, robuster verhält als das arithmetische Mittel.

2.2 Linguistische Begriffe

In diesem Abschnitt werden die wichtigsten linguistischen Begriffe eingeführt, die innerhalb dieser Arbeit an zahlreichen Stellen Verwendung finden werden. Die Reihenfolge der Begriffe entspricht dabei dem logischen Aufbau einer Sprache: Von den kleinsten atomaren sprachlichen Einheiten bis hin zum Text als Ganzes.

Definition 2.1. ***Graphem:*** *Unter einem Graphem versteht man die kleinste unterscheidbare Einheit der schriftlichen Ebene, [55].*

Definition 2.2. ***Morph:*** *Ein Morph beschreibt eine Sequenz von Grapheme, die wiederum ein Morphem in einen bestimmten Kontext repräsentieren, [82].*

Definition 2.3. ***Morphem:*** *Ein Morphem beschreibt die kleinste (nicht weiter zerlegbare) bedeutungstragende Einheit einer Sprache, [82].*

Beispiel: Sei die Wortform ω gegeben durch *Unsichtbarkeit*. Dann hat ω die folgenden Morpheme:

Wortbildungsmorphem	Stamm-Morphem	Wortbildungsmorphem	Wortbildungsmorphem
Un	sicht	bar	keit

Definition 2.4. ***Token:*** *Unter dem Begriff Token wird eine Konkatention (Aneinanderreihung) von Zeichen und/oder Symbole verstanden, die als eine Einheit fungieren. Getrennt werden Tokens in der Regel durch entsprechende Seperatoren (in indoeuropäische Sprachen[1] stellen für gewöhnlich Leerzeichen derartige Seperatoren dar). Die Symbole die eine Einheit bilden liegen dabei ein endliches Alphabeth zugrunde, welches neben rein alphabetischen Lettern auch Ziffern oder Sonderzeichen beinhalten können[2].*

Zur besseren Verdeutlichung zeigt das folgende Beispiel einige Tokens:

`SOS!!!, neo78XD, „Kommunikation", 5-Tage-Woche, :-)`

Zusammengefasst stellt ein Token die abstrakte Form eines Wortes dar. Das bedeutet ein Wort ist immer ein Token, die Umkehrung jedoch gilt nicht im Allgemeinen.

Definition 2.5. ***Type:*** *Unter dem Begriff Type versteht man eine abstrakte Klasse zu denen ein Token angehört. Types können somit als unterscheidbare Tokens verstanden werden, [88].*

Das folgende Beispiel verdeutlicht den Begriff der Types. Sei dazu ein Satz σ gegeben durch:

$\sigma =$ `Wenn Fliegen hinter Fliegen fliegen, fliegen Fliegen Fliegen nach.`

Die Tokens[3] bzgl. σ lauten damit:

`Wenn, Fliegen, hinter, Fliegen, fliegen, fliegen, Fliegen, Fliegen, nach`

Die Types bzgl. der Tokens lauten damit wiederum: `wenn, fliegen, hinter, nach`. Hieraus wird ersichtlich, dass Types unabhängig von der Groß/Kleinschreibung sind und keine mehrfach vorkommente Tokens (also Duplikate) darstellen.

Definition 2.6. ***Wort:*** *Ein Wort beschreibt eine sprachliche Grundeinheit, die je nach sprachlicher Ebene unterschiedliche charakteristische Eigenschaften vorweist. Ein Wort kann damit unterschiedlich definiert werden, beispielsweise: graphisches Wort, morphologisches Wort, syntaktisches Wort, lexikalisches Wort, etc., [55].*

Definition 2.7. ***Wortform:*** *Unter einer Wortform wird eine konkrete Realisierung eines Wortes, im Kontext eines Satzes verstanden, [55].*

Definition 2.8. ***Lexem:*** *Ein Lexem beschreibt den Grundbestandteil eines Wortes. Damit kann ein Lexem lexikalisch äquivalente Wortformen repräsentieren, [82].*

Sei beispielsweise das Lexem ω gegeben durch `schreiben`. Dann kann ω die folgenden Wortformen aufweisen:

$\omega = \{$ `schreibte, schrieb, geschrieben, verschrieben,` $\ldots\}$

[1] Zu den indoeuropäischen Sprachen zählen unter anderem: Deutsch, Englisch, Französisch, Spanisch, etc.

[2] Dies ist gleichzeitig die wesentliche Unterscheidung zwischen Tokens und Wörtern.

[3] Hierbei wurden Interpunktionszeichen nicht berücksichtigt.

Definition 2.9. ***Konstituente:*** *Eine Konstituente stellt eine syntaktische Gliederungseinheit eines Satzes dar, [55].*

Das folgende Beispiel verdeutlicht den Begriff der Konstituente. Sei ein Satz σ wie folgt gegeben:

$$\sigma = \texttt{Der Hund liegt gerne auf der Couch}$$

Dann hat σ die folgenden vier Konstituenten, wobei σ selbst auch eine Konstituente darstellt:

$$(\,(\,\texttt{Der Hund}\,),\,(\,\texttt{liegt}\,),\,(\,\texttt{gerne}\,),\,(\,\texttt{auf der Couch}\,)\,)$$

Definition 2.10. ***Phrase:*** *Eine Phrase bezeichnet Konstituenten, die in bestimmten syntaktischen Hinsichten eine Einheit bilden und dadurch eine gewisse Selbständigkeit aufweisen, [55].*

Definition 2.11. ***Satz:*** *Ein Satz beschreibt eine abgeschlossene sprachliche Einheit, die aus mindestens einem Wort besteht.*

Definition 2.12. ***Text:*** *Als Text wird eine (relativ) abgeschlossene schriftliche Äußerung verstanden, die im Allgemeinen aus einer grammatisch und inhaltlich zusammenhängenden Folge von Sätzen besteht, [55].*

Definition 2.13. ***Textsorte:*** *Eine Textsorte*[1] *bezeichnet eine Klasse oder Menge von Texten, die einem prototypischen Textmuster folgen, [63].*

Zu den wissenschaftlichen Textsorten zählen beispielsweise: Monographien, Protokolle, Bibliographien, Disertationen, Dokumentationen oder auch Enzyklopädien.

Definition 2.14. ***Sprachregister:*** *Ein Sprachregister (kurz Register) bestimmt die Form eines Textes. Darunter kann eine von der Standardsprache abweichende Sprachverwendung verstanden werden, die von situationsbezogenen Faktoren abhängt, [43, Seite: 48].*

Zu solchen Faktoren zählen beispielsweise:

- *„Kanalfaktoren (z.B. Telefon, E-Mail Internet, Telegramm)“* [43, Seite: 48],
- *„soziale Rollen (z.B. Höflichkeitsformulierungen im Deutschen)“* [43, Seite: 48],
- *„reglementierte Formulierungen (z.B. im juristischen Kontext, Lehrbücher)“* [43, Seite: 48].

Die folgende Tabelle (adaptiert aus [34]) zeigt einige typische Register im Deutschen:

[1] In der Literatur ist dieser Begriff auch bekannt unter den folgenden Bezeichnungen: Textart, Stilart, Darstellungsart, Darstellungsform, Textklasse, Texttyp, etc.

Register	Erläuterung/Beispiele
Geschriebene Sprache:	Geprägt duch Verwendung von Wörtern aus dem gehobenen und formellen Sprachgebrauch, z.B. `Diskrepanz, erörtern.` [34]
Veraltet:	Ausdrücke, die im Deutschen nicht verwendet jedoch (noch) verstanden werden, z.B. `Abort (Toilette)`. [34]
Veraltend:	Ausdrücke, die meistens von älteren Generationen verwendet werden, z.B. `Leibesübungen (Sport)`. [34]
Formell:	Ausdrücke, die in gesprochener und geschriebener Sprache verwendet werden und zum förmlichen Sprachgebrauch gehören, z.B. `Bewirtung, wohnhaft sein`. [34]
Neutral:	Ausdrücke oder Begriffe, die keinem Register angehören und einem neutralen Sprachgebrauch zugeordnet werden können. [34]
Umgangssprachlich/salopp:	Ausdrücke, die öfters zwischen Freunden oder Familienmitgliedern in unförmlichen Unterhaltungen bzw. privaten Schriftverkehr verwendet werden, z.B. `aufgemschmissen sein, jemanden übers Ohr hauen, Bruchbude`. [34]
Abwertend/beleidigend:	Ausdrücke, die allgmein als vulgär, bleidigend und abwertend empfunden werden, z.B. `Fresse halten, krepieren`. [34]
Ironisch/humorvoll	Ausdrücke, die scherzhaft oder ironisch verwendet werden, z.B. `Eierkopf, Gequassel`. [34]
Fachsprachlich:	Ausdrücke, die einem bestimmten Fachbereich (z.B. Medizin, Chemie, Mathematik, etc.) angehören und von Außenstehenden in der Regel nicht verwendet werden, z.B. Tympanum (Mittelohr), Destillation (thermisches Trennverfahren), differenzieren (Ableitung bilden). [34]

Tabelle 1: Register im Deutschen (adaptiert aus [34]).

Definition 2.15. ***Korpus:*** *Der Begriff Korpus (plural „Korpora“) beschreibt eine Sammlung von schriftlichen (oder auch gesprochenen) Äußerungen in mindestens einer Sprache. Zu den Bestandteilen eines Korpus zählen neben den eigentlichen Daten auch Metadaten, bzw. linguistische Annotationen, [62, Seite: 40].*

Anmerkung: Die Korpora die innerhalb dieser Arbeit erstellt wurden, enthalten Dokumente aus unterschiedlichen Textsorten und Registern.

2.3 Sprachliche Ebenen

Daten, wie beispielsweise Bilder, Audiodateien oder Videos sind aus verschiedenen Ebenen zusammengesetzt, wobei „Ebenen“ je nach Kontext unterschiedlich interpretiert werden können. In Bildbearbeitungsprogrammen beispielsweise werden Ebenen dazu verwendet um Rahmen, Textfelder, Verläufe, etc. als eine Art Container darzustellen. Das fertige Bild resultiert dann aus einer Zusammenfügung aller Ebenen. Bei Audio- bzw. Videobearbeitungsprogrammen dagegen, können Ebenen dazu verwendet werden, um bestimmte Effekte wie z.B. fließende Melodieübergänge oder Rotoscoping (Zeichnen von Bilderfolgen) zu realisieren. Analog dazu enthalten textuelle Daten ebenfalls derartige Ebenen. Hierbei werden diese zunächst nach [**?**] mit dem Begriff „Sprachliche Ebenen[1]“ bezeichnet und gliedern sich wie folgt auf:

Sprachliche Ebene:	**Definition:**
Phonetik	*„Theorie der Produktion, der Rezeption und der akustischen Eigenschaften von menschlichen (Sprach-) Lauten“* [91].
Phonologie	*„Theorie des Lautsystems einer (oder der) Sprache(n)“* [91].
Morphologie	*„Theorie der internen Struktur der Wörter“* [91].
Syntax	*„Theorie der internen Struktur der Sätze; Satzlehre“* [91].
Semantik	*„Theorie der Bedeutung sprachlicher Ausdrücke“* [91].
Pragmatik	*„Theorie des Gebrauchs sprachlicher Ausdrücke; oder: Theorie der Handlungen, die Sprecher mit Sprachzeichen vollziehen“* [91].

Sprachliche Ebenen können aus computerlinguistischer Sicht als übergeordnete Schichten eines Textes aufgefasst werden. Je höher eine Schicht dabei liegt, desto größer ist ihr sprachlicher Umfang und damit wiederum komplexer bzgl. derer computerlinguistischer Realisierung. Die folgende Illustration zeigt den schemenhaften Aufbau dieser Schichten (bzw. sprachlichen Ebenen):

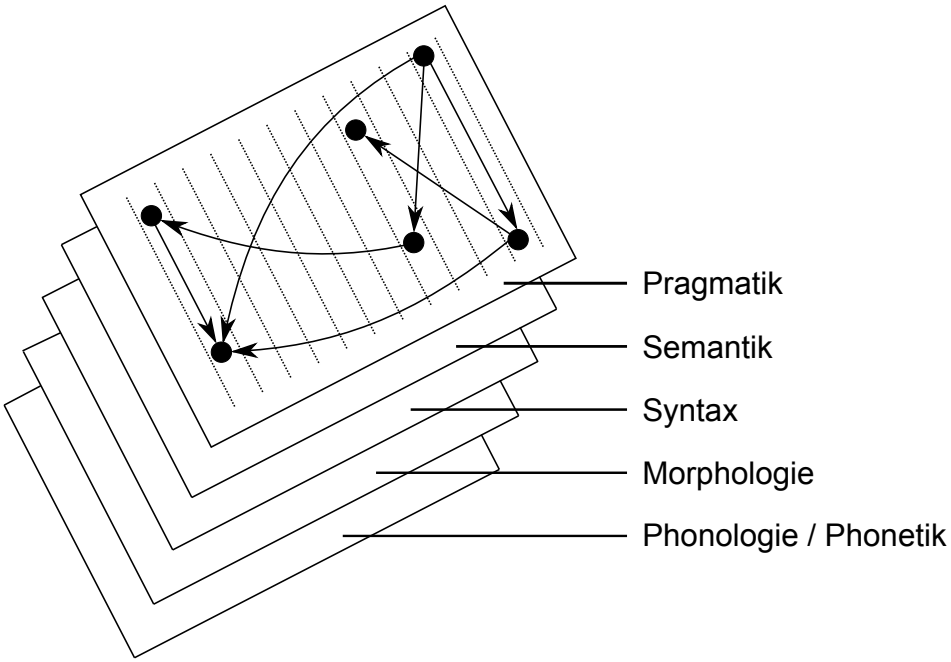


Abbildung 1: Sprachliche Ebenen eines Textes.

[1] In [43, Seite: 323] werden diese auch als „Linguistische Ebenen“ bezeichnet.

In der Autorschaftsanalyse spielen sprachliche Ebenen eine bedeutende Rolle. Jede einzelne Ebene beinhaltet individuelle Merkmale, die dazu beitragen können, einen Autorenstil annähernd bestimmen zu können. Bevor jedoch näher darauf eingegangen wird, wie solche Merkmale aussehen und wie dadurch Autorenstile beschrieben werden können, sollte zunächst diskutiert werden, was genau unter dem unpräzisen Begriff „Stil“ verstanden werden kann.

2.4 Stil

In der Literatur existieren für den Begriff „Stil“ viele unterschiedliche Auffassungen, sodass hier nicht wirklich von einer eindeutigen Definition gesprochen werden kann. Gansel fasst beispielsweise in [32] diesen Begriff wie folgt zusammen:

- *„Stil = Gesamtheit der in einem Text verwendeten Stilelemente in ihrem Zusammenwirken“.* [32]

Sowinski [95] wiederum dehnt den Begriff des Sprachstils in seinem Werk deutlich stärker aus. Seinen Beobachtungen zufolge kann Stil angesehen werden als:

- *„sprachlicher Schmuck“*, [95]
- *„individuelle Eigenart des Sprachausdrucks“*, [95]
- *„Spiegelung psychischen Erlebens“*, [95]
- *„Einheit der künstlerischen Gestaltung“*, [95]
- *„Abweichung von einer Norm“*, [95]
- *„zeit-und gruppengebundener Sprachausdruck“*, [95]
- *„gattungsgebundene Ausdrucksweise“*, [95]
- *„funktionale Redeweise“*, [95]
- *„Auswahl zwischen mehreren sprachlichen Möglichkeiten“*, [95]
- *„Gesamtheit quantitativer Merkmale“*, [95]
- *„Auswirkung besonderer grammatischer Regeln“*, [95]
- *„Teil der Textbedeutung“*, [95]
- *„besondere Form der Textrezeption“*, [95]

Was jedoch aus der Sicht von Experten nicht als Stil angesehen werden kann, versucht Golcher in [37] wie folgt auszugrenzen:

- *„In welcher Sprache/Dialekt jemand schreibt, ist keine Stilfrage.“* [37]
- *„Das Thema (topic), über das jemand schreibt, ist keine Stilfrage.“* [37]
- *„Das Genre eines Textes wird auch nicht als Stil bezeichnet.“* [37]

Daraus folgert Golcher wiederum:

- *„Alle irgendwie nachweisbaren Unterschiede, die noch übrig bleiben, gelten als Stil.“* [37].

Aus all diesen Aussagen wird deutlich, weshalb der Stilbegriff nur schwer zu definieren ist. Dies hat wiederum zur Folge, dass aus mathematischer Sicht ebenfalls keine eindeutige Definition des Silbegriffs möglich ist.

Dessen ungeachtet, kann Stil trotz der fehlenden Eindeutigkeit zumindest für einen Autoren individuell (und damit annähernd eindeutig) definiert werden. Unterstützt wird diese Idee durch mehrere Theorien, welche öfters in der Literatur[1] zu finden sind. Drommel[2] [23] beispielsweise hält an der folgenden Aussage fest:

- *„Es wird davon ausgegangen, dass jeder Mensch seinen individuellen sprachlichen Code und seine individuelle kommunikative Programmierung hat. Unser Individualprogramm bewirkt, dass wir uns in gleichen Sprachsituationen gleich verhalten, also die gleichen für uns typischen sprachlichen Merkmale produzieren“*, [23].

Das Finden dieser spezifischen Merkmale kann damit als Basis der Autorschaftsanalyse verstanden weden. Konkret geht es also darum Autorenstile (im Rahmen dieser Arbeit ausgedrückt in geschriebener Sprache) mit Hilfe von Stilmerkmalen voneinander zu unterscheiden. Doch wie genau lässt sich eine solche Unterscheidung bewerkstelligen? In der Forschung[3] hat sich dazu der Ansatz etabliert, Stil durch eine Menge von geeigneten Stilmerkmalen zu approximieren. Die Anspielung auf „approximieren“ ist hierbei hervorzuheben, da es extrem schwierig - wenn gar unmöglich - ist, Stil vollständig durch Stilmerkmale abzubilden, sodass hier tatsächlich nur eine Annäherung des individuellen Stils realisiert werden kann. Trotz dieser Problematik wird im Rahmen dieser Arbeit eine mathematische Definition angestrebt. Es wird daher versucht, Stil mit Hilfe der obengenannten Erkenntnisse und der eingeführten Notation wie folgt zu formalisieren:

Definition 2.16. $\mathcal{STIL} \approx \ \textit{Stilmerkmale} \ = \{ f_1, f_2, \dots \} = F_1 \cup F_2 \cup \dots = \mathbb{F}$

Informell bedeutet dies, dass der individuelle Stil eines Autors durch eine hinreichend große Menge von Stilmerkmalen annähernd repräsentiert werden kann. Analog zu dieser Definition kann Stil auch als ein abstraktes Modell illustriert werden. Das Resultat des Modells spiegelt dabei den gegenwärtigen Stil eines Autors $\mathcal{A}$ wieder:

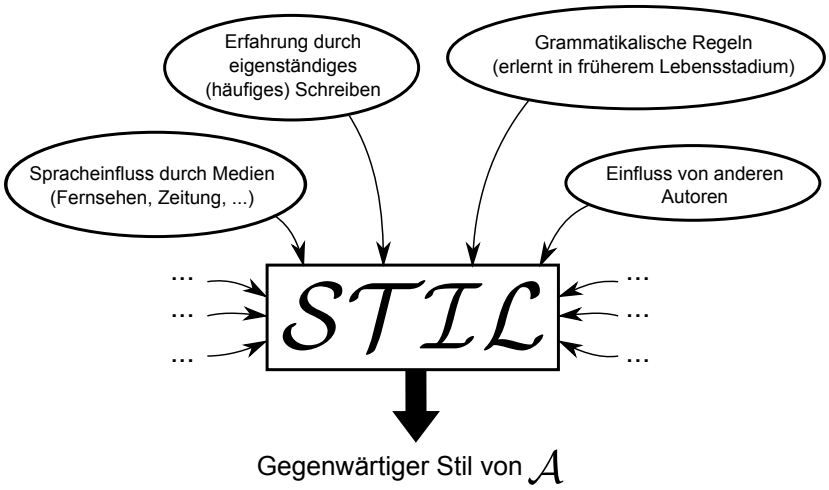


Abbildung 2: Abstraktes Stil-Modell.

Aus diesem Modell kann die Erkenntnis gewonnen werden, dass die Definition des Stilbegriffs nicht nur von der Annäherung durch sprachliche Merkmale, sondern auch vom Entwicklungsstadium von $\mathcal{A}$ abhängt. Es bleibt nun zu klären, was genau Stilmerkmale darstellen und wie diese aus einem Text entnommen werden können.

[1] Siehe dazu beispielsweise: [4, 74].

[2] Prof. Dr. Raimund H. Drommel zählt zu den besten Sprachprofilern Deutschlands. Er blickt auf eine 25 Jahre langen Berufserfahrung in dieser Branche. [22]

[3] In der Sprachwissenschaft existiert eine spezielle Disziplin mit der Bezeichnung „Stilistik“, welche sich ausschließlich mit dem Phänomen Stil und den damit verbundenen Stilmitteln auseinandersetzt (nähere Details auf Seite: 80).

2.5 Features (Einführung)

Aus dem letzten Abschnitt wurde ersichtlich, dass Stilmerkmale (ab hier auch „Features“ genannt) den essentiellen Kern der Autorschaftsanalyse ausmachen. Anders ausgedrückt: Ohne Features wäre eine Unterscheidung der Autorenstile praktisch undenkbar. Es soll daher in diesem Abschnitt kurz erläutert werden, was unter den Begriff der Features verstanden werden kann. In Kapitel 5 wird später ausführlicher auf diese Thematik eingegangen.

Um Features näher beschreiben zu können, wird nachfolgend eine (eigene) Definition eingeführt, die innerhalb dieser Arbeit gilt:

Definition 2.17. *Ein Feature beschreibt ein charakteristisches Merkmal eines Dokuments, welches beispielsweise durch ein Satzglied (Zeichen, Wort, Phrase/Konstituente, etc.) oder eine Kombination von diesen, repräsentiert werden kann. Dieser kann hierbei wahlweise qualitativ (kategorisch) oder quantitativ (numerisch) ausgedrückt werden.*

Das folgende Beispiel verdeutlicht diese Definition. Sei dazu zunächst ein Dokument $\mathcal{D}_\mathcal{A}$ wie folgt gegeben:

$\mathcal{D}_\mathcal{A} =$ „Endlos dehnen sich die Felder und Wälder in Ostpreußen und verlieren sich nach der Steppenewigkeit Rußlands. Wie eine blitzende Kette sind die vielen Seen in die bäuerliche Erde verstreut. Darüber wölbt sich der gewaltige Himmel. Im freien Raum zwischen den Wolken und den Straßen sausen im Winter die krachenden Stürme. Die Menschen, die hier wohnen, sind ernst und verschlossen." [7]

Seien weiterhin zwei Features $f_1 =$ `Adjektive` und $f_2 =$ `Einfache Nominalphrasen` (Artikel + Nomen) gegeben. Eine Anwendung[1] von f_1, f_2 auf $\mathcal{D}_\mathcal{A}$ würde dann die folgenden Satzglieder ergeben:

$\boxed{f_1} \rightsquigarrow \mathcal{D}_\mathcal{A}$ = (`Endlos, blitzende, bäuerliche, gewaltige, freien, krachenden, ernst, verschlossen`)

$\boxed{f_2} \rightsquigarrow \mathcal{D}_\mathcal{A}$ = (`die Felder, der Steppenewigkeit, den Wolken, den Straßen, Die Menschen`)

Bei genauerer Betrachtung fällt auf, dass die Anwendung beider Features fast die Hälfte des Textes wiedergibt. Dies ist dahingehend problematisch, weil dadurch eine sogenannte Überanpassung (engl. *Overfitting*) bezüglich des Autorschaftsanalyse-Szenarios entsteht. Eine Überanpassung hätte zur Folge, dass ein Autorschaftsanalyse-System nicht in der Lage wäre zu generalisieren. Mit anderen Worten: Falls die oberen Satzglieder aus $\mathcal{D}_\mathcal{A}$ in einem anderen Dokument $\mathcal{D}_\mathcal{A}{}'$ nicht (oder nur im kleineren Umfang) vorkommen, dann würde die korrekte Attribution der Autorschaft von $\mathcal{D}_\mathcal{A}{}'$ höchstwahrscheinlich scheitern. Aus diesem Grund sollten Features nicht ausschließlich eine kategorische Form aufweisen, sondern vielmehr durch entsprechende Häufigkeiten[2] der betroffenen Satzglieder numerisch ausgedrückt werden, z.B.:

- $f_1(\mathcal{D}_\mathcal{A}) =$ Anzahl aller Adjektive im Verhältnis zu allen Wörtern in $\mathcal{D}_\mathcal{A} = \frac{8}{59} \approx 0.14$.
- $f_2(\mathcal{D}_\mathcal{A}) =$ Anzahl aller Sätzen im Verhältnis zu allen Nominalphrasen in $\mathcal{D}_\mathcal{A} = \frac{5}{5} = 1$.

[1] Die genauere Bezeichnung dieses Vorgangs lautet „Feature Entnahme“ und stellt in einem Autorschaftsanalyse-Szenario einen der ersten und wichtigsten Schritte dar.

[2] Das Kapitel 5.7.1 erläutert diese Thematik im Detail.

2.6 Text Normalisierung

Bevor Features aus den unterschiedlichsten sprachlichen Ebenen eines Textes $\mathcal{D}$ entnommen werden können, müssen (bzw. sollten) davor eine Reihe von Aktionen auf $\mathcal{D}$ durchgeführt werden, um den Text und die darin befindlichen Satzglieder auf eine einheitliche Form zu bringen. Die durchzuführenden Aktionen werden in dieser Arbeit unter dem Begriff Text Normalisierung[1] zusammengefasst und beinhalten unter anderem die Behandlung folgender Probleme:

Problem	**Behandlung**
Akronyme, Abkürzungen	Ignorieren, entfernen oder mittels geeigneter Wortlisten ausschreiben.
Überflüssige Symbole	Entfernen mit Hilfe regulärer Ausdrücke.
URL's, Mailadressen	Entfernen mit Hilfe spezieller Werkzeuge (z.B. HTML-Parser).
Smilies	Entfernen oder mittels spezieller Wortlisten/Wörterbücher substituieren.
Fremdsprachige Abschnitte	Entfernen mithilfe spezieller Werkzeuge (z.B. Language Guesser).
Variable Leerräume, mehrfache Buchstaben	Ersetzung durch ein einzigen Leerzeichen, bzw. Buchstaben.

Beobachtung: Ein weiteres „Problem“ das in dieser Tabelle nicht vorkommt, stellt die vermischte Form von Groß und Kleinschreibung dar. Intuitiv würde sich als Behandlung anbieten, den Text durchgängig klein zu schreiben, jedoch könnten dadurch wertvolle Features[2] verloren gehen. Aus diesem Grund wird von einer solchen Behandlung in dieser Arbeit abgesehen.

Im Allgemeinem lässt sich der Prozeß der Text Normalisierung nicht zu 100% automatisieren. Dies hat viele unterschiedliche Gründe. So wird beispielsweise ein Language Guesser für den folgenden Satz nur schwer die Entscheidung fällen können, ob dieser in Deutsch oder Englisch vorliegt:

$$\sigma_1 = \texttt{Michael hat sich zwei coole Tools downgeloadet.}$$

In σ_1 sind neben einen bekannten amerikanischen Vornamen drei Anglizismen enthalten, die in ihrer Gesamtheit (vier englisch-klingende Wörter gegen drei deutsche Wörter) dazu führen können, einen Language Guesser bei seiner Entscheidungsfindung, zu welcher Sprache σ_1 angehört, zu irritieren.

Ein anderes Problem betrifft Akronyme und Abkürzungen. Diese sind nämlich in vielen Fällen auf bestimmte Domänen begrenzt (z.B. Internet-Jargon, Verwaltungssprache, Schülersprache, etc.). Wenn diese anhand von allgemeinsprachlichen Wortlisten ausgeschrieben werden, kann dies zu unvorhergesehenen Ergebnissen führen. Der folgende Satz verdeutlicht diese Problematik:

$$\sigma_2 = \texttt{Schau doch mal im MHB nach !}$$

Die Abkürzung `MHB` kann ohne den genaueren Kontext von σ_2 nur schwer erraten bzw. ausgeschrieben werden. Je nach Domäne kann daher diese Abkürzung die unterschiedlichsten Bedeutungen haben. Im Kontext der Studentensprache bedeutet `MHB` = `Modulhandbuch`. In BWL-Fachjargon bedeutet `MHB` dagegen `Management-Handbuch` und in der Fachsprache der Staatssicherheit bedeutet `MHB` wiederum `Menschenhändlerbande`. Damit also derartige Textentstellungen vermieden werden können, sollten Akronyme/Abkürzungen entweder ignoriert oder jedoch durch einen Menschen ausgeschrieben werden, der die Domäne des Dokuments kennt. Eine derartige manuelle Ausschreibung ist jedoch mit einem größeren Mehraufwand verbunden, die mit zunehmender Textlänge steigt und daher wohlüberlegt werden sollte.

[1] In der Literatur auch bekannt unter dem Begriff „Pre-Processing“.

[2] Details dazu finden sich im Kapitel 5.5.2.

3 Computerlinguistische Grundlagen

In diesem Kapitel werden aufbauend auf die eingeführten linguistischen Begriffe aus Kapitel 2.2 computerlinguistische Konzepte und Verfahren erläutert, auf die im späteren Verlauf dieser Arbeit (insbesondere in den Kapiteln 5, 6 und 7) Bezug genommen wird.

3.1 Natural Language Processing

Die Disziplin des Natural Language Processing (kurz NLP), welche als ein Teilbereich der künstlichen Intelligenz eingeordnet werden kann, beschäftigt sich mit der rechnergestützten Verarbeitung natürlicher Sprache. Ausgehend von einer freien Übersetzung nach Liddy, [65] kann NLP wie folgt definiert werden:

Definition 3.1. ***NLP:*** *NLP ist ein theoretisch motivierter Bereich von rechnergestützten Verfahren, um natürlichsprachige Texte auf einer oder mehreren Ebenen der linguistischen Analyse zu untersuchen. Das Ziel dabei ist es eine menschenähnliche Sprachverarbeitung für eine Reihe von Aufgaben oder Anwendungen zu realisieren, [65].*

Liddy beschreibt in [65], dass die Ursprünge von NLP in den späten vierziger Jahren vorzufinden sind, als das erste maschinelle Übersetzungssystem[1] entwickelt wurde. Dieses wurde während des zweiten Weltkriegs, unter Zuhilfenahme von kryptographischen und informationstheoretischen Verfahren benutzt, um feindliche Codes zu brechen. Nach dieser Zeit wurde NLP zunächst nur theoretisch weiter erforscht, bis sich der wahre Durchbruch einige Jahrzehnte später zeigte, nachdem Computersysteme immer größere Kapazitäten und mehr Rechenleistung vorweisen konnten.

Im Kontext der Autorschaftsanalyse spielt NLP eine tragende Rolle, da diese Werkzeuge bereitstellt, mit deren Hilfe Texte in sprachliche Ebenen abgebildet werden können. Aus den sprachlichen Ebenen können wiederum Stilmerkmale gewonnen werden, die die Basis einer Autorschaftsanalyse darstellen. Im nächsten Abschnitt werden die wichtigsten NLP-Werkzeuge vorgestellt, die für die Realisierung des Frameworks in Kapitel 8 benötigt wurden.

3.2 NLP-Werkzeuge für die Autorschaftsanalyse

Eine automatische Analyse, Verarbeitung und Informationsgewinnung bzgl. eines Textes erfordert typischerweise den kollaborativen Einsatz zahlreicher Werkzeuge, vor allem solcher aus dem Gebiet des NLP's. In diesem Abschnitt werden daher die wichtigsten Werkzeuge aus dem NLP-Umfeld erläutert, die gebraucht werden, um eine Autorschaftsanalyse durchzuführen. Der nachfolgende Beispielsatz σ dient dabei als Referenzbeispiel für einige der genannten Werkzeuge:

$$\sigma = \texttt{Der kleine David spielte mit der Katze im Garten.}$$

3.2.1 Reguläre Ausdrücke

Reguläre Ausdrücke[2] (engl. *Regular Expression*, kurz: RegEx) beschreiben abstrakte Suchmuster die auf Texte bzw. Zeichenketten angewendet werden. Sie werden häufig eingesetzt um Zeichenketten auf formale Kriterien oder Zusammensetzungen zu übergeprüfen. So lassen sich mit regulären Ausdrücken beispielsweise Postleitzahlen, E-Mail-Adressen, URL's oder auch Bestellnummern validieren.

Des Weiteren kommen reguläre Ausdrücke zur Anwendung, wenn Zeichenketten (oder Teile davon) gesucht und durch andere Zeichenketten ersetzt[3] werden sollen.

---

[1] Maschinelle Übersetzung zählt heutzutage als eine eigenständige Disziplin, die ein Teilbereich von NLP darstellt.

[2] Da reguläre Ausdrücke aufgrund ihrer Komplexität ganze Bücher füllen können, wird hier für weiterreichende Informationen auf eine entsprechende Literatur (z.B. [69]) verwiesen.

[3] Populäre Beispiele sind hier z.B. die Suchen/Ersetzen-Dialoge in Office-Anwendungen.

Das folgende Beispiel illustiert die Anwendung eines regulären Ausdrucks RegEx auf σ um alle Tokens zu finden, die mit einem großen oder kleinen „`k`“ anfangen, mindestens einen Kleinbuchstaben von „`a`“ bis „`z`“ enthalten und schließlich mit einem „`e`“ enden. Sei dazu der reguläre Ausdruck gegeben durch: RegEx = `[Kk][a-z]+[e]`, dann gilt:

$$\boxed{\text{RegEx}} \rightsquigarrow \sigma = (\ \texttt{kleine, Katze}\ )$$

Reguläre Ausdrücke sind ein essentieller Bestandteil in der Autorschaftsanalyse. Sie werden unter anderem im Rahmen der Text Normalisierung benötigt, um Rauschen (z.B. URL's, HTML/XML-Tags oder überflüssige Symbole) zu entfernen. Eine andere deutlich wichtigere Anwendung regulärer Ausdrücke betrifft die Entnahme von Features aus den unterschiedlichsten sprachlichen Ebenen eines Textes.

3.2.2 Stemmer

Unter dem Begriff „Stemming“ wird der Prozess der Zurückführung eines Wortes auf dessen Stamm-Morphem verstanden. Als Stemmer werden dabei die jenigen Algorithmen bezeichnet, die diesen Prozess ausführen. Das folgende Beispiel verdeutlicht das Resultat eines Stemmers:

$$\boxed{\text{Stemmer}} \rightsquigarrow \sigma = (\ \texttt{Der, klei, David, spiel, mit, der, Kat, im, Gar}\ )$$

Hieraus lässt sich leider die Schwäche eines Stemmers erkennen. Suffixe von Wörtern im Deutschen (bzw. auch in anderen Sprachen) lassen sich algorithmisch nicht ohne Weiteres korrekt entfernen, um daraus den Stamm-Morphem zu erhalten. Der Grund dafür ist, dass viele Wörter nur zum Schein aussehen als wären sie flektiert, sodass Stemmer, bedingt durch ihre einfach gehaltenen Regeln, Wortendungen irrtümlich entfernen die eigentlich zu den Stamm-Morphemen gehören und nicht entfernt werden dürfen. Ein anderes Problem, welches aus dem Beispiel nicht ersichtlich wird, betrifft die einseitige Zurückführung der Wörter. Die meisten Stemmer sind nur in der Lage Suffixe durch entsprechende Regeln zu entfernen, Präfixe hingegen bleiben unberührt:

$$\boxed{\text{Stemmer}} \rightsquigarrow \texttt{verspielt} = \texttt{verspiel}$$

Hier hat der Stemmer das Suffix „`t`“ korrekt entfernt, der Präfix „`ver`“ wurde jedoch ignoriert. Das korrekte Ergebnis wäre hier also der Stamm-Morphem „`spiel`“ gewesen. Ein Stemmer wird in dieser Arbeit (in Kombination mit regulären Ausdrücken) benötigt, um die sprachliche Ebene „Morphologie“ abzubilden, sodass dadurch Features aus dem Text gewonnen werden können.

3.2.3 $n-$Gramm Konstruktion

Unter diesem Begriff wird ein einfaches Verfahren verstanden, mit dessen Hilfe eine Folge von überlappenden Textfragmenten der Größe n erzeugt werden. Die Textfragmente werden hierbei als $n-$Gramme bezeichnet und können entweder aus Buchstaben, Tokens oder Ähnlichem bestehen. Werden Buchstaben als Textfragmente verwendet, so heißen diese Buchstaben $n-$Gramme, werden dagegen Tokens verwendet, so werden diese Token $n-$Gramme genannt. Je nach Größe von n werden auch stellvertretend die folgenden Bezeichnungen für $n-$Gramme verwendet:

Bezeichnung:	n
`Unigramme`	`1`
`Bigramme`	`2`
`Trigramme`	`3`
`Tetragramme`	`4`
`Pentagramme`	`5`
⋮	⋮

Tabelle 2: $n-$Gramm Bezeichnungen.

Das folgende Beispiel verdeutlicht die Idee der $n-$Gramm Konstruktion:

$$\boxed{\text{Buchstaben Bigramme}} \rightsquigarrow \sigma = (\texttt{ De, er, kl, le, ei, in, ne, Da, av, vi, id, ..., im, Ga, ar, rt, te, en, n. })$$

$$\boxed{\text{Token Bigramme}} \rightsquigarrow \sigma = (\texttt{ Der kleine, kleine David, David spielte, ..., Katze im, im Garten. })$$

Die Anzahl der erzeugten $n-$Gramme hängt hierbei von n sowie der Anzahl der Elemente $\{$ `Buchstaben`, `Wörter`, `Phrasen`, $\ldots\}$ in der Grundgesamtheit (in diesem Fall σ) ab. Für die Anzahl der Buchstaben Bigramme in σ gilt so z.B. $\#(\texttt{Buchstaben}, \sigma) - n + 1$.

Die $n-$Gramm Konstruktion wird innerhalb dieser Arbeit für zahlreiche Zwecke verwendet. Neben der Generierung von Features stellt diese einen wichtigen Bestandteil einiger Verfahren dar, bei denen z.B. überlappende Satzsegmente konstruiert werden müssen, um dadurch Regionen im Text zu lokalisieren, die Stil-Inkonsistenzen enthalten.

3.2.4 Tokenizer

Unter dem Begriff „Tokenizing“ wird die Segmentierung einer Zeichenkette bzw. eines Textes anhand eines vorgegebenen Separators (Trennzeichen) verstanden. Als Tokenizer wird ein Algorithmus bezeichnet, der diese Segmentierung ausführt. Das Resultat eines Tokenizers ist eine Folge von Tokens, die im folgendem Beispiel anhand eines Leerzeichen-Separators segmentiert wurden:

$$\boxed{\text{Tokenizer}} \rightsquigarrow \sigma = (\texttt{ Der, kleine, David, spielte, mit, der, Katze, im, Garten. })$$

Ein Tokenizer wird innerhalb dieser Arbeit unter anderem dafür benötigt, wortbasierte[1] Features aus Texten zu gewinnen. Des Weiteren dienen die resultierenden Tokens, bzw. deren absoluten Häufigkeiten, als ein Normalisierungsmaß für viele (nicht zwingenderweise wortbasierte) Features.

3.2.5 Sentence Tokenizer

Ein Sentence Tokenizer verhält sich analog zu einem Tokenizer. Das Resultat ist hier jedoch eine Folge von Sätzen anstelle von Tokens. Als Beispiel soll hier wieder das Textfragment $\mathcal{D}$ von Seite: 25 dienen:

$$\boxed{\text{Sentence Tokenizer}} \rightsquigarrow \mathcal{D} = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$$

Die resultierenden Sätze lauten dabei:

$\sigma_1 =$	„Endlos dehnen sich die Felder und Wälder in Ostpreußen und verlieren sich nach der Steppenewigkeit Rußlands." [7]
$\sigma_2 =$	„Wie eine blitzende Kette sind die vielen Seen in die bäuerliche Erde verstreut." [7]
$\sigma_3 =$	„Darüber wölbt sich der gewaltige Himmel." [7]
$\sigma_4 =$	„Im freien Raum zwischen den Wolken und den Straßen sausen im Winter die krachenden Stürme." [7]
$\sigma_5 =$	„Die Menschen, die hier wohnen, sind ernst und verschlossen." [7]

Ein Sentence Tokenizer hat in dieser Arbeit vielfältige Aufgaben. Neben der Gewinnung von satzbasierten Features und deren Normalisierung, wird dieser vor allem für die intrinsische Exploration benötigt. Dort dient er als ein unverzichtbares Werkzeug für die Zerlegungsstrategien bzgl. der Dokumente.

[1] Siehe dazu Kapitel 5.5.4 sowie Kapitel 5.5.5

3.2.6 Part-Of-Speech Tagger

Unter dem Begriff Part-Of-Speech Tagger (kurz: POS-Tagger) wird ein Programm verstanden, welches für jeden Token innerhalb eines Textes diejenige Wortart zuordnet, die am wahrscheinlichsten dafür in Frage kommt, [15]. Dafür benötigt der POS-Tagger einen sogenannten Trainingskorpus (auch Sprachmodell genannt), welcher Regeln oder Wahrscheinlichkeiten enthält, mit deren Hilfe die Zuweisung der Wortarten erfolgt. Der Trainingskorpus benötigt wiederum ein zuvor festgelegtes Tagset[1], das die Menge aller zuweisbaren Wortarten darstellt. Das folgende Beispiel verdeutlicht das Resultat eines POS-Taggers:

$$\boxed{\text{POS-Tagger}} \rightsquigarrow \sigma = \Big((\tau_1, \omega_1),\, (\tau_2, \omega_2),\, \ldots,\, (\tau_{10}, \omega_{10}) \Big)$$

Die jeweiligen Tupeln (Token, Wortart) = (τ_i, ω_i) lauten dabei:

Token τ_i	**Wortart** ω_i	**Bedeutung**
τ_1 = `Der`	ω_1 = `ART`	Bestimmter Artikel
τ_2 = `kleine`	ω_2 = `ADJA`	Attributives Adjektiv
τ_3 = `David`	ω_3 = `NE`	Eigennamen
τ_4 = `spielte`	ω_4 = `VVFIN`	Finites Verb
τ_5 = `mit`	ω_5 = `APPR`	Präposition (linke Zirkumposition)
τ_6 = `der`	ω_6 = `ART`	Bestimmter Artikel
τ_7 = `Katze`	ω_7 = `NN`	Nomen
τ_8 = `im`	ω_8 = `APPRART`	Präposition
τ_9 = `Garten`	ω_9 = `NN`	Nomen
τ_{10} = `.`	ω_{10} = `$.`	Satzbeendende Interpunktion

In der Praxis werden hauptsächlich drei Arten von POS-Taggern unterschieden:

1. **Regelbasierte POS-Tagger:** Diese Tagger basieren auf festgelegten Regeln die manuell erstellt worden sind und liefern in der Regel qualitativ hochwertige Ergebnisse. Gleichzeitig erfordern sie jedoch einen enormen Aufwand für die Erstellung der Regeln, sodass diese für reale Anwendungsszenarien kaum noch zum Einsatz kommen.

2. **Stochastische POS-Tagger:** Diese Tagger basieren auf statistische Modelle (in der Regel Hidden Markov Modelle) und haben gegenüber regelbasierter Tagger den Vorteil, dass sie automatisch trainiert und auch auf ähnliche[2] Sprachen adaptiert werden können. Dafür greifen sie auf große Wahrscheinlichkeitstabellen anstelle von Regeln zurück, [40]. Stochastische POS-Tagger sind heutzutage weit verbreitet und liefern sehr hohe Erkennungsraten hinsichtlich der vorhergesagten Wortarten. In [35] berichten Giesbrecht et al. über den *Bidirectional MaxEnt Stanford Tagger*[3], der mit einer Erkennungsrate von 97.63 % für die Anwendung auf bestimmte Domänen (z.B. Nachrichtentexte oder moderner Literatur) als praxistauglich eingestuft werden kann.

3. **Hybride POS-Tagger:** Hierbei handelt es sich um eine Kombination beider Ansätze.

POS-Tagger stellen sehr wichtige NLP-Werkzeuge dar, da sie als Basis für andere (komplexere) NLP-Werkzeuge wie etwa Chunker oder Parser dienen. Innerhalb der Autorschaftsanalyse wird ein POS-Tagger unter anderem dazu benötigt, um verschiedene Features zu generieren.

---

[1] Im Anhang (Seite: 166) befindet sich eine vollständige Liste aller POS-Tags (Wortarten) die sowohl innerhalb dieser Arbeit als für das Framework verwendet wurden.

[2] Ähnlich bedeutet dabei z.B. Deutsch und Englisch, nicht jedoch Hebräisch und Japanisch.

[3] Dieser POS-Tagger wurde innerhalb des entwickelten Frameworks ebenfalls verwendet.

3.2.7 Chunker

Unter dem Begriff Chunker wird ein Programm verstanden, welches eine „flache“ Analyse von Sätzen durchführt. Eine flache Analyse bedeutet dabei, dass die Sätze nicht vollständig bis zur POS-Tag Ebene analysiert, sondern stattdessen nur die darin befindlichen Satzgliedern (z.B. Nominalphrasen) als zusammengehörende Einheiten bestimmt werden [14]. Diese Einheiten werden Chunks (bzw. genauer „IOB-Chunks“) genannt und sind wie folgt definiert:

Definition 3.2. ***IOB-Chunks:*** *„beschreiben eine flache syntaktische Schicht über die getaggten Wortformen, welche in Form von Chunk-Tags auf die Token abbildbar ist.“, [15, Seite: 53].*

- *B-K: Anfang einer Chunk-Konstituente K*
- *I-K: Fortsetzung der Chunk-Konstituente K*
- *0: Nicht zugeordnet (wird auch chink genannt)*

Das folgende Beispiel zeigt das Resultat eines Chunkers nach dessen Anwendung auf σ, wobei hier nur Nominalphrasen `NP` als Konstituenten betrachtet werden:

$$\boxed{\text{Chunker}} \rightsquigarrow \sigma = \begin{array}{l} \texttt{[Der]\_B-NP [kleine]\_I-NP [David]\_I-NP} \\ \texttt{[spielte]\_O} \\ \texttt{[mit]\_O} \\ \texttt{[der]\_B-NP [Katze]\_I-NP} \\ \texttt{[im]\_O} \\ \texttt{[Garten]\_B-NP} \end{array}$$

Wie man aus dieser Ausgabe entnehmen kann, ist ein Chunker nicht immer unfehlbar bei seiner Entscheidung, welche Satzglieder tatsächlich zusammengehören. So müsste hier die Konstituente „`im Garten`“ nämlich ebenfalls zu einem `B-NP`-Chunk zusammengefasst werden, was jedoch vom Chunker übersehen wurde. Die Qualität des Chunking-Ergebnisses basiert vor allem darauf, auf welches Sprachmodell es trainiert wurde. Die obige Ausgabe wurde so beispielsweise mit dem „OpenNLP Chunker“ erzeugt[1].

Chunker dienen in der Autorschaftsanalyse der Generierung sowie die Normalisierung einiger Features (z.B. durchschnittliche Länge von Nominalphrasen).

3.2.8 Parser

Ein Parser ist ein komplexes NLP-Werkzeug, dass das Ziel hat zu einem gegebenen Text (welcher in einer Form von tokenisierten Sätzen $\sigma_1, \sigma_2, \ldots, \sigma_n$ vorliegt) die zugrundeliegende syntaktische Struktur anhand einer festgelegten Grammatik bestimmen zu können. Hierfür benötigt dieser, analog zu den bereits genannten NLP-Werkzeugen, ein (statistisches) Sprachmodell, welches unter anderem Syntaxregeln enthält.

Parser lassen sich hinsichtlich der Konstruktion des Ableitungsbaums in sieben Parsingstrategien unterteilen, [87]. Im Folgenden werden die zwei bekanntesten Strategien kurz erläutert:

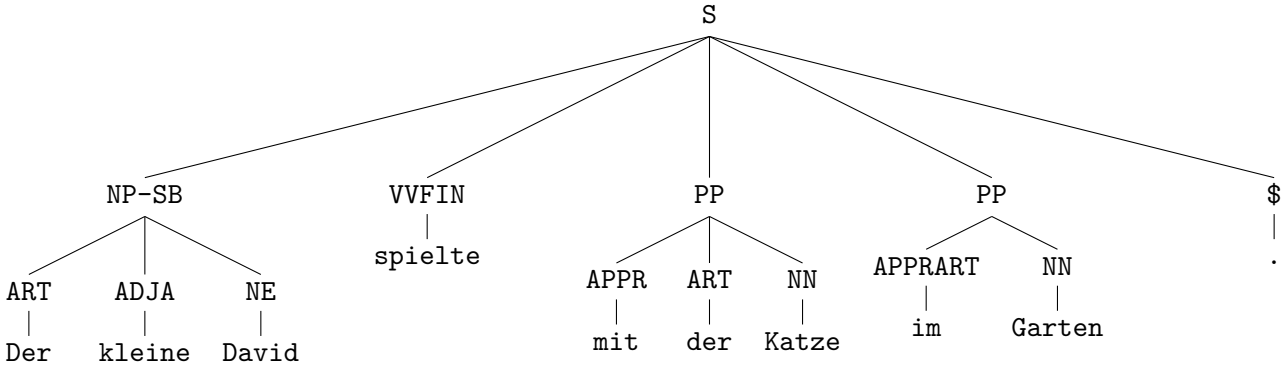
- **Top-Down:** Bei dieser Strategie werden ausgehend von einem Wurzelknoten `S` solange mögliche Zerlegungen von Konstituenten getestet, bis eine Zerlegung gefunden wurde, die dem Eingabesatz σ_i entspricht, [89].
- **Bottom-Up:** Bei dieser Strategie werden zum Eingabesatz σ_i solange mögliche lexikalische Kategorienzuordnungen sowie Zusammenfassungen von Konstituentenfolgen zu höheren Konstituenten getestet, bis der Wurzelknoten `S` erreicht ist, [89].

[1] Siehe in diesem Zusammenhang Kapitel 8.2.2.

Das folgende Beispiel verdeutlicht das Resultat eines Parsers:

```
Parser ⇝ σ =  (S
                (NP-SB (ART Der) (ADJA kleine) (NE David))
                (VVFIN spielte) (PP (APPR mit) (ART der) (NN Katze))
                (PP (APPRART im) (NN Garten))
                (\$.)
              )
```

Hieraus lässt sich grob die syntaktische Struktur von σ erkennen, welcher mit Hilfe der Tags aus dem sogenannten „Penn-Treebank Tagset[1]“ annotiert wurde. Da dieses Format in erster Linie für Maschinen anstatt für Menschen konzipiert wurde, kann stattdessen auch die folgende Ableitungsbaum-Darstellung bzgl. σ betrachtet werden:



Innerhalb dieser Arbeit wurde ein Parser in erster Linie dafür gebraucht, um die sprachliche Ebene „Syntax“ abzubilden. Anhand der erzeugten syntaktischen Struktur der Sätze, konnten wiederum unterschiedliche Features aus den Texten gewonnen werden.

Beobachtung: So praktisch Parser auch erscheinen mögen, so haben diese leider auch einige nennenswerte Nachteile. Der wohl wichtigste Nachteil betrifft dabei das Laufzeitverhalten. Tests haben ergeben, dass für das Parsing eines Dokuments mit der Größenordnung einer knappen DIN-A4 Seite ca. zwei bis drei Minuten Verarbeitungszeit einkalkuliert werden muss.

Da innerhalb dieser Arbeit mehrere Tausend Dokumente untersucht worden sind, musste eine Strategie gefunden werden die Anwendung des Parsers auf ein Minimum zu begrenzen. Wie diese Strategie umgesetzt wurde, wird in Kapitel 8 näher erläutert.

[1] Im Anhang (Seite: 167) befindet sich die Auflistung aller Elemente des Tagsets.

4 Grundlagen des Maschinellen Lernens

In diesem Kapitel werden einige Grundlagen des Maschinellen Lernens (kurz ML) beschrieben, die innerhalb dieser Arbeit die Basis mehrerer Autorschaftsanalyse-Verfahren bilden. Zunächst wird eine kurze Einführung in die Disziplin des ML's gegeben, wobei auch andere Disziplinen eingekreist werden, von denen viele ML-Verfahren inspiriert wurden. Danach werden einige Begriffe und zwei der wichtigsten Lernmethoden erläutert, mit denen sich ML-Verfahren einordnen lassen. Anschließend werden Metriken vorgestellt, die sowohl für ML- als auch Autorschaftsanalyse-Verfahren einen wichtigen Bestandteil darstellen.

Im weiteren Verlauf des Kapitels wird auf das zentrale Thema der Klassifikation eingegangen. Hierbei werden drei Klassifikationsverfahren vorgestellt, die zum einen für die Realisierung einiger Verfahren innerhalb dieser Arbeit verwendet wurden und zum anderen als Baselines für diese in den Experimenten (Kapitel 10) gedient haben. Danach wird sich ein Unterkapitel der Thematik des Clusterings annehmen, welche in ML neben der Klassifikation ebenfalls eine wichtige Rolle einnimmt. Hierbei wird ein bekanntes Clustering-Verfahren beschrieben, dass für die Realisierung einiger Autorschaftsanalyse-Verfahren in dieser Arbeit verwendet wurde. Zum Ende des Kapitels werden schließlich einige Evaluierungsmöglichkeiten genannt, mit denen die in dieser Arbeit präsentierten Verfahren evaluiert werden.

4.1 Einführung in Maschinelles Lernen

ML stellt, genauso wie die Disziplin des Natural Language Processing, einen Teilbereich der künstlichen Intelligenz dar. Anders als in NLP, sind hier jedoch künstliche Systeme der zentrale Gegenstand die in der Lage sind, aus gegebenen Informationen zu „lernen“, [70]. Dabei ist diese Art des Lernens nicht mit der eines Menschen gleichzusetzen, da der Lernprozess hier nur auf eine Zielsetzung beschränkt ist und - zumindest bis heute noch - den menschlichen Lernfähigkeiten nicht ebenbürtig ist.

Im Kontext des ML's wird unter dem Begriff des Lernens die selbständige Erkennung von Gesetzmäßigkeiten oder Mustern verstanden mit denen Daten verallgemeinert werden können. Hierbei werden von den künstlichen Systemen Informationen generiert, die das erlernte Wissen repräsentieren. Die künstlichen Systeme, die sich algorithmisch beschreiben lassen, werden in ML-Jargon als Lerner- bzw. Lernalgorithmen bezeichnet, während die erzeugten Informationen Modelle[1] genannt werden, [70].

ML ist ein stetig wachsendes interdisziplinäres Wissenschaftsfeld, welches unter anderem Querbezüge aus Statistik, Robotik, Philosophie, Psychologie oder Biologie aufweist. Dementsprechend finden sich in ML zahlreiche Verfahren aus diesen Gebieten wieder, wie etwa Konzeptlernen (Psychologie), Bayessches Lernen (Statistik) oder genetische bzw. evolutionäre Algorithmen (Biologie). Aufgrund ihrer guten Generalisierungsfähigkeit kommen ML-Verfahren in den unterschiedlichsten Anwendungsszenarien zum Einsatz. Neben Spracherkennung, (Hand-)Schrifterkennung, Kundensegmentierung oder Prozessoptimierung werden ML-Verfahren auch für die Aufdeckung von Kreditkartenmissbrauch verwendet.

[1] Andere gängige Bezeichnungen sind auch „Konzepte“ oder „Hypothesen“.

4.1.1 Begriffe in ML

Im folgenden werden die wichtigsten Begriffe des ML's zusammengefasst, die innerhalb dieses Kapitels verwendet werden.

- **Objekt:** Ein Objekt o beschreibt einen Gegenstand aus der realen Welt. Eine Menge von r Objekten wird durch $\mathcal{O} = \{ o_1, o_2, \ldots, o_r \}$ symbolisiert.
- **Klasse:** Eine Klasse c beschreibt eine semantische Einteilung bzgl. o. Eine Menge von m Klassen wird durch $\mathcal{C} = \{ \texttt{Klasse}_1, \texttt{Klasse}_2, \ldots, \texttt{Klasse}_m \}$ symbolisiert, dabei können mehrere Objekte $o_1, o_2, \ldots, o_k$ ein und derselben Klasse $c_i \in \mathcal{C}$ zugeordnet sein.
- **Feature:** Ein Feature[1] f beschreibt ein Merkmal, welches charakteristisch für die Klassenzugehörigkeit des zu klassifizierenden Objekts o ist.
- **Instanz:** Eine Instanz $\mathcal{F}$ beschreibt einen $n-$dimensionalen Feature-Vektor, welcher ein Objekt o anhand von n Features repräsentiert. Eine Menge von r Instanzen wird durch $I = \{ \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r \}$ symbolisiert.
- **Trainingsmenge:** Eine Trainingsmenge $\mathcal{T}_{train} = I \times \mathcal{C} = \{ (\mathcal{F}_1, c_1), (\mathcal{F}_2, c_2), \ldots, (\mathcal{F}_r, c_r) \}$ ist eine Menge von r Instanzen, wobei jede Instanz mit einer zuvor festgelegten Klasse $c_i \in \mathcal{C}$ assoziiert ist. Ein Tupel $(\mathcal{F}_i, c_j) \in \mathcal{T}_{train}$ wird als eine Trainings-Instanz bezeichnet und vereinfacht durch $\mathcal{F}_{ic_j}$ symbolisiert.
- **Testmenge:** Eine Testmenge $\mathcal{T}_{test} = \{ \mathcal{F}_{\varepsilon_1}, \mathcal{F}_{\varepsilon_2}, \ldots \}$ ist eine Menge von Instanzen, die keiner Klasse zugeordnet sind. Ein einzelnes $\mathcal{F}_{\varepsilon_i}$ wird dabei als Test-Instanz bezeichnet.
- **Klassifikator:** Ein Klassifikator kann als eine abstrakte Funktion $\text{classify} : I \longrightarrow \mathcal{C}$ verstanden werden, die ein Objekt o (welches durch $\mathcal{F}$ repräsentiert wird) einer vordefinierten Klasse c zuordnet. Der Vorgang dieser Zuordnung wird dabei als Klassifikation bezeichnet. Anhand von $\mathcal{T}_{train}$ kann ein Klassifikator trainiert werden um dadurch ein Zuordnungsmodell (kurz Modell genannt) zu generieren. Mittels eines solches Modells können anschließend unbekannte Test-Instanzen klassifiziert werden.

In der Terminologie der Autorschaftsanalyse entspricht ein Objekt o_i, ein Dokument $\mathcal{D}_i$ und eine Klasse c_j dagegen einem Autor $\mathcal{A}_j$. Instanzen in ML sind identisch mit den Feature-Vektoren im Kontext der Autorschaftsanalyse (beide werden hierbei durch $\mathcal{F}$ symbolisiert). Die Mengen $\mathcal{T}_{train}$ und $\mathcal{T}_{test}$ entsprechen wiederum den Mengen $\mathbb{D}_{train}$ bzw. $\mathbb{D}_{test}$ in der Autorschaftsanalyse.

4.1.2 Lernmethoden in ML

ML-Verfahren lassen sich in unterschiedliche[2] Lernmethoden einordnen. Bekannt sind insbesondere die folgenden beiden:

- **Überwachtes Lernen:** Bei dieser Lernmethode ist die Klassenzugehörigkeit der Instanzen im Voraus bekannt. Das Ziel hierbei ist die Klassifikation von neuen (und damit unbekannten) Instanzen mit Hilfe eines trainierten Klassifikators.
- **Unüberwachtes Lernen:** Bei dieser Lernmethode ist die Klasseneinteilung der Instanzen von vornherein unbekannt. Das Ziel dieser Lernmethode ist die automatische Strukturierung von Instanzen. Den resultierenden Strukturen[3] können wiederum Klassenbezeichnungen zugeordnet werden, wobei dieser „semantische" Vorgang zumeist manuell durch den Menschen erfolgt.

[1] Im ML-Kontext öfters auch als *Attribut* bezeichnet.

[2] Heutzutage existieren viele weitere Unterscheidungen, die hier der Einfachheit halber nicht erwähnt werden.

[3] In der Literatur wird hierfür öfters der Begriff „Cluster" verwendet.

4.2 Metriken

Der Begriff Metrik kann im Kontext der Mathematik als eine axiomatisch definierte Größe von Abständen aufgefasst werden. Metriken stellen in ML einen essentiellen Bestandteil vieler Verfahren dar und sind dabei wie folgt definiert:

Definition 4.1. *Sei M eine nichtleere Menge. Eine Funktion $m\colon M \times M \to \mathbb{R}$ heißt Metrik auf M, falls sie die folgenden Eigenschaften nach [11] erfüllt:*

1.) **Selbstidentität:** $\forall x \in M : \Big(m(x,x) = 0 \Big)$

2.) **Positivität:** $\forall x, y \in M : \Big((x \neq y) \wedge m(x,y) > 0 \Big)$

3.) **Symmetrie:** $\forall x, y \in M : \Big(m(x,y) = m(y,x) \Big)$

4.) **Dreiecksungleichung:** $\forall x, y, z \in M : \Big(m(x,y) \leq m(x,z) + m(z,y) \Big)$

4.2.1 Distanzfunktionen

Nach Breiteneder, [11] stellt eine Distanzfunktion $dist : M \times M \to (\mathbb{R}^+ \cup \{0\})$ eine Metrik dar (sofern die vier zuvor genannten Eigenschaften erfüllt sind). Mit Hilfe einer Distanzfunktion lassen sich je zwei Instanzen bzw. Feature-Vektoren elementweise vergleichen, um so beurteilen zu können, ob diese zueinander ähnlich bzw. unähnlich sind. Sie stellen damit (insbesondere für die Lernmethode „Unüberwachtes Lernen“) ein unentbehrliches Werkzeug dar, da sich mit ihrer Hilfe ähnliche Instanzen gruppieren lassen und so bestimmte Strukturen innerhalb einer Trainingsmenge erkennbar werden (dazu später mehr).

4.2.2 Ähnlichkeitsfunktionen

Im Rahmen dieser Arbeit wird eine Ähnlichkeitsfunktion analog zu einer Distanzfunktion als eine Metrik[1] betrachtet. Der Begriff Ähnlichkeitsfunktion ist dabei jedoch in der Literatur nicht immer einheitlich definiert. Der Grund dafür liegt darin, dass die Semantik der „Ähnlichkeit“ je nach Kontext unterschiedlich aufgefasst und daher schwer vereinheitlicht werden kann. Die gängisten Bezeichnungen in der Literatur lauten unter anderem *Similarity Metric*, *Similarity Function* oder auch *Similarity Measure*. Fälschlicherweise wird auch manchmal der Begriff *Distanzfunktion* synonym verwendet, was semantisch gesehen nicht richtig ist, da hier eine Ähnlichkeit und keine Distanz gemessen wird. Im Rahmen dieser Arbeit wurde entschieden den Begriff „Ähnlichkeitsfunktion“ zu verwenden und hierfür eine eigene Definition[2] einzuführen.

Eine Ähnlichkeitsfunktion bildet zwei Vektoren auf einen Wert ab, der ihre Ähnlichkeit beschreibt. Als Definitionsmenge sind reele und damit auch binäre Zahlen zulässig. Da innerhalb dieser Arbeit die Form der Vektoren keine größere Rolle spielt, wird stellvertretend für binäre und reele Vektoren die Notation $X = (x_1, x_2, \ldots, x_n)$ und $Y = (y_1, y_2, \ldots, y_n)$ verwendet. Damit lässt sich eine Ähnlichkeitsfunktion wie folgt formalisieren:

Definition 4.2. *Eine Ähnlichkeitsfunktion ist definiert als:*

$$sim : \mathbb{R}^n \times \mathbb{R}^n \longrightarrow \{\, q \mid (q \in \mathbb{R}) \wedge (0 \leq q \leq 1) \,\}$$

Die konkrete Abbildungsvorschrift lautet damit $sim(X,Y) \mapsto [0;1]$, wobei 0 (Minimum) die größtmögliche Unähnlichkeit und 1 (Maximum) die maximale Ähnlichkeit zwischen X und Y beschreibt.

[1] Der Grund dafür kann aus der Tabelle 3 (Seite 3) entnommen werden.

[2] Es sei hier ausdrüklich erwähnt, dass diese von anderen Definitionen in der Literatur eventuell abweicht.

Beobachtung: Ein Wert α der sich zwischen den Grenzen 0 und 1 befindet, lässt sich im Allgemeinen nicht eindeutig interpretieren, da die Semantik hierbei kontextabhängig ist. Es kann allenfalls festgehalten werden, dass mit zunehmender Größe von α, die zwei Vektoren X und Y „ähnlicher“ werden.

4.2.3 Beziehung zwischen Distanz- & Ähnlichkeitsfunktionen

Es existiert ein Zusammenhang zwischen Distanz- und Ähnlichkeitsfunktionen. Beide können ineinander (in der Regel) problemlos überführt werden. Um beispielsweise aus einer Distanzfunktion $dist(X, Y)$ eine Ähnlichkeitsfunktion $sim(X, Y)$ zu erhalten, können die folgenden Formeln verwendet werden:

Formel	**Quelle**
$sim(X,Y) = \frac{1}{1 + dist(X,Y)}$	[9, 43]
$sim(X,Y) = \frac{1}{e^{dist(X,Y)}}$	[81]

Tabelle 3: Beziehung zwischen Distanz- & Ähnlichkeitsfunktionen.

4.2.4 Auswahl einiger Distanz- und Ähnlichkeitsfunktionen

In der Praxis existiert eine unüberschaubare Anzahl an Distanz- und Ähnlichkeitsfunktionen. Choi et al. fassen in [92] beispielsweise 76 verschiedene Ähnlichkeitsfunktionen alleine nur für binäre Vektoren[1] zusammen. Nachfolgend werden einige bekannte Distanz- und Ähnlichkeitsfunktionen aufgeführt, die auch innerhalb dieser Arbeit verwendet wurden:

Distanzfunktionen:

Funktionsbezeichnung	**Formel**
Minkowski Distanz	$dist_{Minkowski}(X,Y) = \sqrt{\sum_{i=1}^{n} \Big(\lvert x_i - y_i \rvert^{\lambda}\Big)^{\frac{1}{\lambda}}}$
Manhattan Distanz	$dist_{Manhattan}(X,Y) = \sum_{i=1}^{n} \Big(\lvert x_i - y_i \rvert\Big)$
Euklidische Distanz	$dist_{Euclid}(X,Y) = \sqrt{\sum_{i=1}^{n} \Big(x_i - y_i\Big)^2}$
Canberra Distanz	$dist_{Canberra}(X,Y) = \sum_{i=1}^{n} \frac{\lvert x_i - y_i \rvert}{\lvert x_i \rvert + \lvert y_i \rvert}$

Tabelle 4: Eine Auswahl von einigen Distanzfunktionen (zusammengetragen aus: [50, 107]).

Beobachtung: Die Minkowski Distanz ist eine verallgemeinerte Form der Manhattan Distanz ($\lambda = 1$), Euklidischen Distanz ($\lambda = 2$) und weiteren Distanzfunktionen. Die jeweilige Distanzfunktion resultiert somit aus dem gewählten Parameter $\lambda \in \mathbb{R}$. Für den Sonderfall $\lambda = \infty$ ergibt sich die sogenannte Tschebyscheff Distanzfunktion:

$$dist_{Tscheby}(X,Y) = \max\left(\lvert x_1 - y_1 \rvert, \lvert x_2 - y_2 \rvert, \ldots, \lvert x_n - y_n \rvert\right)$$

[1] Es wird vermutet, dass für reele Vektoren deutlich mehr Ähnlichkeitsfunktionen existieren. Eine entsprechende Internet Recherche ergab leider keine verlässliche Auskunft darüber.

Ähnlichkeitsfunktionen für binäre Vektoren:

Funktionsbezeichnung	**Formel**
Kosinus Ähnlichkeit	$sim_{Cosine}(X,Y) = \frac{\alpha}{\sqrt{(\alpha+\beta)(\alpha+\gamma)}}$
Overlap Koeffizient	$sim_{Overlap}(X,Y) = \frac{\alpha}{min(\alpha+\beta, \alpha+\gamma)}$
Jaccard Koeffizient	$sim_{Jaccard}(X,Y) = \frac{\alpha}{\alpha+\beta+\gamma}$
Dice Koeffizient	$sim_{Dice}(X,Y) = \frac{2\alpha}{2\alpha+\beta+\gamma}$

Tabelle 5: Einige Ähnlichkeitsfunktionen für binäre Vektoren (zusammengetragen aus: [39, 92]).

Hierbei haben $\alpha, \beta, \gamma, \delta$ die folgende Semantik bzgl. der Vektorelemente $x_i \in X$ sowie $y_i \in Y$:

- α = Anzahl derjenigen Vektorelemente für die $x_i = y_i = 1$ gilt.
- β = Anzahl derjenigen Vektorelemente für die $x_i = 0, y_i = 1$ gilt.
- γ = Anzahl derjenigen Vektorelemente für die $x_i = 1, y_i = 0$ gilt.
- δ = Anzahl derjenigen Vektorelemente für die $x_i = y_i = 0$ gilt.

Ähnlichkeitsfunktionen für reele Vektoren:

Funktionsbezeichnung	**Formel**
Kosinus Ähnlichkeit	$sim_{Cosine}(X,Y) = \frac{\sum_{i=1}^{n}(x_i y_i)}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$
Overlap Koeffizient	$sim_{Overlap}(X,Y) = \frac{\sum_{i=1}^{n} min(x_i,\ y_i)}{min\Big(\sum_{i=1}^{n} x_i\ ,\ \sum_{i=1}^{n} y_i\Big)}$
Jaccard Koeffizient	$sim_{Jaccard}(X,Y) = \frac{2\sum_{i=1}^{n}(x_i y_i)}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2 - \sum_{i=1}^{n}(x_i y_i)}$
Dice Koeffizient	$sim_{Dice}(X,Y) = \frac{2\sum_{i=1}^{n}(x_i y_i)}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2}$

Tabelle 6: Einige Ähnlichkeitsfunktionen für reele Vektoren (zusammengetragen aus: [39, 92]).

Neben binären und reelen Vektoren lassen sich Ähnlichkeitsfunktionen zusätzlich auch auf Mengen übertragen. Dafür wird die gleiche Definition wie in Kapitel 4.2.2 verwendet, wobei hier anstelle von Vektoren, zwei beliebige Mengen N, M als Definitionsmenge zugrunde liegen. Mengenbasierte Ähnlichkeitsfunktionen werden genau dann benötigt, falls es sich bei den Elementen der Mengen nicht um quantitative, sondern stattdessen um qualitative Features[1] handelt. In der folgenden Tabelle

[1] Siehe dazu Kapitel 7.1.3.2.

werden insgesammt vier solcher mengenbasierten Ähnlichkeitsfunktionen aufgeführt:

Funktionsbezeichnung	**Formel**
Kosinus Ähnlichkeit	$sim_{Cosine}(N, M) = \frac{\lvert N \cap M \rvert}{\sqrt{\lvert N \rvert \cdot \lvert M \rvert}}$
Overlap Koeffizient	$sim_{Overlap}(N, M) = \frac{\lvert N \cap M \rvert}{min(\lvert N \rvert, \lvert M \rvert)}$
Jaccard Koeffizient	$sim_{Jaccard}(N, M) = \frac{\lvert N \cap M \rvert}{\lvert N \cup M \rvert}$
Dice Koeffizient	$sim_{Dice}(N, M) = \frac{2\lvert N \cap M \rvert}{\lvert N \rvert + \lvert M \rvert}$

Tabelle 7: Einige Ähnlichkeitsfunktionen für Mengen (adaptiert aus: [72, Seite: 299]).

4.2.5 Beobachtungen für Distanz- und Ähnlichkeitsfunktionen

Ausgehend von X, Y können die folgenden Beobachtungen festgehalten werden:

- $sim(X, Y) = 1$ impliziert nicht unbedingt, dass X identisch zu Y ist.
- Im Allgemeinen gilt $sim_1(X, Y) \neq sim_2(X, Y)$. In Worten: Das Resultat einer Ähnlichkeitsfunktion sim_1 entspricht im Allgemeinen nicht dem Resultat einer anderen Ähnlichkeitsfunktion sim_{id_2} (Ausnahmen sind hier jedoch durchaus möglich).
- Distanz- und Ähnlichkeitsfunktionen können auf folgende Datentypten angewendet werden:
  - Punkte (in einem metrischen Raum)
  - Binär- und reelwertige Vektoren
  - Mengen

4.3 Klassifikation

Dieser Abschnitt führt in die zentrale Thematik der Klassifikation ein. Dazu wird die Idee dahinter in Worten zusammengefasst und anhand einer Illustration verdeutlicht. Danach werden unterschiedliche Klassifikationsformen erläutert, die sich an die Anzahl der festgelegten Klassen richten. Anschließend werden drei Klassifikationsverfahren vorgestellt, mit deren Hilfe einige Autorschaftsanalyse-Verfahren in dieser Arbeit umgesetzt wurden.

4.3.1 Einleitung

Unter dem Begriff der Klassifikation kann generell die semantische Einteilung von Objekten verstanden werden, die sich durch bestimmte Merkmale voneinander unterscheiden. Betrachtet ein Mensch beispielsweise zwei Objekte $o_1 =$ `Auto` und $o_2 =$ `Flugzeug`, so kann er beide anhand weniger Merkmale (wie etwa Größe oder Form) voneinander unterscheiden. Klassifikationsverfahren in ML folgen dem gleichen Prinzip. Auch hier werden Objekte hinsichtlich ihrer Merkmale kategorisiert, jedoch nicht anhand eines Menschen, sondern automatisiert durch eine Maschine. Um dies zu realisieren, müssen zuvor die Objekte in einer für die Maschine interpretierbare Form übersetzt werden. Hierfür werden die Objekte in Feature-Vektoren überführt, deren Elemente spezifische Eigenschaften der Objekte darstellen. Die Features, die hierfür eingesetzt werden, können dabei sowohl quantitativ oder qualitativ angegeben werden (jedoch nicht gemischt). Die Aufgabe eines Klassifikationsverfahrens (kurz Klassifikators) ist es nun anhand einer Trainingsmenge, welche bereits klassifizierte Objekte in einer Vektorrepräsentation enthält, ein Modell zu erlernen. Dieses besteht dabei aus unterschiedlichen Kriterien die genau festlegen, wie neue (und damit unbekannte) Objekte zu klassifizieren sind. Schematisch kann eine Klassifikation wie folgt illustriert werden:

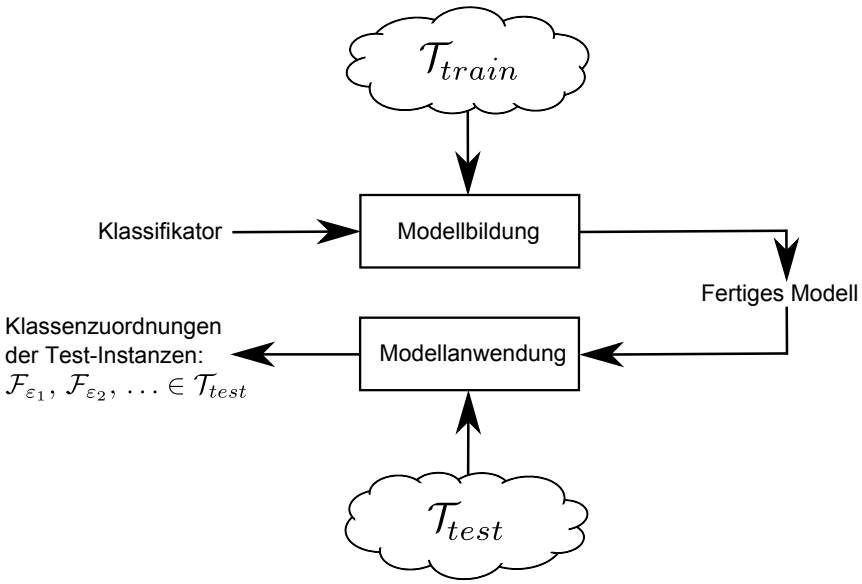


Abbildung 3: Schematische Darstellung einer Klassifikation.

4.3.2 Klassifikationsformen

Je nach Anzahl der Klassen lassen sich in ML drei Klassifikationsformen unterscheiden:

- **Binäre Klassifikation:** Diese stellt die bekannteste und einfachste Form der Klassifikation dar. Die Aufgabe besteht darin Test-Instanzen in maximal zwei möglichen Klassen einzuteilen. Ein typisches Szenario für eine binäre Klassifikation ist z.B. die Spam Erkennung mit den Klassen { `Spam`, `Nicht-Spam` }.

- **Multiklassen Klassifikation:** Bei dieser Klassifikationsform existieren $m > 1$ Klassen zu denen die Test-Instanzen zugeordnet werden sollen. Ein typisches Szenario für eine Multiklassen Klassifikation stellt beispielsweise die Genre-Erkennung von Webseiten dar. Hierbei sind unter anderem folgende Klassen möglich { `Sport`, `Politik`, `Wirtschaft`, ... }.

- **Ein-Klassen Klassifikation:** Bei dieser Klassifikationsform existiert nur eine mögliche Klasse c_1, zu der nur bestimmte Test-Instanzen zugeordnet werden dürfen, während ungeeignete Instanzen dagegen abzuweisen sind. Das Problem hierbei dabei ist, dass nur Trainings-Beispiele für c_1 vorliegen, und es daher äußerst schwierig ist, ungeeignete Instanzen zu erkennen, um diese nicht fälschlicherweise als c_1 zu klassifizieren. In der Praxis existieren zahlreiche Szenarien für die Ein-Klassen Klassifikation. Eine davon stellt z.B. die Erkennung verdächtiger Finanztransaktionen dar. Eine mögliche Klasse wäre hier { `unauffällige Überweisungen` }, sodass Instanzen die außerhalb dieser Klasse liegen Überweisungen darstellen, die beispielsweise zu hohe Beträge enthalten oder zu untypischen Zeiten getätigt wurden.

In der Autorschaftsanalyse sind insbesondere die Ein-Klassen- und Multiklassen Klassifikation vertreten. So kann z.B. die Autorschafts-Attribution mit der Multiklassen Klassifikation gleichgesetzt werden, wobei die möglichen Autoren hier die m unterschiedlichen Klassen $c_1, c_2, \ldots, c_m$ darstellen.

Die Autorschafts-Verifikation stellt dagegen eine Ein-Klassen Klassifikation[1] dar. Hier existiert nur die Klasse c_1, die den zu verifizierenden Autor darstellt. Bei der intrinsischen Exploration handelt es sich ebenfalls um eine Ein-Klassen Klassifikation. Die einzige Klasse c_1 die hierbei vorliegt, repräsentiert in diesem Fall einen konsistenten Schreibstil, sodass Instanzen die außerhalb dieser Klasse liegen stilistische Ausreißer darstellen.

[1] In Kapitel 6.3 sowie Kapitel 6.4 wird dies genauer erläutert.

4.3.3 k-Nearest Neighbor

Der k-Nearest Neighbor (kurz $k-$NN) Klassifikator stellt eines der einfachsten Klassifikationsverfahren in ML dar. Es handelt sich hierbei um ein Verfahren, welches der Lernmethode „Überwachtes Lernen“ zugrunde liegt. Dies bedeutet, dass der Benutzer eine Trainingsmenge mit Instanzen vorgibt, deren Klassenzugehörigkeiten im Vorfeld bekannt sein müssen. Die genauere Arbeitsweise des Verfahrens wird nachfolgend erklärt.

Sei $\mathcal{T}_{train}$, $\mathcal{F}_\varepsilon$ sowie k (die Anzahl der nächsten Nachbarn) gegeben. Die Aufgabe des $k-$NN Klassifikators besteht darin $\mathcal{F}_\varepsilon$ mit Hilfe aller $\mathcal{F}_{j_{c_i}} \in \mathcal{T}_{train}$ sowie einer Metrik (Ähnlichkeits- oder Distanzfunktion) zu ihrer geeignetsten Klasse $c_{max} \in \mathcal{C}$ zuzuordnen. Die Klassifikation erfolgt dabei anhand der folgenden zwei Schritte. Dazu wird zur Vereinfachung von einer Distanzfunktion $dist(\cdot, \cdot)$ als gewählte Metrik ausgegangen:

1. Im ersten Schritt werden zunächst Distanzen zwischen $\mathcal{F}_\varepsilon$ und sämtlichen $\mathcal{F}_{j_{c_i}} \in \mathcal{T}_{train}$ berechnet. Jede berechnete Distanz $d_j = dist(\mathcal{F}_\varepsilon, \mathcal{F}_{j_{c_i}})$ wird darauf mit ihrer korrespondierenden Klasse c_i assoziiert und in einer Folge gespeichert:

$$Distances = \Big((c_1, d_1),\, (c_2, d_2),\, \ldots,\, (c_m, d_m) \Big)$$

2. Im zweiten Schritt müssen die k nächsten Nachbarn von $\mathcal{F}_\varepsilon$ ermittelt werden. Dazu wird die Folge $Distances$ zuerst nach den Distanzen d_j absteigend sortiert, um anschließend die ersten k Tupeln zu selektieren, aus denen dann die Klassenbezeicher c_i entnommen werden. Die extrahierten k Klassenbezeichnungen werden dann ebenfalls in einer Folge abgespeichert:

$$Nearest\,Neighbors = (\, c_1,\, c_2,\, \ldots,\, c_k \,)$$

   Im letzten Schritt gilt es eine Mehrheitsentscheidung bzgl. aller c_i in $Nearest\,Neighbors$ durchzuführen. Diejenige Klasse c_{max} die dabei am häufigsten vorkommt, gewinnt die Mehrheitsentscheidung und wird der Test-Instanz $\mathcal{F}_\varepsilon$ zugeordnet, sodass $\varepsilon = c_{max}$ gilt. Damit ist die Klassifikationsaufgabe abgeschlossen.

Das folgende Beispiel illustriert anhand von vier Abbildungen den gesamten Prozess der $k-$NN Klassifikation und fasst die Schritte in knappen Worten zusammen. Hierbei sind die Klassen c_1, c_2, die Features f_1, f_2 und $k = 3$ gegeben:

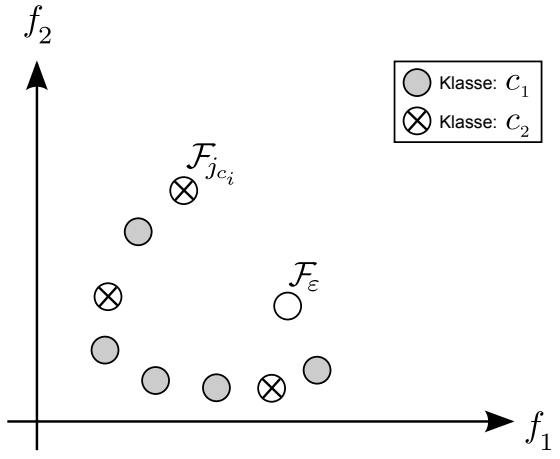


Abbildung 4: Darstellung von $\mathcal{F}_\varepsilon$ und allen $\mathcal{F}_{j_{c_i}}$ im zweidimensionalen Feature-Raum.

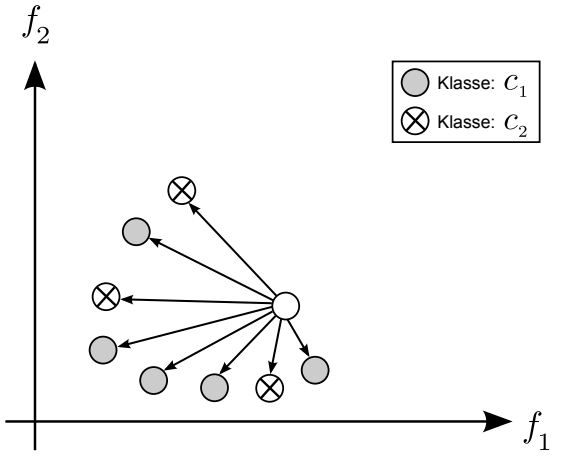


Abbildung 5: Berechnung aller Distanzen zwischen $\mathcal{F}_\varepsilon$ und allen $\mathcal{F}_{j_{c_i}}$

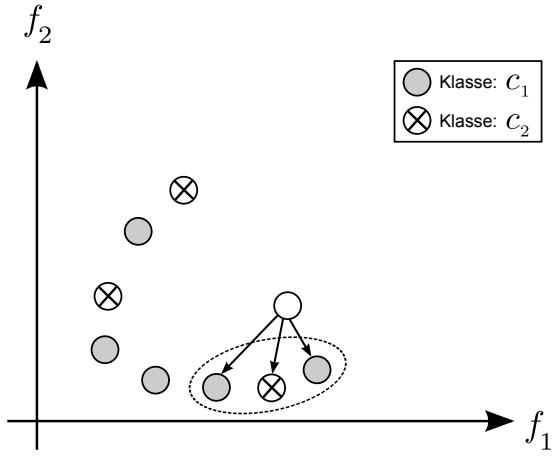


Abbildung 6: Bestimmung der k nächsten Nachbarn von $\mathcal{F}_\varepsilon$ (Ermittlung der k kürzesten bzw. kleinsten Distanzen).

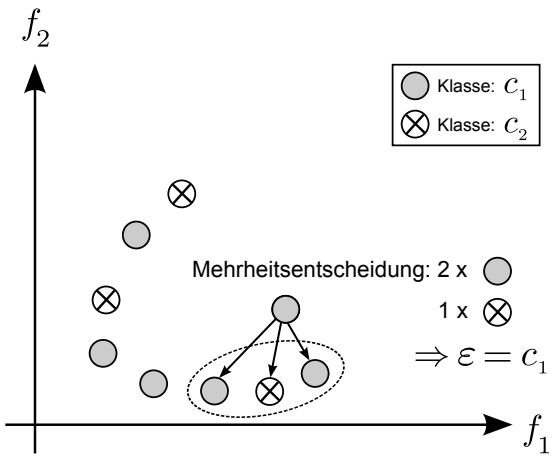


Abbildung 7: Klassifikation von $\mathcal{F}_\varepsilon$ anhand einer Mehrheitsentscheidung der k nächsten Nachbarn.

Anmerkungen:

- **Feature-Typen:** Ein großer Vorteil von k-NN gegenüber anderer Klassifikatoren ist der, dass dieser sowohl mit qualitativen als auch quantitativen Features umgehen kann. Je nachdem welche Features vorliegen, muss eine entsprechende Metrik ausgewählt werden.
- **Schlechte Features:** Der Literatur[1] zufolge ist der k-NN Klassifikator anfällig für verrauschte oder irrelevante Features (im Gegensatz zu anderen Klassifikatoren). Besonders problematisch sind Szenarien, bei denen viele Features vorliegen, aber nur wenige von ihnen tatsächlich relevant sind. Dadurch kann es schnell zu Verzerrungen des Klassifikationsergebnisses kommen. Es empfiehlt sich daher für k-NN eine Strategie einzusetzen, mit der möglichst nur relevante Features für die Klassifikation ausfindig gemacht werden können. Im Kapitel 5.8 werden dazu entsprechende Strategien erläutert.
- **Fehlendes Training/Modell:** In der Literatur wird k-NN öfters als ein „Lazy Learner“ bezeichnet, da sich die Trainingsphase des Klassifikators im Grunde nur auf die Speicherung sämtlicher Trainings-Instanzen in der Menge $\mathcal{T}_{train}$ beschränkt. Die Kernfunktionalität des Verfahrens findet sich daher nicht hier, sondern stattdessem im Klassifikationsschritt wieder, bei dem die Distanzen zwischen $\mathcal{F}_\varepsilon$ und sämtlichen $\mathcal{F}_{j_{c_i}} \in \mathcal{T}_{train}$ berechnet werden, [16]. Durch das fehlende Training entfällt auch die Generierung eines Modells, was gleichzeitig von Vorteil und Nachteil sein kann:
  - **Vorteil:** Es wird stets mit den aktuellsten Instanzen gearbeitet. Dies hat zur Folge, dass zu $\mathcal{T}_{train}$ fortlaufend neue Trainings-Instanzen hinzugefügt bzw. bestehende Instanzen darin verändert werden können, ohne dabei ein erneutes Training im Kauf zu nehmen.
  - **Nachteil:** Ist $\mathcal{T}_{train}$ statisch, sodass keine neuen Instanzen hinzukommen bzw. bestehende Instanzen darin nicht verändert werden, so werden für jede Klassifikation einer neuen Test-Instanz unnötige (gleiche) Berechnungen durchgeführt. Bei anderen Klassifikationsverfahren die ein Modell erzeugen, werden diese dagegen vermieden, sodass nur Features aus der Test-Instanz mit denen des Modells verglichen werden, um dadurch die Klassifikationsentscheidung bestimmen zu können.
- **Wahl von k:** Die Güte des k-NN Klassifikationsergebnisses geht mit der Wahl der k nächsten Nachbarn einher. In der Literatur[2] existieren dazu zahlreiche Empfehlungen, die meistens im

[1] Siehe dazu z.B. [41, Seite: 349] oder [19].

[2] In [71, Seite: 274] werden beispielsweise die Werte 3 oder 5 für k empfohlen.

Bereich $3 \leq k \leq 9$ liegen. Wichtig hierbei ist es k stets ungerade zu wählen, da ansonsten bei einer gleichen Anzahl an Nachbarn mit unterschiedlichen Klassen, keine Mehrheitsentscheidung möglich ist. Um ein brauchbares k in der Praxis zu finden, ist es üblich mehrere Iterationen mit unterschiedlichen $k-$Größen durchzuführen und die Ergebnisse hinsichtlich der vorhergesagten Klassen, sowie der Fehlklassifikationen zu vergleichen, [16]. Allerdings wirkt sich dies negativ auf das Laufzeitverhalten aus.

4.3.4 Naive Bayes

Der Naive Bayes (kurz NB) Klassifikator gehört zu den bekanntesten Vertretern der statistischen Klassifikatoren im ML Umfeld. In der Praxis existieren zahlreiche Anwendungsgebiete bei denen NB ein unentbehrliches Werkzeug darstellt. So wird in der Medizin beispielsweise NB für die Diagnostik vieler Krankheitsbilder (wie etwa Krebs oder Diabetes) verwendet. In der Informatik dagegen, wird NB unter anderem für die Textklassifikation benutzt, wobei insbesondere die Erkennung von Spam-Mails eines der populärsten Beispiele hierbei ist. Der wesentliche Unterschied zu $k-$NN besteht darin, dass die Klassifikation beim NB nicht anhand von Distanz- oder Ähnlichkeitsfunktionen bestimmt wird, sondern stattdessen auf eine Berechnung sogenannter A-priori-Wahrscheinlichkeiten beruht. Ein weiterer Unterschied zu $k-$NN betrifft die verwendeten Features. Während $k-$NN sowohl mit quantitativen als auch qualitativen Features umgehen kann, ist NB nur auf Letztere beschränkt. Falls somit quantitative Features vorliegen, müssen diese zuvor anhand einer geeigneten Strategie diskretisiert werden, damit NB darauf angewendet werden kann. Die genauere Arbeitsweise des NB Klassifikators wird im Folgenden in Anlehnung an Han, [41] und Manning, [71, Seiten: 238–242] beschrieben. Hierbei wurde die Notation entsprechend angepasst.

Ausgehend von einer Trainingsmenge $\mathcal{T}_{train}$ lautet die Aufgabe des NB Klassifikators, ein probabilistisches Modell zu erlernen, mit dessen Hilfe die Klassenzugehörigkeit einer Test-Instanz $\mathcal{F}_\varepsilon$ vorhergesagt werden kann. Die Vorhersage, dass $\mathcal{F}_\varepsilon$ einer bestimmten Klasse $c_i \in \mathcal{C}$ angehört, ist dabei gegeben durch:

$$P(c_i \,|\, \mathcal{F}_\varepsilon) = \frac{P(c_i) \cdot P(\mathcal{F}_\varepsilon \,|\, c_i)}{P(\mathcal{F}_\varepsilon)}$$

Die relevanten Elemente dieser Formel haben dabei die folgenden Bedeutungen:

- $P(c_i \,|\, \mathcal{F}_\varepsilon)$ ist die A-posteriori-Wahrscheinlichkeit (bedingte Wahrscheinlichkeit) der Klasse c_i.
- $P(c_i)$ ist die A-priori-Wahrscheinlichkeit (Anfangswahrscheinlichkeit) der Klasse c_i.
- $\mathcal{F}_\varepsilon$ repräsentiert eine $n-$dimensionale Test-Instanz mit dem Aufbau: $(\, f_1,\, f_2,\, \ldots,\, f_n\,)$, wobei jedes f_k ein nominales Feature (und damit eine Zeichenkette) darstellt.

Aufgrund der Tatsache, dass der Nenner $P(\mathcal{F}_\varepsilon)$ in der oberen Gleichung für alle $c_i \in \mathcal{C}$ konstant ist und damit keinen Einfluss auf die Bestimmung auf $P(c_i \,|\, \mathcal{F}_\varepsilon)$ einnimmt, kann die Gleichung auch wie folgt vereinfacht werden:

$$P(c_i \,|\, \mathcal{F}_\varepsilon) = P(c_i) \cdot P(\mathcal{F}_\varepsilon \,|\, c_i)$$

Das Ziel des NB Klassifikators ist es diejenige Klasse $c_{max} \in \mathcal{C}$ zu finden, die für $\mathcal{F}_\varepsilon$ am wahrscheinlichsten ist. Formal entspricht dies der folgenden Formel:

$$c_{max} = \underset{c_i \in \mathcal{C}}{\arg\max}\, P(c_i \,|\, \mathcal{F}_\varepsilon) = \underset{c_i \in \mathcal{C}}{\arg\max} \Big(P(c_i) \cdot \prod_{k=1}^{n} P(f_k \,|\, c_i) \Big)$$

Die Berechnung der Wahrscheinlichkeiten $P(c_i)$ und einer einzelnen $P(f_k \,|\, c_i)$ erfolgt durch:

$$P(c_i) = \frac{\#(\mathcal{F}_{j_{c_i}}, \mathcal{T}_{train})}{|\, \mathcal{T}_{train} \,|} \quad , \quad P(f_k \,|\, c_i) = \frac{\#(\, f_k, \mathcal{F}_{j_{c_i}})}{\#(\, t, \mathcal{F}_{j_{c_i}})}$$

Dabei gilt die folgende Semantik hinsichtlich der einzelnen Funktionen:

- $\#(\mathcal{F}_{j_{c_i}}, \mathcal{T}_{train})$ beschreibt die Anzahl aller $\mathcal{F}_j$ in $\mathcal{T}_{train}$, die der Klasse c_i angehören.
- $\#(\, f_k, \mathcal{F}_{j_{c_i}}\,)$ beschreibt die Häufigkeit eines Features f_k innerhalb aller Test-Instanzen $\mathcal{F}_j$, die der Klasse c_i angehören.
- $\#(\, t, \mathcal{F}_{j_{c_i}}\,)$ beschreibt die Häufigkeit aller Einheiten t (in der Regel Tokens) innerhalb aller Test-Instanzen $\mathcal{F}_j$, die der Klasse c_i angehören.

Um Fälle zu vermeiden bei denen $P(f_k \,|\, c_i) = 0$ gilt und dadurch die Multiplikation aller bedingten Wahrscheinlichkeiten ebenfalls Null zur Folge hat, kann eine Laplace-Glättung angewendet werden. Diese fügt dem Zähler das Gewicht von 1 und dem Nenner das Gewicht von $|V|$ hinzu, wobei V das Vokabular von $\mathcal{T}_{train}$ und damit, alle eindeutigen Zeichenketten in sämtlichen $\mathcal{F}_{j_{c_i}}$ bezeichnet:

$$P(f_k \,|\, c_i) = \frac{\#(\, f_k, \mathcal{F}_{j_{c_i}}\,) + 1}{\#(\, t, \mathcal{F}_{j_{c_i}}\,) + |V|}$$

Da ein einzelnes $P(f_k \,|\, c_i)$ in der Regel sehr klein ist (z.B. 0.000315), kann durch die Multiplikation aller bedingten Wahrscheinlichkeiten schnell ein Fließkommaüberlauf entstehen. Manning rät daher in [71, Seiten: 238–242] das Produkt von $P(c_i)$ und allen $P(f_k \,|\, c_i)$ zu logarithmieren, um dadurch größere Werte zu erhalten. Durch die Logarithmierung entstehen dabei Summen anstelle von Produkten:

$$\left(P(c_i) \cdot \prod_{k=1}^{n} P(f_k \,|\, c_i) \right) = \left(\log(\, P(c_i)\,) + \sum_{k=1}^{n} \log(P(f_k \,|\, c_i)) \right)$$

Hierbei entstehen bei der Berechnung von $P(c_i)$ und allen $P(f_k \,|\, c_i)$ negative Werte, da die Wahrscheinlichkeiten stets (zumindest in dieser Arbeit) kleiner als 1 sind. Um die Positivität wiederherzustellen, wird daher jedem Logarithmus ein negatives Vorzeichen vorangestellt. Die Klassifikation von $\mathcal{F}_\varepsilon$ erfolgt somit durch die folgende Berechnung:

$$c_{max} = \arg\max_{c_i \in \mathcal{C}} \left(-\log(\, P(c_i)\,) + \sum_{k=1}^{n} -\log(\, P(f_k \,|\, c_i)\,) \right)$$

Damit gilt die Klassifikationsaufgabe als abgeschlossen.

Anmerkungen:

- **Performanz:** Im Gegensatz zu k-NN verhält sich NB deutlich schneller bei der Klassikation der Test-Instanzen. Der Grund dafür ist das $\mathcal{F}_\varepsilon$ nicht mit sämtlichen $\mathcal{F}_{j_{c_i}} \in \mathcal{T}_{train}$ verglichen werden muss, sondern stattdessen auf das erzeugte Modell zurückgegriffen wird, indem die vorab berechneten Wahrscheinlichkeiten vermerkt sind. Dadurch reduziert sich die Laufzeit enorm, da die Klassifikation einer Test-Instanz auf einer simplen arithmetischen Berechnung beruht.
- **Klassikationsqualität:** In der Praxis liefert der NB Klassifikator gute (wenn auch nicht optimale) Ergebnisse. Dieser wird dabei sehr oft als eine Baseline benutzt, um damit andere Klassifikationsverfahren vergleichen zu können (so wie auch in dieser Arbeit).
- **Naivität:** Der NB Klassifikator benutzt eine „naive“ Unabhängigkeitsannahme, welche besagt, dass sämtliche Features voneinander unabhängig sind. Im Bezug auf (natürlichsprachige) Dokumente wird diese Annahme in den meisten Fällen verletzt, da Wörter in der Regel immer von Anderen abhängen.
- **Parametrisierung:** Im Gegensatz zu vielen Klassifikatoren ist beim NB (bis auf die Trainingsmenge und der Test-Instanz) keine Parametrisierung vorhanden. Dadurch entfällt die Suche nach optimalen Parametern, wie beispielsweise das k beim $k-$NN.

4.3.5 Support Vector Machine

Ein weiterer Klassifikator, der ebenfalls der Lernmethode „Überwachtes Lernen“ angehört und heutzutage ein De-Facto-Standard für zahlreiche Disziplinen darstellt, ist der sogenannte Support Vector Machine (kurz SVM[1]) Klassifikator, welcher 1992 von Vladimir Vapnik [41] entwickelt wurde. SVM hat ein breites Anwendungsspektrum, zu dem unter anderem die folgenden Disziplinen zählen:

- Bild- und (Hand-)Schrifterkennung, [25]
- Umsatzprognosen für Unternehmen, [25]
- Textklassifikation, [25]
- Erkennung von Tumoren, [25]
- Sprecheridentifikation, [41]
- Vorhersage von Zeitserien, [41]

Die Grundidee der SVM wird nachfolgend zunächst informell beschrieben, bevor anschließend die mathematische Vorgehensweise in Detail erläutert wird.

Ausgehend von einer Menge von Trainings-Instanzen, die in einem $n-$dimensionalen Feature-Raum angeordnet sind, gilt es bei SVM eine spezifische Hyperebene zu finden, mit deren Hilfe der Raum in genau zwei Bereiche (ober- bzw. unterhalb der Hyperebene) separiert werden soll. Jeder Bereich repräsentiert dabei eine Klasse $c_i \in \{+1, -1\}$. Damit kann festgehalten werden, dass SVM der Klassifikationsform „Binäre Klassifikation“ angehört und damit (zumindest vorerst) nur auf zwei Klassen beschränkt ist. Hierbei repräsentiert $c_1 = +1$ den Bereich oberhalb und $c_2 = -1$ den Bereich unterhalb der Hyperebene. Zwischen diesen Bereichen verläuft ein bestimmter Abstand, der nachfolgend als Korridor bezeichnet wird. Das Ziel von SVM ist es dabei die Breite des Korridors soweit wie möglich zu maximieren, um dadurch die Klassen der Trainings-Instanzen so gut es geht voneinander unterscheiden zu können. Je breiter der Korridor dabei ist, desto generalisierter wird dadurch das Modell, mit dessen Hilfe unbekannte Test-Instanzen klassifiziert werden können. Die folgende Illustrierung zeigt dazu ein Szenario in einem zweidimensionalen Feature-Raum, um sich die Separierung des Feature-Raums bildlich vorzustellen:

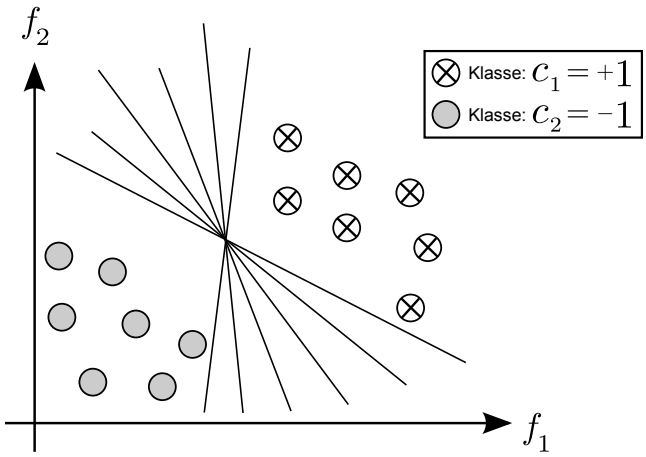


Abbildung 8: Mögliche Separierungen eines zweidimensionalen Feature-Raums.

Wie hieraus ersichtlich wird, existieren unendlich viele Hyperebenen (bzw. im konkreten Fall Geraden), um die Trainings-Instanzen hinsichtlich ihrer Klassenzuordnung voneinander abzugrenzen. Die Frage, die sich hier stellt, ist welche dieser Hyperebenen ein Optimum darstellt. Die Antwort darauf ist der zentrale Gegenstand der SVM. Es geht also darum die bestmögliche trennende Hyperebene zu finden, sodass in jedem Bereich möglichst nur Instanzen liegen, die allesamt einer Klasse c_i angehören. Im Folgenden wird nun die genauere mathematische Arbeitsweise der SVM nach Eberhardt [24], Markowetz [73] und Manning [71, Seiten: 293–321] beschrieben. Dazu wird zunächst, der Einfachheit halber,

[1] Es hat sich eingebürgert auch die Plural-Form SVM's zu schreiben, obwohl eigentlich nur von einem Algorithmus die Rede ist.

die Idee der SVM anhand eines Falls erläutert, bei dem sich der Feature-Raum linear separieren lässt. Im späteren Verlauf wird dann diese Idee auf einen Fall übertragen, bei dem sich der Feature-Raum nicht-linear separieren lässt.

Arbeitsweise einer linearen SVM: Analog zu $k-$NN und NB wird auch bei SVM eine Trainingsmenge $\mathcal{T}_{train}$ sowie eine Test-Instanz $\mathcal{F}_\varepsilon$ benötigt, um die Klassifikation durchzuführen. Das besondere bei der SVM ist, dass die Instanzen inklusive ihrer Klassenbezeichner direkt in die Berechnungen einfließen, die während der Trainingsphase entstehen. Dies schränkt das Verfahren dahingehend ein, dass hier nur numerische Features innerhalb aller Instanzen $\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r$ zulässig und die Klassenbezeichner mit $c_1 = +1$ und $c_2 = -1$ festgelegt sind.

Gegenstand der SVM Klassifikation ist das Finden einer optimal trennenden Hyperebene, wobei eine Hyperebene $\mathcal{H}$ zunächst wie folgt definiert ist:

$$\mathcal{H} = \{\, \mathcal{F}_i \in I \mid \langle w, \mathcal{F}_i \rangle + b = 0 \,\}$$

Hierbei stellt w ein Normalenvektor[1] dar, der senkrecht auf $\mathcal{H}$ steht und in die Richtung des Ursprungs zeigt. Der Term b beschreibt wiederum eine Parallelverschiebung[2] von $\mathcal{H}$ und $\langle \cdot, \cdot \rangle$ stellt dagegen ein Skalarprodukt dar. Der Abstand zwischen dem Ursprung und $\mathcal{H}$ lässt sich mit $\frac{|b|}{\|w\|}$ berechnen, wobei $\|.\|$ die euklidische Norm darstellt. $\mathcal{H}$ kann jedoch weder durch w noch durch b eindeutig bestimmt werden, da sich deren Koeffizienten frei skalieren lassen. Formal bedeutet dies:

$$\forall \mu \in \mathbb{R} \setminus \{\, 0 \,\} \Big(\{\, \mathcal{F}_i \mid \langle w, \mathcal{F}_i \rangle + b = 0 \,\} = \mathcal{H}_1 \Leftrightarrow \mathcal{H}_2 = \{\, \mathcal{F}_i \mid \langle \mu w, \mathcal{F}_i \rangle + \mu b = 0 \,\} \Big)$$

Um eine Eindeutigkeit zu ermöglichen müssen Seperationsbedingungen hinsichtlich der Klassengrenzen angegeben werden:

$$c_i = \begin{cases} +1, & \text{falls } \langle w, \mathcal{F}_i \rangle + b \geq 1 \\ -1, & \text{falls } \langle w, \mathcal{F}_i \rangle + b \leq -1 \end{cases}$$

Diese lassen sich auch zu einer sogenannten kanonischen Hyperebene wie folgt zusammenfassen:

$$c_i(\langle w, \mathcal{F}_i \rangle + b) \geq 1 \text{ für alle } i = 1, 2, \ldots, r$$

Als nächstes gilt es den Korridor ρ, der zwischen den Klassengrenzen verläuft, zu maximieren. Hierfür müssen die Abstände zwischen $\mathcal{H}$ und sämtlichen $\mathcal{F}_i$ anhand der folgenden Abstandsfunktion berechnet werden:

$$d(\mathcal{H}, \mathcal{F}_i) = c_i \Big(\Big\langle \frac{w}{\|w\|}, \mathcal{F}_i \Big\rangle + \frac{b}{\|w\|} \Big)$$

Da in SVM zwei Klassen existieren, müssen die Abstände jeweils für $c_1 = -1$ und $c_2 = +1$ ermittelt werden. Für zwei Trainings-Instanzen $(\mathcal{F}_i, +1)$ und $(\mathcal{F}_j, -1)$ gilt somit zunächst:

$$\langle w, \mathcal{F}_i \rangle + b = +1 \text{ und } \langle w, \mathcal{F}_j \rangle + b = -1$$

Durch Einsetzen in die Abstandsfunktion und anschließender Umformung erhält man für ρ:

$$\Big\langle \frac{w}{\|w\|}, \mathcal{F}_i \Big\rangle + \frac{b}{\|w\|} = \frac{1}{\|w\|} \text{ und } \Big\langle \frac{w}{\|w\|}, \mathcal{F}_j \Big\rangle + \frac{b}{\|w\|} = -\frac{1}{\|w\|}$$

$$\Rightarrow \Big\langle \frac{w}{\|w\|}, \Big(\mathcal{F}_i - \mathcal{F}_j \Big) \Big\rangle = \frac{2}{\|w\|} = \rho$$

Den Korridor $\rho = \frac{2}{\|w\|}$ zu maximieren ist hierbei gleichbedeutend mit der Minimierung von $\frac{\|w\|}{2} = \frac{1}{2}\|w\|$ unter der Nebenbedingung: $c_i(\langle w, \mathcal{F}_i \rangle + b) \geq 1$ für alle $i = 1, 2, \ldots, r$.

---

[1] Im ML-Jargon als *Weight Vector* bezeichnet.

[2] Im ML-Jargon als *Bias* bekannt.

Im nächsten Schritt werden Lagrange-Multiplikatoren $\alpha_i \in \mathbb{R}$ mit $\alpha_i \geq 0$ eingeführt, um dadurch das primale Optimierungsproblem anhand einer Lagrange-Funktion $L(w, b, \alpha)$ zusammenzufassen:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{r} \alpha_i \Big(c_i (\langle w, \mathcal{F}_i \rangle + b) - 1 \Big)$$

Die Lagrange-Funktion wird für w sowie b minimiert und anschließend für die α_i maximiert. Die Minimierung von $L(w, b, \alpha)$ erfordert daher zunächst die folgenden partiellen Ableitungen:

$$(1) \qquad \frac{\partial}{\partial b} L(w, b, \alpha) = 0 \quad \Rightarrow \sum_{i=1}^{r} \alpha_i c_i = 0$$

$$(2) \qquad \frac{\partial}{\partial w} L(w, b, \alpha) = 0 \quad \Rightarrow \sum_{i=1}^{r} \alpha_i c_i \mathcal{F}_i = w$$

Das Einsetzen von (1) und (2) in $L(w, b, \alpha)$ ergibt anschließend das duale Optimierungsproblem:

$$\textbf{Maximierung von:} \qquad W(\alpha) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} \alpha_i \alpha_j c_i c_j \langle \mathcal{F}_i, \mathcal{F}_j \rangle$$

$$\textbf{Nebenbedingungen:} \quad \alpha_i \geq 0 \qquad \text{und} \qquad \sum_{i=1}^{r} \alpha_i c_i = 0$$

Die Lösung des dualen Problems ergibt diejenigen α_i die $W(\alpha)$ maximieren. Damit kann die Berechnung des Normalenvektors w anhand der folgenden Linearkombination angegeben werden:

$$w = \sum_{i=1}^{r} \alpha_i c_i \mathcal{F}_i$$

sodass dadurch diejenige Hyperebene $\mathcal{H}$ gefunden werden kann, welche die größte Korridorbreite aufweist. Die Lage dieser (optimalen) Hyperebene wird dabei durch spezielle Trainings-Instanzen bestimmt, die zu ihr am nächsten liegen und als Stützvektoren[1] SV bezeichnet werden. Die Stützvektoren enthalten als einzige Lagrange-Multiplikatoren mit $\alpha_i \neq 0$, sodass $SV = \{ \mathcal{F}_i \mid \forall i \{ 1, 2, \ldots, r \} (\alpha_i \neq 0) \}$ gilt. Mit anderen Worten, sämtliche $\mathcal{F}_i \in SV$ definieren die Lösung zur Findung von w (und damit gleichzeitig zur Findung der optimalen Hyperebene $\mathcal{H}$), während alle anderen Instanzen keinerlei Einfluß auf die Klassifikation von $\mathcal{F}_\varepsilon$ haben. Aus der Tatsache, dass für alle Stützvektoren $\alpha_i \neq 0$ und für alle anderen Instanzen $\alpha_i = 0$ gilt, kann die Berechnung von w vereinfacht werden zu:

$$w = \sum_{\mathcal{F}_i \in SV} \alpha_i c_i \mathcal{F}_i$$

Anhand von w und einen beliebigen Stützvektor $\mathcal{F}_i \in SV$ kann nun die Parallelverschiebung b durch das Einsetzen in die Nebenbedingung wie folgt berechnen werden:

$$\text{Nebenbedingung:} \quad c_i(\langle w, \mathcal{F}_i \rangle + b) = 1 \quad \Rightarrow \quad b = \frac{1}{c_i} - \langle w, \mathcal{F}_i \rangle$$

Schließlich kann mit w und b die Entscheidungsfunktion classify(.) aufgestellt werden, die der Test-Instanz $\mathcal{F}_\varepsilon$ ihre entsprechende Klasse c_i zuordnet:

$$\text{classify}(\mathcal{F}_\varepsilon) = \text{sign}(\langle w, \mathcal{F}_\varepsilon \rangle + b) = \text{sign}\Big(\sum_{\mathcal{F}_i \in SV} \Big(\alpha_i c_i \langle \mathcal{F}_i, \mathcal{F}_\varepsilon \rangle \Big) + b \Big)$$

[1] Engl. *Support Vectors* bzw. im ML-Jargon *Support Vector Machines*, daher auch die Bezeichnung des SVM Verfahrens.

Hierbei ist die Signumfunktion $\text{sign}(\lambda)$ mit $\lambda \in \mathbb{R}$ gegeben durch:

$$\text{sign}(\lambda) = \begin{cases} +1, & \text{falls } \lambda > 0 \\ 0, & \text{falls } \lambda = 0 \\ -1, & \text{falls } \lambda < 0 \end{cases}$$

sodass die Test-Instanz $\mathcal{F}_\varepsilon$ je nach Ergebniss von $\text{classify}(\mathcal{F}_\varepsilon)$ oberhalb $(+1)$, unterhalb (-1) oder auf (0) der Hyperebene $\mathcal{H}$ fällt. Die folgende Abbildung fasst die wesentlichen Begriffe der SVM Klassifikation schematisch zusammen:

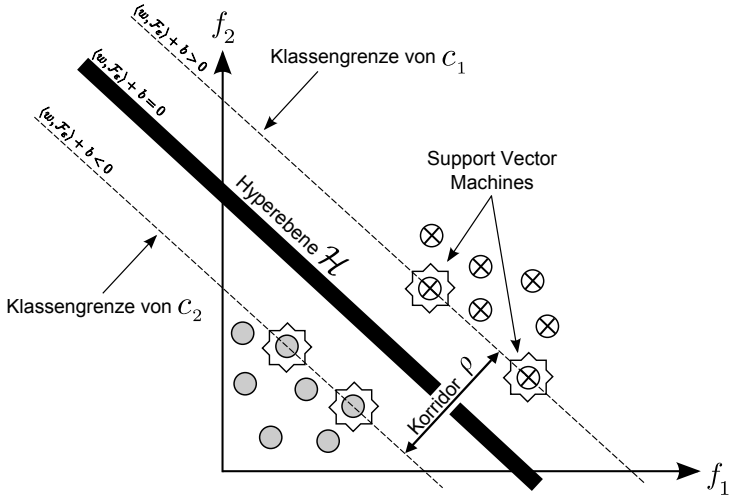


Abbildung 9: Wichtige Begriffe innerhalb der SVM Klassifikation.

Bisher wurde die Idee der SVM anhand eines Falls beschrieben, bei dem sich die Instanzen durch eine Hyperebene $\mathcal{H}$ perfekt trennen ließen, sodass im Bereich oberhalb $\mathcal{H}$ nur Instanzen der Klasse c_1 und unterhalb $\mathcal{H}$ wiederum nur Instanzen der Klasse c_2 angeordnet sind. Da in der Praxis ein solcher Idealfall nur selten vorkommt, muss eine Strategie überlegt werden, wie Ausreißer behandelt werden können, ohne dabei auf die lineare Separierung verzichten zu müssen. Eine Strategie die hier angewendet werden kann, ist die Umformung des bisherigen SVM Modells zu einem „Soft-Margin SVM Modell“. Ein solches Modell ermöglicht es mit Ausreißern umzugehen, indem sogenannte Schlupfvariablen (engl. *Slack Variables*) eingeführt werden. Schlupfvariablen δ_1, δ_2, ... stellen (ausgehend von $\mathcal{H}$) Entfernungen zu Ausreißer dar, die jeweils auf der falschen Klassenseite liegen. Die folgende Abbildung verdeutlicht dies:

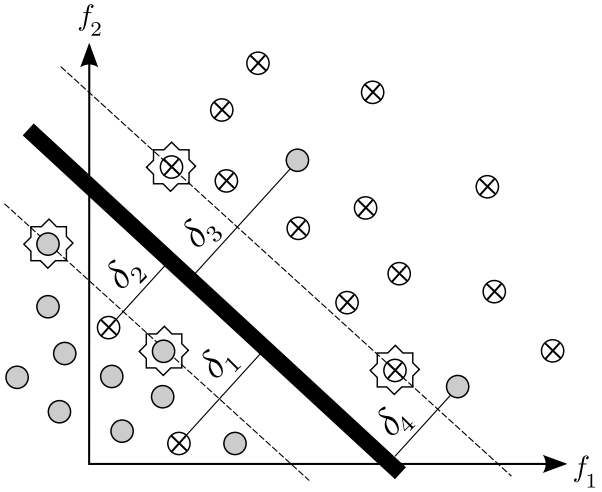


Abbildung 10: Einführen von Schlupfvariablen.

Mit Hilfe der Schlupfvariablen kann die Hyperebene $\mathcal{H}$ in ihrer bisherigen Form bestehen bleiben, selbst wenn einige Ausreißer präsent sind, die eine eindeutige Trennung des Feature Raums verhindern. Dafür muss jedoch ein entsprechendes Fehlergewicht[1] $\vartheta > 0$ in Kauf genommen werden, welches dabei durch den Benutzer reguliert werden kann. Mit der Einführung der δ_1, δ_2, ... ändert sich dementsprechend das Modell, jedoch nur minimal. So muss für die Maximierung des Korridors ρ nicht mehr $\frac{1}{2}\|w\|$, sondern stattdessen die leicht abgewandelte Form minimiert werden:

$$\frac{1}{2}\|w\| + \vartheta \sum_{i=1}^{r} \alpha_i c_i \mathcal{F}_i \text{ , mit der Nebenbedingung: } c_i\big(\langle w, \mathcal{F}_i\rangle + b\big) \geq 1 - \xi_i \text{ für } i = 1, 2, \ldots, r$$

Das duale Optimierungsproblem ändert sich minimal zu:

$$\textbf{Maximierung von:} \qquad W(\alpha) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} \alpha_i \alpha_j c_i c_j \langle \mathcal{F}_i, \mathcal{F}_j \rangle$$

$$\textbf{Nebenbedingungen:} \quad 0 \leq \alpha_i \leq \vartheta \qquad \text{und} \qquad \sum_{i=1}^{r} \alpha_i c_i = 0$$

Wie ersichtlich wird, tauchen hier die Schlupfvariablen nicht auf. Damit kann festgehalten werden, dass der einzige Unterschied zum dualen Optimierungsproblem des vorherigen Modells nur das (konstante) Fehlergewicht ϑ darstellt. Damit kann die SVM nun auf eine Menge von Trainings-Instanzen trainiert werden, die Ausreißer enthält. Die Lösung des dualen Optimierungsproblems lautet damit:

$$w = \sum_{i=1}^{r} \alpha_i c_i \mathcal{F}_i \qquad \text{sowie} \qquad b = \frac{1}{c_i}(1 - \xi_i) - \langle w, \mathcal{F}_i \rangle$$

Die restlichen Schritte sind die gleichen wie beim vorherigen Modell und werden daher nicht aufgeführt.

Nicht-lineare SVM's: In diesem Abschnitt wird der Fall beschrieben, bei dem Trainings-Instanzen derart im Feature Raum angeordnet sind, dass keine lineare Separierung möglich ist. Um diese dennoch zu ermöglichen, kann eine spezielle Abbildung benutzt werden, die es erlaubt Instanzen in eine höhere Dimension zu überführen, in dem eine lineare Trennung vorgenommen werden kann. Die folgende Illustration verdeutlicht die Idee dazu:

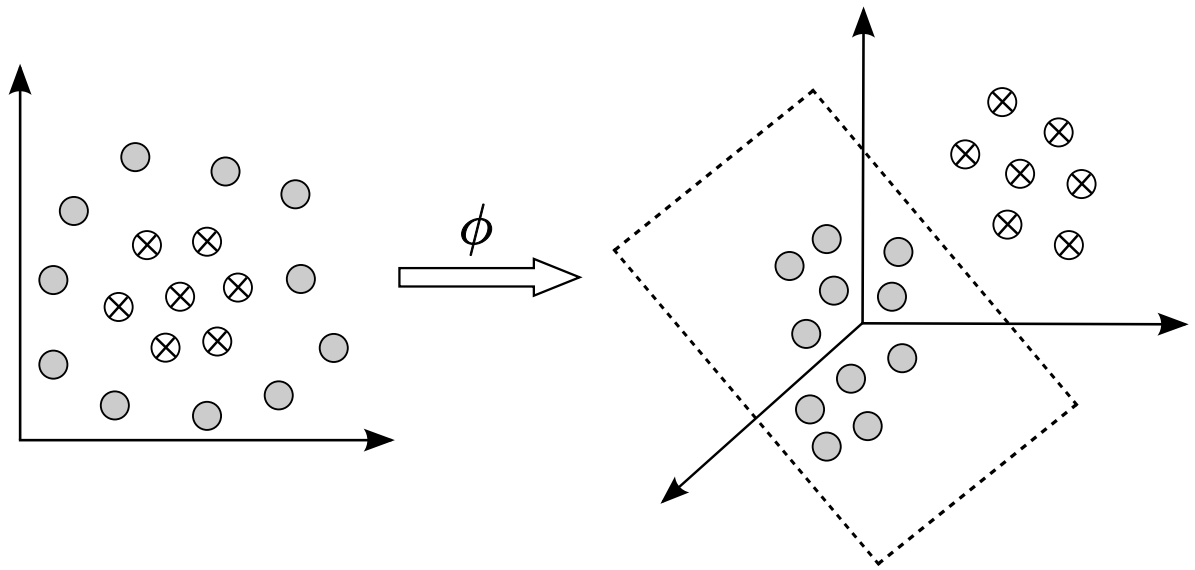


Abbildung 11: Transformation in einem höherdimensionierten Raum mit Hilfe einer Abbildung.

[1] In der Literatur auch bekannt als *Regularization Term*, [71, Seite: 301].

Die Abbildung ϕ ist dabei definiert durch $\phi : \mathbb{R}^r \to \mathbb{R}^s$, $\mathcal{F}_j \mapsto \phi(\mathcal{F}_j)$ für $s > r$. Beispiel: Sei $\mathcal{F}_j \in \mathbb{R}^2$ mit $\mathcal{F}_j = (f_1, f_2)$ und $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ gegeben, dann gilt z.B.:

$$(f_1, f_2) \mapsto (g_1, g_2, g_3) \text{ mit } g_1 = f_1^2,\ g_2 = \sqrt{2} f_1 f_2,\ g_3 = f_2^2$$

Hinweis: Anstelle der konkreten Zielmenge $\mathbb{R}^s$ wird in der Literatur öfters ein beliebiger Hilbert-Raum H angegen, in dem ein Skalarprodukt definiert ist. Daher wird nachfolgend H als Zielmenge verwendet.

Das vorgestellte lineare SVM Modell kann mit Hilfe der Abbildung $\phi : \mathbb{R}^r \to H$ erweitert werden, sodass dadurch Instanzen linear voneinander abgegrenzt werden können, obwohl dies im ursprünglichen Raum nicht möglich war. Die Änderungen die dafür gemacht werden müssen, betreffen zunächst den Normalenvektor w und die Parallelverschiebung b:

$$w = \sum_{\phi(\mathcal{F}_i) \in SV} \alpha_i c_i \phi(\mathcal{F}_i) \quad \text{sowie} \quad b = \frac{1}{c_i}(1 - \xi_i) - \langle w, \phi(\mathcal{F}_i) \rangle$$

Das duale Optimierungsproblem lautet:

$$\textbf{Maximierung von:} \qquad W(\alpha) = \sum_{i=1}^{r} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{r} \alpha_i \alpha_j c_i c_j \langle \phi(\mathcal{F}_i), \phi(\mathcal{F}_j) \rangle$$

$$\textbf{Nebenbedingungen:} \quad 0 \leq \alpha_i \leq \vartheta \qquad \text{und} \qquad \sum_{i=1}^{r} \alpha_i c_i = 0$$

Daraus resultiert für die Entscheidungsfunktion:

$$\text{classify}(\mathcal{F}_\varepsilon) = \text{sign}\big(\langle w, \phi(\mathcal{F}_\varepsilon) \rangle + b \big) = \text{sign}\Big(\sum_{\mathcal{F}_i \in SV} \Big(\alpha_i c_i \langle \phi(\mathcal{F}_i), \phi(\mathcal{F}_\varepsilon) \rangle \Big) + b \Big)$$

Das Modell ist damit sehr ähnlich zu dem der linearen SVM, jedoch ist es hier erforderlich Skalarprodukte der Form $\langle \phi(\mathcal{F}_i), \phi(\mathcal{F}_\varepsilon) \rangle$ zu berechnen, was einen enormen Rechenaufwand mit sich bringen kann. Daher kann anstelle des Skalarprodukts eine Funktion benutzt werden, die zwar in $\mathbb{R}^r$ „existiert“, sich jedoch wie ein Skalarprodukt in H verhält. Diese Funktion wird Kernel genannt und ist wie folgt definiert:

$$K : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R} \quad \text{und der konkreten Abbildungsvorschrift:} \quad (\mathcal{F}_i, \mathcal{F}_j) \mapsto K(\mathcal{F}_i, \mathcal{F}_j) = \langle \phi(\mathcal{F}_i), \phi(\mathcal{F}_j) \rangle$$

In der Praxis existieren viele bekannte Kernelfunktionen, die wichtigsten drei lauten:

Kernel	**Formel**	**Parameter**
Linearer Kernel	$K(\mathcal{F}_i, \mathcal{F}_j) = \langle \mathcal{F}_i, \mathcal{F}_j \rangle$	*keine*
Polynomieller Kernel	$K(\mathcal{F}_i, \mathcal{F}_j) = (\gamma \langle \mathcal{F}_i, \mathcal{F}_j \rangle + \tau)^\beta$	$\beta, \gamma, \tau \in \mathbb{R}$
Gauss-Kernel	$K(\mathcal{F}_i, \mathcal{F}_j) = e^{-\gamma \lVert \mathcal{F}_i - \mathcal{F}_j \rVert^2}$	$\gamma \in \mathbb{R}$

Tabelle 8: Eine Auswahl an Kernelfunktionen (adaptiert aus [73]).

Die Verwendung von Kernels wird in der Literatur manchmal als „Kernel-Trick“ bezeichnet. Der Grund dafür ist, dass die Verwendung von Kernels einer Black-Box Manier gleicht. Das heißt, dass das Interna des Kernels nicht relevant ist, sondern stattdessen nur dessen Ergebniss.

Anmerkungen:

- **Garantiertes Optimum:** Aufgrund der Tatsache, dass beim SVM ein (konvexes) quadratisches Optimierungsproblem gelöst wird, ist das Erreichen eines globalen Minimums (und damit Optimums) stets garantiert. Der Grund dafür ist, dass jede Lösung solcher Optimierungsprobleme stets lokal und gleichzeitig global ist, [24]. Diese Eigenschaft unterscheidet die SVM von vielen anderen Klassifikatoren (z.B. $k-$NN oder Neuronale Netze).

- **Overfitting:** Ein wesentlicher Vorteil von SVM stellt dessen Resistenz gegenüber Overfitting (Überanpassung) dar. Der Grund dafür ist, dass die SVM für das Erlernen eines Modells nur die Stützvektoren benötigt, während die restlichen Trainings-Instanzen unberücksichtigt bleiben.

- **Klassifikation & Regression:** Neben der reinen Klassifikation, kann SVM auch für Regression verwendet werden. Hierbei werden statt diskreter Klassen numerische Werte vorhergesagt, [21, Kapitel: 4].

- **Klassifikationsform:** Ein entscheidender Nachteil von SVM ist, dass hier nur eine binäre Klassifikation möglich ist. Eine Multi-Klassen-Klassifikation ist daher nur über Umwege möglich und stellt ein bis heute noch aktives Forschungsproblem dar. Es existieren zahlreiche Ansätze, um die Funktionalität eines SVM Klassifikators auf $m > 2$ Klassen zu erweitern. Im Folgenden werden die zwei bekanntesten Strategien kurz beschrieben:

  - **One-Against-All:** Bei diesem Ansatz werden insgesamt $m = |\mathcal{C}|$ Entscheidungsfunktionen erzeugt, sodass jede Klasse c_i gegen alle anderen $c_j \in \mathcal{C} \setminus \{c_i\}$ antreten muss. Für $\mathcal{C} = \{ c_1, c_2, c_3 \}$ würde dies wie folgt aussehen: $(c_1$ vs. $\{ c_2, c_3 \})$, $(c_2$ vs. $\{ c_1, c_3 \})$ und $(c_3$ vs. $\{ c_1, c_2 \})$. Die Klassifikation einer Test-Instanz $\mathcal{F}_\varepsilon$ erfolgt hier anhand des Maximums aller Klassen-Vorhersagen, welche durch die m Entscheidungsfunktionen getroffen wurden.
  - **One-Against-One:** Bei diesem Ansatz werden insgesamt $\frac{m(m-1)}{2}$ Entscheidungsfunktionen für jede paarweise Klassenkombination gebildet. Für $\mathcal{C} = \{ c_1, c_2, c_3 \}$ würde dies z.B. wie folgt aussehen: $(c_1$ vs. $c_2)$, $(c_1$ vs. $c_3)$ und $(c_3$ vs. $c_2)$. Die Klassifikation einer Test-Instanz $\mathcal{F}_\varepsilon$ erfolgt hier durch eine Mehrheitsentscheidung hinsichtlich der Klassen-Vorhersagen aller m Entscheidungsfunktionen.

  Welche der beiden Strategien dabei verwendet werden sollte, ist vom jeweiligen Szenario abhängig und kann daher nicht pauschal festgelegt werden. In der Literatur (siehe z.B. Milgram et al. in [75]) wird öfters berichtet, dass für Szenarien mit einer kleinen Klassenanzahl die One-Against-All Strategie bessere Ergebnisse liefern kann als die andere Variante.

- **Training/Erweiterung des Modells:** Analog zu Naive Bayes stellt bei SVM die Änderung/Erweiterung eines Modells eine Schwäche dar, da jede Änderung von $\mathcal{T}_{train}$ ein erneutes Training zur Folge hat. Handelt es sich bei $\mathcal{T}_{train}$ jedoch um eine statische Trainingsmenge, dann zeigt sich die wahre Stärke der SVM, da diese für die Klassifikation nur auf die Stützvektoren zurückgreifen muss und nicht auf Berechnungen bzw. Vergleiche mit allen Trainings-Instanzen in $\mathcal{T}_{train}$ angewiesen ist, wie dies z.B. beim $k-$NN der Fall ist.

- **Wahl des Kernels:** Die Wahl des Kernels für nicht-linear separierbare Instanzen stellt bei SVM ein empirisches Problem dar. In der Forschung ist bisher noch kein Kernel gefunden worden, der effizient und gleichzeitig universell einsetzbar ist. Daher wird öfters auf bekannte Kernels zurückgegriffen, selbst dann, wenn diese für das jeweilige Szenario ungeeignet sind.

- **Interpretation:** Im Gegensatz zu anderen Klassifikatoren kann das Ergebniss der SVM besser interpretiert werden, da sich diese geometrisch veranschaulichen lässt, [24].

4.4 Clustering

Clustering ist ein weiter wichtiger Bestandteil des ML's. Im Gegensatz zur Klassifikation geht es hierbei jedoch nicht darum unbekannte Instanzen zu klassifizieren, sondern stattdessen diese entsprechend ihrer Ähnlichkeit in Cluster zu gruppieren. Der markanteste Unterschied zwischen Klassifikation und Clustering ist, dass beim Letzteren die Klasseneinteilung der Trainings-Instanzen (in der Regel[1]) nicht gegeben ist. Damit gehören Clustering-Verfahren der Lernmethode „Unüberwachtes Lernen“ an.

4.4.1 Clustering-Verfahren

Clustering-Verfahren haben die Aufgabe Instanzen hinsichtlich einer festgelegten Mertrik nach Ähnlichkeiten zu gruppieren. Die Gruppen werden als Cluster bezeichnet und durch $\mathcal{C}_1, \mathcal{C}_2, \ldots$ symbolisiert. Sowohl für die Cluster $\mathcal{C}_i$ als auch die darin befindliche Instanzen $\mathcal{F}_1, \mathcal{F}_2, \ldots$ werden dabei die folgenden zwei Eigenschaften erwünscht:

1. **Inter-Cluster Ähnlichkeit:** Sämtliche Instanzen $\mathcal{F}_x \in \mathcal{C}_i$ sollen sich möglichst signifikant von allen anderen $\mathcal{F}_y \in \mathcal{C}_j$ unterscheiden.
2. **Intra-Cluster Ähnlichkeit:** Sämtliche Instanzen $\mathcal{F}_j \in \mathcal{C}_i$ sollen zueinander eine größtmögliche Ähnlichkeit aufweisen.

Nach Han, [41, Seiten: 398–401] werden unter anderem folgende Clustering-Verfahren unterschieden:

- **Partitionierende Verfahren:** Diese Verfahren unterteilen eine Menge von Instanzen $I = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r\}$ in eine Menge $\mathbb{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ mit $k \leq r$, sodass für jeden Cluster $\mathcal{C}_i$ die folgende Bedingung gilt:

$$\forall i, j \in \{1, 2, \ldots, k\} : \Big((\mathcal{C}_i \subseteq I) \wedge |\mathcal{C}_i| > 0 \Big) \wedge \Big(I = \bigcup \mathcal{C}_i \Big) \wedge (i \neq j) \wedge (\mathcal{C}_i \cap \mathcal{C}_j = \emptyset) \Big)$$

  Partitionierende Verfahren basieren in der Regel auf Distanzfunktionen, wobei jedoch auch Ähnlichkeitsfunktionen zulässig sind. Wichtig hierbei ist zu erwähnen, dass die Instanzen numerische Features enthalten müssen, da diese in Berechnungen einbezogen werden, die oftmals Differenzen oder Durchschnittsbildungen erfordern und diese nur für numerische Merkmale definiert sind. Als Distanzfunktion kommt in vielen Fällen die euklidische Distanz zum Einsatz. Innerhalb dieses Unterkapitels wird eine beliebige Distanzfunktion verwendet und durch $dist(\cdot, \cdot)$ symbolisiert. Der bekannteste Vertreter der partitionierenden Verfahren stellt der $k-$Means Algorithmus dar, der im Anschluß genauer beschrieben wird.

- **Dichtebasierte Verfahren:** Diese Verfahren versuchen eine Menge von Instanzen in Bereiche mit gleicher Dichte zu unterteilen. Die Bereiche verdichten sich dabei, solange die Dichte von Instanzen in ihrer Nachbarschaft einen festgelgten Schwellwert überschreitet. Dichtebasierte Verfahren, die vor allem in der Bildverarbeitung für die Bestimmung von zusammenhängenden Bildsegmenten verwendet werden, haben einige interessante Vorteile gegenüber partitionierenden Verfahren. Zum einen muss hier nicht im voraus die Anzahl der Cluster k vorgegeben werden und zum anderen eignen sie sich gut um Ausreißer zu identifizieren. Der bekannteste Vertreter der dichtebasierten Verfahren stellt der DBSCAN-Algorithmus dar.

- **Hierarchische Verfahren:** Diese Verfahren versuchen eine Menge von Instanzen hierarchisch zu unterteilen. Dazu können zwei unterschiedliche Strategien verwendet werden:

  1. **Bottom-up Strategie:** Hierbei wird jede Instanz zunächst als ein einzelner Cluster betrachtet. In jeder Iteration werden dann ähnliche Cluster zusammengefasst, bis zum Schluss

[1] Es existieren durchaus Clustering-Verfahren bei denen die Klassenzugehörigkeit bekannt ist, dabei handelt es sich häufig um hybride Verfahren bei denen klassische Clustering-Verfahren mit Klassifikatoren kombiniert werden, siehe z.B. [29].

nur ein Cluster übrigbleibt, der alle Instanzen umfasst. In der Regel wird hier eine Abbruchbedingung erwartet, die angibt, ab welchem Schritt ähnliche Cluster nicht mehr zusammengefasst werden.

2. **Top-down Strategie:** Bei dieser Strategie wird im ersten Iterationsschritt ein großer Cluster generiert, der alle Instanzen umfasst. In jeder Iteration wird dieser dann in kleinere Cluster unterteilt, bis zum Schluß in jedem Cluster nur eine Instanz übrigbleibt. Auch hier wird in der Regel ein Schwellwert benötigt der angibt, wann die jeweiligen Cluster nicht mehr aufgeteilt werden sollen.

Unabhängig davon welche der beiden Strategie gewählt wird gilt, dass einmal generierte Cluster, nicht mehr geändert werden können (im Gegensatz zu partitionierenden Verfahren).

Neben diesen Verfahren existieren viele andere, die hier aufgrund des Umfangs nicht vollständig aufgeführt werden können.

4.4.2 $k-$Means

In diesem Abschnitt wird der $k-$Means Algorthmus nach Jiang, [50, Kapitel: Clustering] und Manning, [71, Seiten: 321–338] erläutert, der innerhalb dieser Arbeit für die Realisierung einiger Verfahren verwendet wurde. $k-$Means stellt eines der bekanntesten Clustering-Verfahren in (und außerhalb) des ML Umfelds dar. In der Praxis ist das Verfahren aufgrund zahlreicher Faktoren die erste Wahl, wenn ein partitionierendes Clustering verlangt wird. Die wichtigsten Faktoren sind dabei die Schnelligkeit als auch der einfache Implementierungsaufwand. Im Folgenden werden zunächst die Komponenten und anschließend die Arbeitsweise von $k-$Means im Detail beschrieben.

Ausgehend von einer Trainingsmenge $\mathcal{T}_{train} = \{ \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_r \}$ sowie einem Parameter k gilt es bei $k-$Means eine (disjunkte) Partitionierung von $\mathcal{T}_{train}$ vorzunehmen, sodass zum einem k Cluster $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ entstehen und zum anderen ein Gütekriterium Δ hinsichtlich dieser Cluster minimiert wird, der wie folgt definiert ist:

$$\Delta = \sum_{i=1}^{k} \delta_i \quad \text{, für} \quad \delta_i = \sum_{\mathcal{F}_j \in \mathcal{C}_i} \|\mathcal{F}_j - \mathcal{Z}_i\|^2 \quad \text{, mit} \quad \mathcal{Z}_i = \frac{1}{|\mathcal{C}_i|} \sum_{\mathcal{F}_j \in \mathcal{C}_i} \mathcal{F}_j$$

Als Pseudocode kann der $k-$Means Algorithmus (in dessen Standardversion) wie folgt formuliert werden:

1. Initialisiere k Zentroiden $\mathcal{Z}_1, \mathcal{Z}_2, \ldots, \mathcal{Z}_k$, die die Zentren ihrer zugehörigen Cluster $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ repräsentieren. Zentroide stellen dabei zufällige Feature-Vektoren dar, die in $\mathcal{T}_{train}$ nicht vorhanden sind. Alternativ können bei der Initialisierung auch k Medoide verwendet werden. Diese repräsentieren dagegen bestimmte Feature-Vektoren, die in $\mathcal{T}_{train}$ vorkommen.

2. Bestimme die Cluster $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$ anhand von: $\mathcal{C}_i = \{ \mathcal{F}_j \in \mathcal{T}_{train} \mid i = \underset{i=1,2,\ldots,k}{\arg\min}\, dist(\mathcal{F}_j, \mathcal{Z}_i) \}$.

3. Überschreibe für jeden $\mathcal{C}_i$ den alten Wert des Zentroids $\mathcal{Z}_i$ durch: $\mathcal{Z}_i = \frac{1}{|\mathcal{C}_i|} \sum_{\mathcal{F}_c \in \mathcal{C}_i} \mathcal{F}_c$

4. Wiederhole die Schritte (2), (3) bis eine Abbruchbedingung erfüllt wird. Diese kann dabei wie folgt lauten:

   Terminiere, falls...

   - ... eine festgelegte Anzahl an Iterationsschritten erreicht wird. Diese Ansatz limitiert zwar die Laufzeit, kann jedoch zu einem qualitativ schlechten Clustering-Ergebniss führen.

   - ... sich die Zentroide zwischen zwei Iterationsschritten nicht (oder nur minimal) ändern.

- ... zwischen zwei Iterationsschritten ein Cluster wenige Instanzen zugewiesen bekommt.
- ... Δ einen Schwellwert unterschreitet oder sich zwischen zwei Iterationsschritten kaum unterscheidet.

5. Der $k-$Means Algorithmus terminiert und liefert als Resultat die Zerlegung $\mathbb{C} = \{ \mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k \}$.

Das folgende Beispiel illustriert die Arbeitsweise des $k-$Means Algorithmus für $k = 2$, $\mathcal{Z}_1 = (2, 3)$, $\mathcal{Z}_2 = (3, 1)$ und $\mathcal{T}_{train} = \{ \mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5, \mathcal{F}_6 \}$ mit $\mathcal{F}_1 = (0, 2)$, $\mathcal{F}_2 = (1, 1)$, $\mathcal{F}_3 = (2, 2)$, $\mathcal{F}_4 = (4, 1)$, $\mathcal{F}_5 = (5, 1)$, $\mathcal{F}_6 = (5, 2)$:

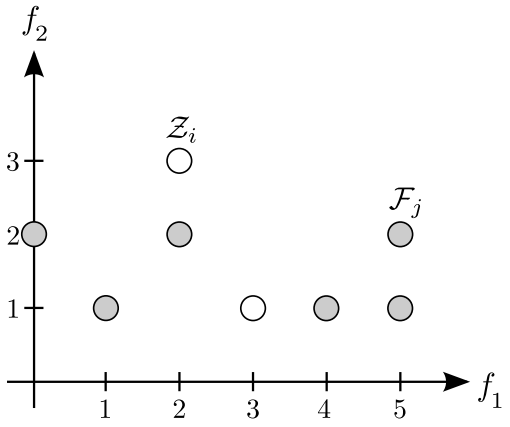


Abbildung 12: $k-$Means: Initialisierung.

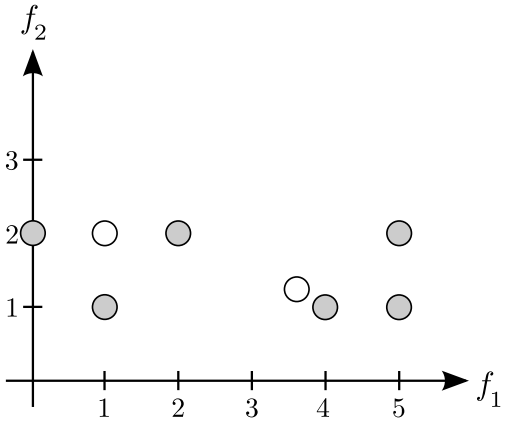


Abbildung 13: $k-$Means: Iteration 1.

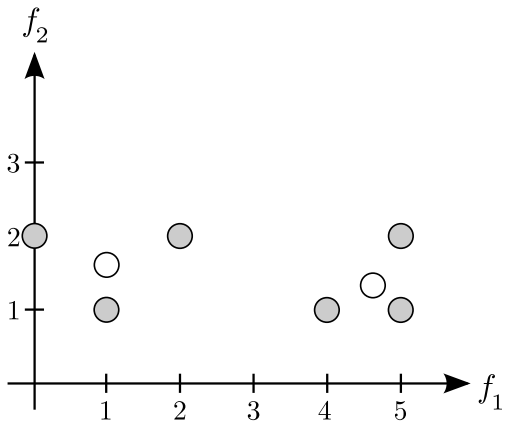


Abbildung 14: $k-$Means: Iteration 2.

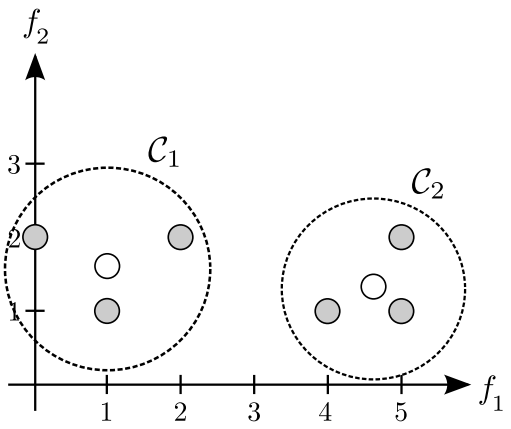


Abbildung 15: $k-$Means: Terminierung.

Anmerkungen:

- **Implementierungsaufwand:** $k-$Means lässt sich dank seiner Kompaktheit leicht implementieren und hinsichtlich nachträglicher Optimierungen in sämtlichen Bereichen (Wahl der Zentroide, Anzahl der Cluster, verwendete Metrik, etc.) modifizieren. In vielen Anwendungsszenarien reicht sogar die Standardversion des Verfahrens aus.
- **Konvergenz:** Ein herausragender Vorteil von $k-$Means ist dessen schnelle Konvergenz zu einem lokalen Minimum. In der Praxis reichen für gewöhnlich wenige Iterationen (z.B. 5 - 10) aus, bis eine Stabilität der Cluster (bzw. Zentroide) gefunden wird, [90].
- **Ausreißeranfälligkeit:** Ein nennenswerter Nachteil von $k-$Means ist der leichtfertige Umgang mit Ausreißern. Da hier sämtliche Instanzen in entsprechende Cluster zugeordnet werden, gilt gleiches auch für Ausreißer, was nicht sinnvoll ist. Dadurch wird z.B. die Eigenschaft Intra-Cluster

Ähnlichkeit verletzt, welche fordert, dass sämtliche Instanzen eines Clusters eine größtmögliche Ähnlichkeit zueinander aufweisen sollen. Als Abhilfe können hierbei Medoiden anstelle von Zentroiden bei der Initialisierung verwendet werden. Handelt es sich bei einem der Medoide um einen Ausreißer, so formt dieser ein einzelnes Cluster dar und wird somit nicht zu einem anderen Cluster zugewiesen.

- **Initialisierung der Zentroide:** Das Clustering-Ergebniss von $k-$Means hängt stark von den k Zentroiden und deren Reihenfolge ab, die zu Beginn initialisiert werden. Um den Einfluss der Zentroiden zu verringern, raten Rothaus et al. in [50, Kapitel: Clustering] $k-$Means mit unterschiedlichen Initialisierungen auf die gleiche Trainings-Instanzen mehrfach anzuwenden und das beste Ergebnis auszuwählen. Eine andere deutlich komplexere Möglichkeit für die Initialisierung der Zentroide stellt das sogenannte *Single Pass Seed Selection* Methode von Pavan et al. dar, welches in [79] nachgeschlagen werden kann.

- **Wahl von k:** Eine große Einschränkung von $k-$Means betrifft dessen Parameter k, welcher im Vorfeld erraten werden muss. Eine mögliche Strategie die hier angewendet werden kann, ist mehrere Durchläufe von $k-$Means mit unterschiedlichen k durchzuführen, wobei für jeden Durchlauf die Struktur des erzeugten Clusterings $\mathbb{C}$ bewertet wird. Die Bewertung kann hierbei z.B. anhand des sogenannten Silhouettenkoeffizienten erfolgen. Im Folgenden wird nach Schubert, [21, Kapitel: Clustering Teil 1] beschrieben, wie dieser berechnet werden kann.

  Sei zunächst der durchschnittliche Abstand einer Instanz $\mathcal{F}$ zu einem Cluster $\mathcal{C}_i$ gegeben durch:

$$a(\mathcal{F}) := dist(\mathcal{F}, \mathcal{C}_i) = \frac{1}{|\mathcal{C}_i|} \sum_{\mathcal{F}'} dist(\mathcal{F}, \mathcal{F}')$$

  Der durchschnittliche Abstand einer Instanz $\mathcal{F}$ zu deren Nachbar-Cluster $\mathcal{C}_j$ sei gegeben durch:

$$b(\mathcal{F}) := \arg \min_{\mathcal{C}_j \in \mathbb{C} \setminus \{\mathcal{C}_i\}} dist(\mathcal{F}, \mathcal{C}_j)$$

  Die Silhouette einer Instanz ist gegeben durch:

$$\mathcal{S}(\mathcal{F}) = \begin{cases} 0 & , \text{falls } \ a(\mathcal{F}) = 0 \\ \frac{b(\mathcal{F}) - a(\mathcal{F})}{max(\, a(\mathcal{F}), b(\mathcal{F})\,)} & , \text{sonst} \end{cases}$$

  Es gilt hierbei $\mathcal{S}(\mathcal{F}) \rightarrow [-1,\ 1]$, wobei für die Werte -1, 0 und 1 aus diesem Intervall folgende Interpretationen gelten:

$$\begin{aligned} \mathcal{S}(\mathcal{F}) \approx 1 \ &\Leftrightarrow\ a(\mathcal{F}) \approx b(\mathcal{F}) \ &\Rightarrow\ \mathcal{F} \text{ liegt im Cluster } \mathcal{C}_i. \\ \mathcal{S}(\mathcal{F}) \approx 0 \ &\Leftrightarrow\ a(\mathcal{F}) \ll b(\mathcal{F}) \ &\Rightarrow\ \mathcal{F} \text{ liegt zwischen } \mathcal{C}_i \text{ und } \mathcal{C}_j. \\ \mathcal{S}(\mathcal{F}) \approx -1 \ &\Leftrightarrow\ b(\mathcal{F}) \ll a(\mathcal{F}) \ &\Rightarrow\ \mathcal{F} \text{ liegt eher im Nachbar-Cluster } \mathcal{C}_j. \end{aligned}$$

  Nun gilt es die Silhouette eines Clusters $\mathcal{C}_i$ zu bestimmen, diese ist dabei gegeben durch:

$$\mathcal{S}(\mathcal{C}_i) = \frac{1}{|\mathcal{C}_i|} \sum_{\mathcal{F} \in \mathcal{C}_i} \mathcal{S}(\mathcal{F})$$

  Anhand dieser kann schließlich der Silhouettenkoeffizient des Clusterings $\mathbb{C}$ (also aller $\mathcal{C}_i$) wie folgt berechnet werden:

$$\mathcal{S}(\mathbb{C}) = \frac{1}{|\mathcal{T}_{train}|} \sum_{\mathcal{C}_i \in \mathbb{C}} \mathcal{S}(\mathcal{C}_i)$$

  Das Resultat von $\mathcal{S}(\mathbb{C})$ kann wiederum wie folgt interpretiert werden:

$$\begin{aligned} 0.70 < \mathcal{S}(\mathbb{C}) \leq 1.00 \ &\Rightarrow\ \mathbb{C} \text{ weist eine starke Struktur auf.} \\ 0.50 < \mathcal{S}(\mathbb{C}) \leq 0.70 \ &\Rightarrow\ \mathbb{C} \text{ weist eine brauchbare Struktur auf.} \\ 0.25 < \mathcal{S}(\mathbb{C}) \leq 1.00 \ &\Rightarrow\ \mathbb{C} \text{ weist eine schwache Struktur auf.} \end{aligned}$$

  Der Fall $0.70 < \mathcal{S}(\mathbb{C}) \leq 1.00$ stellt damit eine gute Wahl für k dar.

4.5 Evaluierungsmöglichkeiten in ML

Um ML-Verfahren evaluieren zu können, existieren unterschiedliche Strategien die davon abhängen, welche Kriterien genau gemessen werden sollen. In diesem Abschnitt werden einige bekannte Evaluierungsmöglichkeiten erläutert, mit deren Hilfe sowohl ML- als auch darauf aufbauende Autorschaftsanalyse-Verfahren hinsichtlich ihrer Güte, bewertet werden können.

4.5.1 Konfusionsmatrix

Der Begriff Konfusionsmatrix[1] bezeichnet eine spezielle Tabelle, deren Zeilen und Spalten alle möglichen Kombinationen einer tatsächlichen und einer vorhergesagten Klasse repräsentierten. Eine Konfusionsmatrix wird in ML des Öfteren verwendet, um binäre Klassifikatoren bewerten zu können. Der Aufbau einer Konfusionsmatrix sieht dabei wie folgt aus:

		Vorhergesagte Klasse	
		c_1	c_2
Tatsächliche Klasse	c_1	TP	FN
	c_2	FP	TN

Tabelle 9: Eine Konfusionsmatrix in ihrer einfachsten Form (adaptiert aus [41, Seite: 361]).

Hierbei besitzen die jeweiligen Elemente TP, FN, FP und TN die folgende Semantik:

- TP (*True Positives*): Tatsächlich positive Beispiele, die als positiv klassifiziert wurden.
- FN (*False Negatives*): Tatsächlich positive Beispiele, die als negativ klassifiziert wurden.
- FP (*False Positives*): Tatsächlich negative Beispiele, die als positiv klassifiziert wurden.
- TN (*True Negatives*): Tatsächlich negative Beispiele, die als negativ klassifiziert wurden.

Anhand dieser Elemente lassen sich nun unterschiedliche Evaluierungsmaße herleiten:

Maß:	**Berechnung:**	**Erläuterung:**
Recall	$\frac{TP}{TP+FN}$	Anzahl der positiv vorhergesagten Beispiele, im Verhältnis zur Gesamtheit der tatsächlich positiven Beispiele.
Precision	$\frac{TP}{TP+FP}$	Anzahl der positiv vorhergesagten Beispiele, im Verhältnis zur Gesamtheit aller positiv vorhergesagten Beispiele.
Specificity	$\frac{TN}{TN+FP}$	Anzahl der negativ vorhergesagten Beispiele, im Verhältnis zur Gesamtheit der tatsächlich negativen Beispiele.
Accuracy	$\frac{TP+TN}{TP+FP+TN+FN}$	Anzahl der positiv vorhergesagten Beispiele, im Verhältnis zur Gesamtheit aller Beispiele.
$F_\beta - Measure$	$\frac{(1+\beta^2)\cdot Recall\cdot Precision}{(\beta^2\cdot Precision)+Recall}$	Harmonisches Maß zwischen Recall und Precision in Abhängigkeit von $\beta \in \mathbb{R}$ (typischerweise wird $\beta = 1$ gewählt).

Tabelle 10: Bekannte Evaluierungsmaße im ML (Formeln adaptiert aus: [41, Seiten: 360-361, 616])

So sinnvoll eine solche Konfusionsmatrix erscheinen mag, so ist diese für zahlreiche Klassifikations-Szenarien, wie etwa die Autorschafts-Attribution nicht anwendbar, da hier meistens $n > 2$ Klassen

[1] In der Literatur manchmal auch „Wahrheitsmatrix“, Kontingenztafel oder Kreuztabelle genannt.

vorliegen. Das Problem lässt sich jedoch lösen, indem die Konfusionsmatrix hinsichtlich der Zeilen und Spalten einfach auf n Klassen erweitert wird. Die folgende Tabelle zeigt dies beispielhaft anhand von n Autoren, welche die Klassen darstellen:

		Vorhergesagter Autor			
		$\mathcal{A}_1$	$\mathcal{A}_2$	...	$\mathcal{A}_n$
Tatsächlicher Autor	$\mathcal{A}_1$	...	...	...	...
	$\mathcal{A}_2$	...	...	...	...
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$\mathcal{A}_n$	...	...	...	...

Tabelle 11: Eine erweiterte Konfusionsmatrix für n Autoren (Klassen).

Wichtig hierbei ist zu erwähnen, dass die Maße aus Tabelle 10 auch für die erweiterte Konfusionsmatrix gelten (wobei einige Umformungen für Recall und Precision benötigt werden).

Anmerkung: Die Wahl welches Evaluierungsmaß eingesetzt werden soll, hängt in der Regel vom jeweiligen Szenario ab. In Kapitel 10.1 wird erläutert, welche Maße für die Evaluierung der implementierten Autorschaftsanalyse-Verfahren innerhalb dieser Arbeit verwendet wurden.

4.5.2 $k-$fache Kreuzvalidierung

Unter dem Begriff $k-$fache Kreuzvalidierung (engl. *$k-$fold Cross-Validation*) wird ein einfaches Verfahren verstanden, welches eine bestehende Menge von Trainings-Instanzen $\mathcal{T}_{train}$ in k (annähernd) gleich große disjunkte Teilmengen $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k$ zufällig aufteilt. Anhand der resultierenden Teilmengen wird dann ein Klassifikator in $i = 1, 2, \ldots, k$ Iterationen getestet. Das folgende Beispiel verdeutlicht die genauere Arbeitsweise der $k-$fachen Kreuzvalidierung.

Beispiel: Sei $M = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_k\}$ die Menge aller disjunkten Teilmengen von $\mathcal{T}_{train}$. In der Iteration $i = 1$ dient $\mathcal{T}_1$ als Testmenge $\mathcal{T}_{test_1}$, während alle anderen $\mathcal{T}_i \in M \setminus \{\mathcal{T}_1\}$ zu einer neuen Trainingsmenge $\mathcal{T}_{train_1} = \mathcal{T}_2 \cup \mathcal{T}_3 \cup \ldots \cup \mathcal{T}_k$ zusammengeführt werden.

In der Iteration $i = 2$ dient wiederum $\mathcal{T}_2$ als Testmenge $\mathcal{T}_{test_2}$, während sämtliche $\mathcal{T}_i \in M \setminus \{\mathcal{T}_2\}$ zu $\mathcal{T}_{train_2} = \mathcal{T}_1 \cup \mathcal{T}_3 \cup \ldots \cup \mathcal{T}_k$ zusammengeführt werden.

Diese Prozedur wiederholt sich solange bis als letzte Testmenge $\mathcal{T}_k = \mathcal{T}_{test_k}$ übrig bleibt. Anschließend werden die Klassifikationsergebnisse gesammelt und entsprechend gemittelt.

Beispiel: Für das *Accuracy* Maß in einem Szenario mit zwei Klassen ergeben sich zunächst die folgenden Teilergebnisse:

$$Accuracy_1 = \frac{TP_1 + TN_1}{TP_1 + FP_1 + FN_1 + TN_1}$$

$$Accuracy_2 = \frac{TP_2 + TN_2}{TP_2 + FP_2 + FN_2 + TN_2}$$

$$\vdots$$

$$Accuracy_k = \frac{TP_k + TN_k}{TP_k + FP_k + FN_k + TN_k}$$

Im nächsten Schritt erfolgt die Mittelung der Teilergebnisse:

$$Overall\ Accuracy = \frac{1}{k} \sum_{i=1}^{k} Accuracy_i$$

Das Gesamtergebnis ergibt damit die Güte des Klassifikators hinsichtlich des betroffenen Maßes und kann so als Vergleichszahl gegen andere Klassifikatoren (bzw. Modelle) benutzt werden.

Falls einige $Accuracy_i$ sigifikante Ausreißer darstellen, kann auch anstelle des Durchschnitts ein anderer (robusterer) Lageparameter, wie z.B. der Median verwendet werden:

$$Overall\,MED\,Accuracy = \text{Median}(\,Accuracy_1,\,Accuracy_2,\,\ldots,\,Accuracy_k\,)$$

Sonderfälle: Neben der Standardversion der $k-$fachen Kreuzvalidierung existieren zusätzlich noch einige Sonderfälle. Der Letztgenannte wurde dabei im Rahmen der Experimente in Kapitel 10 eingesetzt:

- ***Holdout Cross-Validation:*** Bei dieser Methode wird $k = 1$ gewählt, sodass hier eine einzige Aufteilung in Training- und Testmenge stattfindet. Der wohl einzige Vorteil dieser Strategie betrifft die Schnelligkeit, mit der die Instanzen getestet werden können, sodass vor allem größere Trainingsmengen schneller verarbeitet werden können, als bei anderen Strategien. Der Nachteil dieser Methode ist jedoch, dass nur ein Teil der Trainings-Instanzen „gesehen“ wird und dadurch kein verlässlches Generalisierungsergebniss bestimmt werden kann.

- ***Stratified Cross-Validation:*** Hierbei wird keine zufällige Aufteilung hinsichtlich der Trainings-Instanzen vorgenommen, sondern stattdessen sichergestellt, dass jede Teilmenge $\mathcal{T}_i$ eine gleiche Klassenverteilung aufweist, [96].

- ***Leave-One-Out:*** Nach Sporleder, [96] stellt dieser einen Extremfall der Kreuzvalidierung dar, bei dem $k = |\mathcal{T}_{train}|$ gilt. Meistens wird diese Strategie bei solchen Szenarien angewendet, in denen nur sehr wenige Trainingsdaten vorliegen. Der Vorteil dabei ist, dass das Ergebnis wenig verzerrt wird, da mit allen verfügbaren Instanzen getestet wird. Ein Nachteil dieser Strategie ist jedoch der sehr hohe Rechenaufwand, der für größere Trainingsmengen nur in Ausnahmefälle vertretbar ist.

5 Features

Im Kapitel 2.5 wurden Features kurz und oberflächlich erläutert. Da diese jedoch den wesentlichen Kern der Autorschaftsanalyse ausmachen, wird in diesem Kapitel detailierter auf die Thematik der Features eingegangen. Von zentraler Bedeutung sind hierbei die sogenannten Feature-Vektoren, die die Repräsentanten der Autorenstile darstellen. Im folgenden Unterkapitel wird dazu eine schematische Darstellung gegeben, die alle benötigten Komponenten umfasst, mit denen sich Feature-Vektoren erstellen lassen. Jede beteiligte Komponente wird dazu gesondert in einem eigenen Unterkapitel beschrieben, um so den sequentiellen Ablauf der Feature-Vektor Generierung aufzuteilen.

5.1 Feature-Vektor Generierung

Wie eingangs erwähnt, werden Features benötigt, um einen Autorenstil approximieren zu können. Hierfür muss zunächst ein Dokument $\mathcal{D}$, welches die Features in sich trägt, in unterschiedliche sprachliche Ebenen überführt werden. Die Überführung erfolgt dabei anhand der NLP-Werkzeugen aus Kapitel 3.2. Aus den resultierenden sprachlichen Ebenen werden im nächsten Schritt Features mit Hilfe von regulären Ausdrücken entnommen. Durch die Entnahme entstehen so zahlreiche Features, die in Feature-Ebenen bzw. in Feature-Kategorien eingeordnet werden. Nachdem die Einordnung der Features erfolgt ist, gilt es diese numerisch auszudrücken. Hierfür wird zunächst für jedes Feature eine absolute Häufigkeit berechnet, die ausdrückt, wie oft dieser in $\mathcal{D}$ vorkommt. Da sich absolute Häufigkeiten jedoch nicht eignen um „faire“ Stilvergleiche zwischen Dokumenten durchzuführen, müssen im nächsten Schritt diese anhand einer Feature-Normalisierung, auf ein einheitliches Intervall skaliert werden. Aufgrund der Tatsache, dass viele Features redundant sind oder keine Aussage besitzen, empfiehlt sich des Weiteren eine Feature-Auswahl unter Zuhilfenahme von Feature-Extraktions- und/oder Feature-Selektionsalgorithmen, um somit relevante Features ausfindig zu machen. Alternativ bzw. zusätzlich dazu, kann eine Feature Gewichtung durchgeführt werden, die es z.B. erlaubt Features mit mehr Aussagekraft höher zu gewichten, um dadurch die Varianz zwischen den einzelnen Feature-Vektoren zu erhöhen. Das Resultat der sequentiell abgearbeiteten Schritte (bzw. Komponenten) ergibt schließlich einen Feature-Vektor, der wiederum einen Autorenstil repräsentiert. Die folgende Abbildung zeigt den schematischten Aufbau der beteiligten Komponenten:

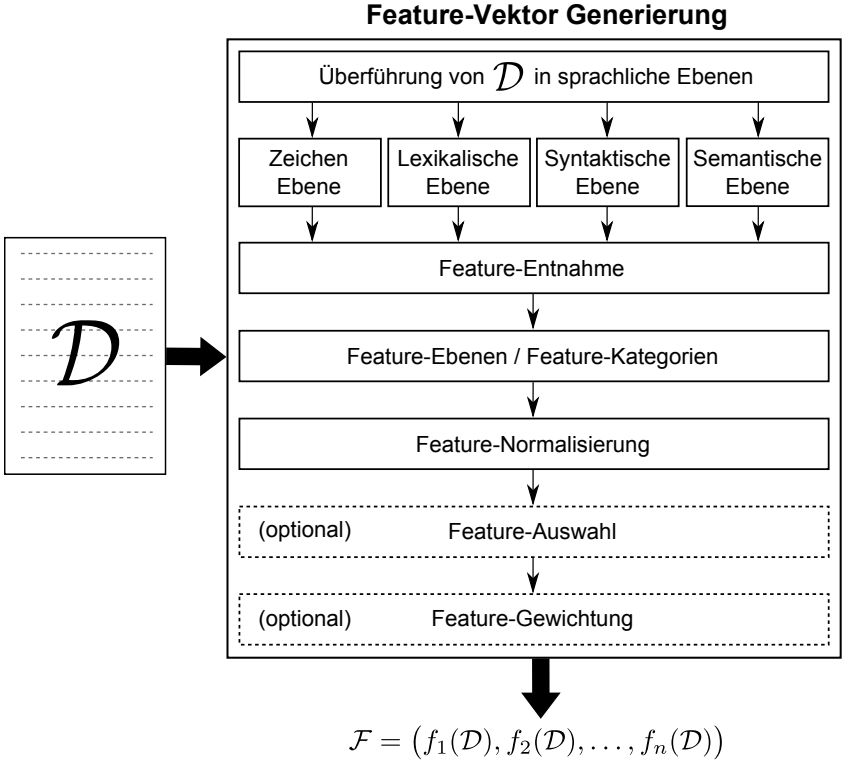


Abbildung 16: Ablauf der Feature-Vektor Generierung aus einem Dokument $\mathcal{D}$.

5.2 Überführung in sprachliche Ebenen

Es existieren unterschiedliche Möglichkeiten, wie Features aus einem Dokument $\mathcal{D}$ entnommen werden können. Die einfachste Form dabei ist diese direkt aus dem Rohtext zu entnehmen. Ein andere Möglichkeit besteht darin $\mathcal{D}$ zunächst in sprachliche Ebenen zu überführen, um anschließend aus diesen Ebenen Features zu entnehmen, die im ursprünglichen Rohtext nicht zugänglich waren. Innerhalb dieser Arbeit werden beide Möglichkeiten betrachtet, wobei im Folgenden die zweite Möglichkeit kurz beschrieben wird.

Ausgehend von den NLP-Werkzeugen aus Kapitel 3.2 kann $\mathcal{D}$ in unterschiedliche Formen überführt werden. Die Formen approximieren dabei die „echten“ sprachlichen Ebenen, da diese nur schwer bzw. unvollständig abgebildet werden können. Die folgende Tabelle fasst zusammen, wie die sprachlichen Ebenen innerhalb dieser Arbeit anhand von NLP-Werkzeuge und Wortlisten approximiert werden:

Sprachliche Ebene	**NLP-Werkzeug / Wortlisten**
Morphologie	Kombination aus Stemmer, reguläre Ausdrücke, $n-$Gramme und Wortlisten (Affixe und freie Stamm-Morpheme)
Syntax	Tokenizer, POS-Tagger (um die Wortarten der Tokens zu erkennen), Sentence Tokenizer, Chunker (um Konstituenten zu erkennen), Parser (um die vollständige Satzstruktur zu erkennen)
Semantik	Kombination von Parser, reguläre Ausdrücke und Synonymwortlisten (um z.B. Synonyme oder Redewendungen zu erkennen)

5.3 Feature-Entnahme

Nachdem $\mathcal{D}$ in sprachliche Ebenen aufgeteilt wurde, gilt es im nächsten Schritt aus diesen Ebenen Features zu entnehmen. Hierfür werden in dieser Arbeit (fast) ausnahmslos reguläre Ausdrücke verwendet. Diese haben den Vorteil, dass sie auf beliebige Zeichenketten angewendet werden können. Als Zeichenketten kommen neben Morpheme, Wörter, Phrasen oder Sätze auch Wortarten bzw. Wortarten $n-$Gramme in Frage. Auf der Seite 25 wurde ein Beispiel gezeigt, wie das Resultat einer Feature-Entnahme aussieht.

5.4 Feature-Ebenen

Nachdem die Features aus den sprachlichen Ebenen entnommen wurden, müssen (bzw. sollten) diese entsprechend eingeordnet werden. In der Literatur existieren hierfür nach [99, Seiten: 3–10] und [68] insgesamt sechs verschiedene Feature-Ebenen:

1. **Zeichen Ebene:** Diese Ebene setzt sich aus zeichenbasierten Features zusammen, wie etwa Buchstaben, Symbole, Affixe oder Buchstaben $n-$Gramme.

2. **Lexikalische Ebene:** Diese Ebene beinhaltet wort- bzw. tokenbasierte Features. Dazu zählen beispielsweise Types, Stemms, Funktionswörter, Komposita oder auch tokenbasierte Maßzahlen wie Wortlängen oder die Anzahl von Vokalen in Tokens.

3. **Syntaktische Ebene:** Diese Ebene enthält satzbasierte Features wie beispielsweise POS-Tags, Syntaxfehler, oder Chunks (z.B. Nominalphrasen, Verbalphrasen, etc.).

4. **Semantische Ebene:** Diese Ebene beinhaltet semantikbasierte Features wie z.B. semantische Abhängigkeiten (Kookkurrenzen) oder semantische Relationen (z.B. Synonymie, Paraphrase, Antonymie, Hyperonymie/Hyponymie, Meronymie, etc.).

5. **Layout Ebene:** Diese Ebene beinhaltet laut Stamatatos [99, Seite: 10] anwendungsspezifische Features. Dazu zählen beispielsweise Einrückungen, verwendete Schriftarten/Schriftgrößen, Signaturen oder ähnliche Layout-Merkmale.

6. **Phonem Ebene:** Diese Ebene beinhaltet laut Loose [68] phonemenbasierte Features, die unter anderem Phonem-N-Gramme, Vokal-N-Gramme oder auch Vokal-Konsonant-N-Gramme einschließen. Derartige Features können nicht in die Ebenen $1-5$ eingeordnet werden, da sie nicht im Text vorkommen, sondern erst durch eine Lautschrift-Umwandlung sichtbar gemacht werden müssen. Eine solche Umwandlung kann beispielsweise mit Hilfe eines Lautschrift-Wörterbuchs[1] durchgeführt werden.

Im Rahmen dieser Arbeit werden nur die Feature-Ebenen $1-4$ betrachtet, die in ihrem Umfang bereits ausreichen, um eine erfolgreiche Autorschaftsanalyse durchzuführen.

5.5 Feature-Kategorien

Die genannten Ebenen aus den vorherigen Abschnitt dienen dazu, Features in „abstrakte Strukturen“ einzuteilen. Um eine verfeinerte Einordnung dieser Strukturen zu ermöglichen, werden innerhalb dieser Arbeit konzeptuelle Kategorien eingeführt. Die Idee dabei ist, den gesamten Feature-Raum $\mathbb{F} = \{ f_1, f_2, \dots \}$ in eine kompaktere Form $\mathbb{F} = F_1 \cup F_2 \cup \dots$ darzustellen. Jedes F_i repräsentiert somit eine konzeptuelle Feature-Kategorie mit einer eindeutigen Bezeichnung. In den nächsten Unterabschnitten werden dazu insgesamt 13 Feature-Kategorien vorgestellt, die neben Kurzbeschreibungen und Beispielen auch Zuordnungen zu ihren entsprechenden Feature-Ebenen enthalten. (hierbei kann eine Kategorie durchaus mehreren Feature-Ebenen angehören). Um die Zuordnung mathematisch formulieren zu können, werden sowohl die Kategorien als auch die Feature-Ebenen als Mengen aufgefasst.

5.5.1 F_1 : Interpunktion

Diese Kategoerie fasst sämtliche Zeichen zusammen, die Interpunktionssymbole darstellen. Ein Interpunktionssymbol kann dabei als ein Satzzeichen verstanden werden, dessen primäres Ziel es ist Sätze zu strukturieren bzw. bestimmte Satzglieder hervorzuheben. Die folgende Tabelle zeigt einen Auszug der bekanntesten Interpunktionssymbole im Deutschen:

.	,	:	;	?	!	-	/	\	(	)	{	}	[	]

Zuordnung: $F_1 \subset$ (Zeichen Ebene).

5.5.2 F_2 : Buchstaben

Diese Kategoerie beinhaltet sämtliche Buchstaben des deutschen Alphabets, wobei zwischen Groß- und Kleinschreibung unterschieden wird:

A	B	...	Z	Ä	Ö	Ü	a	b	...	z	ä	ö	ü	ß

Zuordnung: $F_2 \subset$ (Zeichen Ebene).

5.5.3 F_3 : Buchstaben $n-$Gramme

Diese Kategoerie beinhaltet ausschließlich Buchstaben $n-$Gramme, wobei stets $n > 1$ gilt. Es werden hierbei also nur Bigramme, Trigramme, Tetragramme etc. betrachtet (ausgenommen Unigramme):

$n = 2$	$n = 3$	$n = 4$	$n = 5$	$n = 6$	...
ab	abc	abcd	abcde	abcdef	...

Zuordnung: $F_3 \subset$ (Zeichen Ebene). Je nach Größe von n gilt auch:
$F_3 \subset$ (Zeichen Ebene $\cup$ Lexikalische Ebene).

[1] Siehe dazu z.B. die Webseite des *International Phonetic Association* (`www.LangSci.ucl.ac.uk/ipa`).

5.5.4 F_4: Funktionswörter

Diese Kategoerie enthält ausschließlich Wörter, die über keinen Informationsgehalt verfügen. Zu dieser Art von Wörtern zählen z.B. Artikel, Konjunktionen, Partikel, Präpositionen oder auch Modalwörter:

Artikel				**Konjunktionen**			**Partikel**			**Präpositionen**				**Modalwörter**		
der	die	das	...	und	oder	...	etwa	nur	...	an	auf	in	...	gewiss	leider	...

Zuordnung: $F_4 \subset$ (Lexikalische Ebene).

5.5.5 F_5: Wort-Komplexität

Diese Kategoerie beinhaltet spezifische Komplexitäts-Merkmale von Wörtern. Dazu zählen unter anderem die Vokalanzahl innerhalb bestimmter Wörter, spezifische Wortlängen oder auch die Anzahl von Präfixen bzw. Suffixen, welche durch Derivation (Ableitung) erzeugt wurden. Besonders hervorzuheben ist hier die Wortform der Nomen, die im Deutschen dank der flexiblen Kompositionsmöglichkeiten sehr lang werden können. In der folgenden Tabelle finden sich einige Beispiele für diese Kategorie:

Nomen Wortlänge (34 Buchstaben)	**Nomen (mit 5 Vokale)**	**Derivations-Suffix (mit Länge 2)**
Donauschifffahrtspolizeiverordnung	Buchstabensalat	Grusel ⇒ grusel[ig]

Zuordnung: $F_5 \subset$ (Zeichen Ebene $\cup$ Lexikalische Ebene).

5.5.6 F_6: Phrasen

Diese Kategoerie beinhaltet sämtliche Features, die aus $n \geq 2$ Tokens bestehen. Dazu zählen unter anderem Kollokationen, Token $n-$Gramme, Aufzählungen oder auch Wortzusammensetzungen, die mittels Bindestrich verbunden sind. Features, die in dieser Kategorie nicht abgedeckt werden, sind abgeschlossene Sätze (mit Ausnahme von sehr kurzen Sätzen, die selbst eine Phrase darstellen).

Wörter mit Bindestrich		**Kollokationen**		**Token 3−Gramm**
Top-CD-Spieler	Computer-Tastatur	auf hoher See	Gesetz verabschieden	Er wusste nicht

Zuordnung: $F_6 \subset$ (Lexikalische Ebene $\cup$ Syntaktische Ebene $\cup$ Semantische Ebene).

5.5.7 F_7: POS-Tags

Diese Kategorie beinhaltet alle möglichen POS-Tag Annotationen von Tokens, zu denen unter anderem die Folgenden zählen:

Artikel	**Konjunktion**			**Verben**				**Adverbien**	**Partikel**		
ART	KOUI	KOUS	KON	VVFIN	VAFIN	VVIZU	...	ADV	PTKANT	PTKNEG	...

Zuordnung: $F_7 \subset$ (Syntaktische Ebene).

5.5.8 F_8: POS-Tag $n-$Gramme

Diese Kategoerie enthält Aneinanderreihungen von Wörtern, die durch ihre POS-Tag Darstellung repräsentiert werden. Im Rahmen dieser Arbeit wurden ausschließlich POS-Tag Bi- bzw. Trigramme betrachtet. Es ist jedoch durchaus möglich n beliebig groß zu wählen, wobei jedoch der Recall mit zunehmder Größe abfällt. Die folgende Tabelle zeigt einige Beispiele zu POS-Tag Bi- und Trigrammen:

Bigramme		**Trigramme**	
(APPR PIDAT)	(APPR PPOSAT)	(ART ADJA NN)	(ADV VMFIN ART)

Zuordnung: $F_8 \subset$ (Syntaktische Ebene).

5.5.9 F_9: Satz-Anfänge/Endungen

Diese Kategorie enthält sämtliche Wörter in ihrer POS-Tag Darstellung, die jeweils am Anfang oder am Ende innerhalb eines Satzes vorkommen.

Satzanfang		Satzende	
(ART)	(APPRART)	..... (NN)	..... (VVINF)

Zuordnung: $F_9 \subset$ (Syntaktische Ebene).

5.6 Eigene Features

In diesem Abschnitt werden eigene Features präsentiert, die in der Literatur[1] noch nicht berücksichtigt wurden. Diese werden, analog zu den Kategorien aus dem vorherigen Kapitel, den übergeordneten Feature Ebenen zugeordnet.

5.6.1 F_{10}: Grammatikalische Fehler

Die Idee für diese Kategorie basiert auf die Beschreibung aus [99, Seite: 4–9]. Stamatatos berichtet darin von Fehlern, die als Features verwendet wurden, wobei es sich vorwiegend um Rechtschreibfehler oder um vereinzelte syntaktische Fehler (z.B. nicht übereinstimmende Zeitformen) handelt. Im Gegensatz dazu enthält diese Kategorie weitere Formen von Fehlern, die speziell für die deutsche Grammatik ausgelegt sind. Zu diesen zählen unter anderem falsche Kommasetzungen, Bindestrichfehler, vergessene oder falsch gesetzte Fugenelemente, falsche Kasusmarkierungen (für Dativ, Genitiv, Akkusativ und Nominativ), falsche Genusmarkierungen, falsche Verbstellung im Hauptsatz, Subjekt-Verb-Kongruenz[2], etc. Die folgende Tabelle zeigt zur Verdeutlichung einige Beispiele:

	Bindestrichfehler		Subjekt-Verb-Kongruenz	Vergessene Fugenelemente	
Korrekt:	Nordamerika	Motorradfahren	Ein Kilo Äpfel kostet 3 €.	Arbeitsamt	Tagebuch
Falsch:	Nord-Amerika	Motorrad-fahren	Ein Kilo Äpfel kosten 3 €.	Arbeitamt	Tagbuch

Zuordnung: $F_{10} \subset$ (Lexikalische Ebene $\cup$ Syntaktische Ebene).

5.6.2 F_{11}: Anglizismen

Diese Kategorie beinhaltet Wörter oder Komposita (Wortzusammensetzungen), deren Ursprung aus dem englischen Sprachraum entstammt. Zu dieser Kategorie zählen z.B. die folgenden Wörter:

Babysitter	Decoder	Hacker	Mail	Paper	Sandwich	...

Zuordnung: $F_{11} \subset$ (Lexikalische Ebene).

5.6.3 F_{12}: Redewendungen/Geflügelte Worte

Diese Kategorie beinhaltet Features, die in Form von festen Wendungen vorkommen. Sie unterscheidet sich von den Phrasen Kategorie dadurch, dass die Features hier in der Regel eine abgeschlosse Semantik-Einheit bilden und dadurch eigenständige Sätze darstellen können. Die folgende Tabelle zeigt einige Beispiele für Redewendungen bzw. geflügelte Worte:

Redewendungen		Geflügelte Worte	
Wie es im Buche steht	Unter Dach und Fach	Ab durch die Mitte	Nach Adam Riese

[1] Diese Aussage bezieht sich nur auf diejenige Literatur, die im Rahmen dieser Arbeit verwendet wurde.

[2] Der Begriff „Kongruenz“ steht (im linguistischen Kontext) nach Liedke, [67] für die Übereinstimmung formaler Merkmale von Wörtern in einem Satz.

Anmerkung: Geflügelte Worte sind ebenfalls Redewendungen, welche durch Zitate entstandenen sind, die von eindeutig nachweisbaren Verfassern stammen [113].

Zuordnung: $F_{12} \subset$ (Semantische Ebene).

5.6.4 F_{13}: Text-Komplexität

Diese Kategorie ist angelehnt an die sogenannten Lesbarkeitsformeln (engl. *Readability Measures*), die bereits in zahlreichen Forschungsarbeiten in der Disziplin der Autorschafts-Attribution eingesetzt wurden. Die hier eingesetzten Features unterscheiden sich jedoch gegenüber den existierenden Features, da diese speziell für die deutsche Sprache entwickelt wurden. Die Idee dabei ist statistisch auszudrücken, wie gut ein Text lesbar ist bzw. wie vielfältig dieser geschrieben wurde. Die folgende Tabelle listet einige dieser Features auf:

Lesbarkeitsmaß	**Formel**	**Quelle**
`Amdahls Verständlichkeitsindex`	$180 - \left(\frac{a^2+bc}{ab}\right)$	[28]
`Text-Redundanz-Index von Kuntzsch`	$0.449d - 2.467e - 0.937f - 14.417$	[28]
`Wiener Sachtextformel von Bamberger und Vanecek`	$\frac{1}{a}(0.1935g + 0.1672ah + 0.1297i) - 0.0327\frac{d}{b}$	[28]

Hierbei bedeuten die einzelnen Abkürzungen:

- $a =$ Anzahl der Wörter im Text
- $b =$ Anzahl der Sätze im Text
- $c =$ Anzahl der Silben im Text
- $d =$ Anzahl der einsilbigen Wörter im Text
- $e =$ Anzahl der Satzzeichen im Text
- $f =$ Anzahl der Fremdwörter im Text
- $g =$ Anzahl der Wörter im Text, die drei oder mehr Silben enthalten
- $h =$ Durchschnittliche Wortanzahl
- $i =$ Anzahl der Wörter im Text, die sechs oder mehr Zeichen enthalten

Zuordnung: $F_{13} \subset$ (Zeichen Ebene $\cup$ Lexikalische Ebene $\cup$ Syntaktische Ebene).

5.6.5 Verwendete Wortlisten

Einige der vorgestellten Feature-Kategorien basieren auf Wortlisten, die aus unterschiedlichen Quellen im Internet erschlossen wurden. Die folgende Tabelle listet dazu die betroffenen Kategorien, die dafür benötigten Preprocessing-Schritten sowie die Quellen, aus denen die Daten stammen auf:

F_i	**Preprocessing-Schritte**	**Quelle**
F_3	Konstruktion von Buchstaben Bigramme	[8]
F_4	Nicht notwendig gewesen.	[103]
F_6	Token $n-$Gramm Konstruktion für $n = 2, n = 3$.	[8]
F_8	POS-Tagging sowie Token $n-$Gramm Konstruktion für $n = 2, n = 3$.	[8]
F_9	POS-Tagging sowie Extraktion mittels regulärer Ausdrücke.	[8]
F_{11}	Nicht notwendig gewesen.	[36, 112]
F_{12}	Nicht notwendig gewesen.	[111]

Tabelle 12: Verwendete Wortlisten für einige Feature-Kategorien.

Anmerkung: Für F_3 wurden ursprünglich auch Trigramme verwendet. Diese wurden jedoch verworfen, nachdem festgestellt wurde, dass sie zu einer Verschlechterung der Ergebnisse geführt haben.

5.7 Feature-Normalisierung

In der Praxis liegt nahezu immer der Fall vor, dass die Textlänge eines Dokuments $\mathcal{D}_1$ länger/kürzer ist, als die eines anderen $\mathcal{D}_2$. Dies hat zur Folge, dass die absolute Häufigkeit eines Features f_i alleine nicht repräsentativ ist, da längere Texte gegenüber Kürzeren in der Regel größere Häufigkeiten für f_i aufweisen. Diese Problematik lässt sich jedoch recht einfach lösen, indem sämtliche absolute Feature-Häufigkeiten auf ein einheitliches Maß normalisiert werden, beispielsweise auf ein Intervall wie $[0; 1]$. Um eine solche sogenannte Feature-Normalisierung durchführen zu können, existieren verschiedene Strategien, von denen nachfolgend zwei vorgestellt werden.

- **Globale Normalisierung:** Hier werden zunächst sämtliche absolute Feature-Häufigkeiten $H = (\,\#(f_1, \mathcal{D}),\, \#(f_2, \mathcal{D}),\, \ldots,\, \#(f_n, \mathcal{D})\,)$ innerhalb des Dokuments $\mathcal{D}$ ermittelt. Anschließend wird jedes $\#(f_i, \mathcal{D})$ global normalisiert. Hierbei erfolgt die Normalisierung linear (lin), quadratisch ($quad$), logarithmisch (log) oder radiziert ($root$):

Normalisierung	**Berechnung**
$normalize_{\,lin}\,(\#(f_i, \mathcal{D}))$	$\dfrac{\#(f_i, \mathcal{D}) - min(H)}{max(H) - min(H)}$
$normalize_{\,quad}\,(\#(f_i, \mathcal{D}))$	$\left(\dfrac{\#(f_i, \mathcal{D}) - min(H)}{max(H) - min(H)}\right)^2$
$normalize_{\,log}\,(\#(f_i, \mathcal{D}))$	$\dfrac{\log(\#(f_i, \mathcal{D})) - \log(min(H))}{\log(max(H)) - \log(min(H))}$
$normalize_{\,root}\,(\#(f_i, \mathcal{D}))$	$\sqrt{\dfrac{\#(f_i, \mathcal{D}) - min(H)}{max(H) - min(H)}}$

Tabelle 13: Globale Feature Normalisierung (zusammengetragen aus [56, 86]).

- **Relative Häufigkeit:** Eine weitere verbreitete Möglichkeit um Features zu normalisieren, ist die relative Häufigkeit, die wie folgt definiert ist:

$$relFreq(\#(f_i, \mathcal{D}))) = \frac{\#(f_i, \mathcal{D})}{\#(d, \mathcal{D})}$$

  Hierbei drückt $\#(d, \mathcal{D})$ den Umfang der Grundgesamtheit aus, wobei d eine festgewählte Grundgesamtheit darstellt, die anhand einer Operation (z.B. Tokenisierung oder Satzsegmentierung) aus $\mathcal{D}$ gewonnen wird.

Im Rahmen dieser Arbeit wurde entschieden eine individuelle Form der relativen Häufigkeit als Normalisierungs-Strategie zu verwenden. Was genau bedeutet dabei individuell? Im Kontext der Autorschaftsanalyse fallen in der Regel viele Features aus unterschiedlichen Ebenen an, wie beispielsweise Zeichen, Tokens oder Phrasen. Dadurch ergeben sich unterschiedliche Grundgesamtheiten (z.B. Anzahl von Zeichen, Anzahl von Tokens oder Anzahl von Phrasen), sodass Features damit nicht einheitlich normalisiert werden können.

Im folgenden Unterabschnitt wird die individuelle relative Häufigkeit, symbolisiert durch $f_i(\mathcal{D})$, anhand eines Beispiels genauer erläutert. Im Anschluß daran wird erklärt, warum eine globale Normalisierung für Autorschaftsanalyse-Szenarien ungeeignet ist.

5.7.1 Individuelle relative Häufigkeit

Unter dem Begriff der individuellen relativen Häufigkeit wird in dieser Arbeit eine Normalisierungs-Strategie bezeichnet, die in Abhängigkeit von der absoluten Häufigkeit sowie einer Grundgesamtheit d gebildet wird. Formell entspricht die individuelle relative Häufigkeit der folgenden Gleichung:

$$f_i(\mathcal{D}) = \frac{\#(f_i, \mathcal{D})}{\#(d, \mathcal{D})}$$

Der Unterschied zu der relativen Häufigkeit äußert sich dadurch, dass d hier nicht fest ist, sondern stattdessen individuell gewählt wird. Die Individualität hängt dabei in erster Linie davon ab, aus welcher Kategorie F_j das Feature f_i stammt. In der folgenden Tabelle werden für sämtliche Feature Kategorien $F_1, F_2, \ldots, F_{13}$ die individuellen relativen Häufigkeiten $f_i(\mathcal{D})$ angegeben. Zu beachten ist dabei jedoch, dass ein F_j durchaus mehr als nur eine individuelle relative Häufigkeit enthalten kann, sodass nicht alle $f_i(\mathcal{D})$ in der Tabelle aufgeführt sind:

ID	Kategorie	$f_i(\mathcal{D})$
F_1	Interpunktion	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Tokens}, \mathcal{D})}$
F_2	Buchstaben	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Buchstaben}, \mathcal{D})}$
F_3	Buchstaben $n-$Gramme	$\frac{n \cdot \#(f_i, \mathcal{D})}{\#(\texttt{Buchstaben}, \mathcal{D})}$, für $n \in \{ 2, 3, \ldots, length(f_i) \}$
F_4	Funktionswörter	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Funktionswörter}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Types}, \mathcal{D})}$
F_5	Wort-Komplexität	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Tokens}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Types}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}$
F_6	Phrasen	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Nebensätze}, \mathcal{D})}$
F_7	POS-Tags	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Wörter}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Types}, \mathcal{D})}$
F_8	POS-Tag n-Gramme	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Nebensätze}, \mathcal{D})}$
F_9	Satz-Anfänge/Endungen	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}$
F_{10}	Grammatikalische Fehler	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Wörter}, \mathcal{D})} \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Types}, \mathcal{D})}$
F_{11}	Anglizismen	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Types}, \mathcal{D})}$
F_{12}	Redewendungen/Geflügelte Worte	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Nebensätze}, \mathcal{D})}$
F_{13}	Text-Komplexität	$\frac{\#(f_i, \mathcal{D})}{\#(\texttt{Sätze}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Wörter}, \mathcal{D})}, \frac{\#(f_i, \mathcal{D})}{\#(\texttt{Types}, \mathcal{D})}$

Tabelle 14: Individuelle relative Häufigkeiten in Abhängigkeit von Feature Kategorien.

Das folgende Beispiel verdeutlicht den Normalisierungs-Prozess anhand beider Normalisierungs-Strategien. Sei dazu zunächst das Textfragment aus [7] als $\mathcal{D}$ wie folgt gegeben:

$\mathcal{D} =$ „Endlos dehnen sich die Felder und Wälder in Ostpreußen und verlieren sich nach der Steppenewigkeit Rußlands. Wie eine blitzende Kette sind die vielen Seen in die bäuerliche Erde verstreut. Darüber wölbt sich der gewaltige Himmel. Im freien Raum zwischen den Wolken und den Straßen sausen im Winter die krachenden Stürme. Die Menschen, die hier wohnen, sind ernst und verschlossen." [7]

Im ersten Schritt wird $\mathcal{D}$ in die folgenden vier Feature Ebenen aufgeteilt:

E_1: Zeichen Ebene	E_2: Lexikalische Ebene
E_3: Syntaktische Ebene	E_4: Semantische Ebene

Die Aufteilung des Dokuments wird dabei mit Hilfe der NLP-Werkzeuge aus Kapitel 3.2 durchgeführt. Um beispielsweise die Lexikalische Ebene E_2 zu erhalten, muss ein Tokenizer wie folgt auf $\mathcal{D}$ angewendet werden:

$$\boxed{\text{Tokenizer}} \rightsquigarrow \mathcal{D} = (\texttt{ Endlos, dehnen, sich, die, Felder, und, Wälder, ...}) = E_2$$

Aus den resultierenden vier Feature Ebenen werden anschließend n (im konkreten Beispiel $n = 10$) Features entnommen. Seien dazu die folgenden Features gewählt:

$f_1 =$ `Buchstabe("e")`	$f_2 =$ `Buchstabe("n")`
$f_3 =$ `POS-Tag("Adjektiv")`	$f_4 =$ `POS-Tag("Konjunktiv")`
$f_5 =$ `Interpunktion(",")`	$f_6 =$ `Interpunktion(".")`
$f_7 =$ `Wort-Komplexität("Nomen der Länge 6")`	$f_8 =$ `Satz-Ende("Nomen")`
$f_9 =$ `Satz-Anfang("Artikel")`	$f_{10} =$ `POS-Phrase("Artikel-Nomen")`

Das Resultat der Feature Entnahme aus den Ebenen sowie die daraus resultierenden absoluten Häufigkeiten $\#(f_1, \mathcal{D})$, $\#(f_2, \mathcal{D})$, ..., $\#(f_{10}, \mathcal{D})$ lauten:

f_i	**Resultat der Entnahme**	$\#(f_i, \mathcal{D})$
f_1	(e, e, e, ...)	61
f_2	(n, n, n, ...)	36
f_3	(Endlos, blitzende, bäuerliche, gewaltige, ...)	8
f_4	(und, und, und, und)	4
f_5	(",", ",")	2
f_6	(".", ".", ".", ".", ".", ".")	6
f_7	(Felder, Wälder, Himmel, ...)	6
f_8	((Himmel, ...) (,Stürme, ...))	2
f_9	(Felder, Wälder, Himmel, ...)	6
f_{10}	(die Felder, der Steppenewigkeit, den Wolken, ...)	6

Ausgehend von den absoluten Häufigkeiten, wird nun die Feature Normalisierung durchgeführt. Dazu werden in der folgenden Tabelle die lineare Normalisierung sowie die individuelle relative Häufigkeit für jeden Feature f_i aufgelistet. Letztere benötigt dabei neben $\#(f_i, \mathcal{D})$ zusätzlich noch die Grundgesamtheit, die in der Tabelle ebenfalls mit angegeben ist:

f_i	F_j	$\#(f_i, \mathcal{D})$	**Grundgesamtheit bzgl.** F_j	$f_i(\mathcal{D})$	$normalize_{\,lin}\,(\#(f_i, \mathcal{D}))$
f_1	F_2	61	$\#(\texttt{Buchstaben}, \mathcal{D}) = 315$	$\frac{61}{315} = 0.19$	$\frac{61-1}{61-1} = 1$
f_2	F_2	36	$\#(\texttt{Buchstaben}, \mathcal{D}) = 315$	$\frac{36}{315} = 0.11$	$\frac{36-1}{61-1} = 0.58$
f_3	F_7	8	$\#(\texttt{POS-Tokens}, \mathcal{D}) = 70$	$\frac{8}{70} = 0.11$	$\frac{8-1}{61-1} = 0.12$
f_4	F_7	4	$\#(\texttt{POS-Tokens}, \mathcal{D}) = 70$	$\frac{4}{70} = 0.05$	$\frac{4-1}{61-1} = 0.05$
f_5	F_1	2	$\#(\texttt{Tokens}, \mathcal{D}) = 66$	$\frac{2}{66} = 0.03$	$\frac{2-1}{61-1} = 0.02$
f_6	F_1	5	$\#(\texttt{Tokens}, \mathcal{D}) = 66$	$\frac{5}{66} = 0.08$	$\frac{5-1}{61-1} = 0.07$
f_7	F_5	6	$\#(\texttt{Wörter mit 3 Vokale}, \mathcal{D}) = 13$	$\frac{6}{13} = 0.46$	$\frac{6-1}{61-1} = 0.08$
f_8	F_9	2	$\#(\texttt{Sätze}, \mathcal{D}) = 5$	$\frac{2}{5} = 0.40$	$\frac{2-1}{61-1} = 0.02$
f_9	F_9	1	$\#(\texttt{Sätze}, \mathcal{D}) = 5$	$\frac{1}{5} = 0.20$	$\frac{1-1}{61-1} = 0.00$
f_{10}	F_8	6	$\frac{\#(\texttt{Wörter}, \mathcal{D})}{\#(\texttt{POS-Tokens in der Phrase}, \mathcal{D})} = 29$	$\frac{6}{29} = 0.21$	$\frac{6-1}{61-1} = 0.08$

Tabelle 15: Tabellarischer Vergleich zwischen linearer Normalisierung und individuelle relative Häufigkeiten.

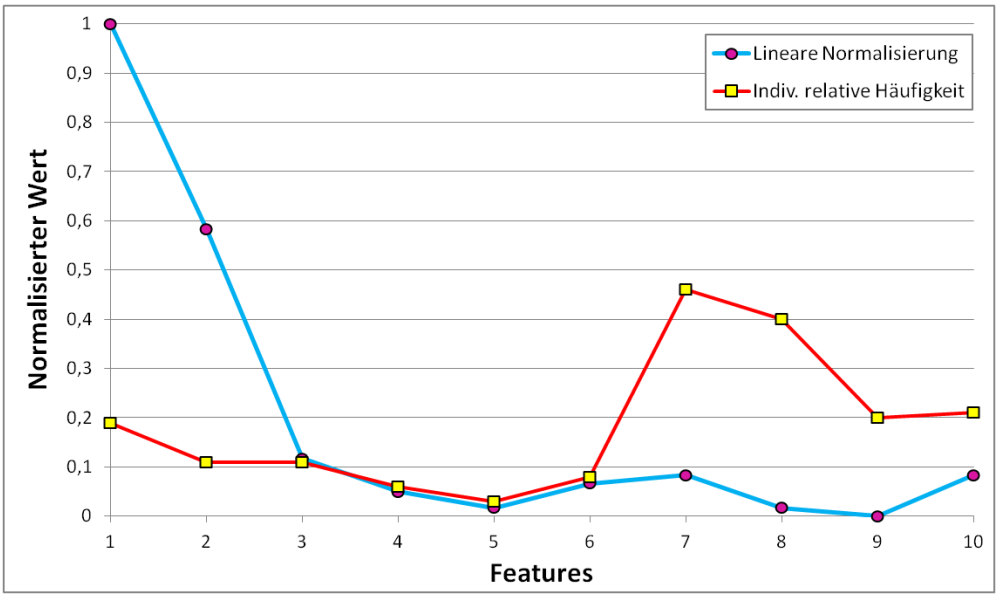


Abbildung 17: Bildlicher Vergleich zwischen linearer Normalisierung und individuelle relative Häufigkeiten.

Wie aus Abbildung 17 ersichtlich ist, liegen bei der linearen Normalisierung die meisten Werte (insgesamt sieben) im kleinsten Teilintervall $[0; 0.1]$. Bei der individuellen relativen Häufigkeit sind die Werte dagegen derart gestreut, dass höchstens drei innerhalb eines Teilintervalls liegen. Mit anderen Worten: Bei der individuellen relativen Häufigkeit existiert eine stärkere Varianz bzgl. der Datenpunkte (bzw. Werte). Je höher die Varianz ist, desto „einzigartiger“ wird dadurch der Feature-Vektor (bzw. der dadurch approximierte Autorenstil). Je einzigartiger ein Feature-Vektor dabei ist, desto besser kann dieser von anderen unterschieden werden.

5.8 Feature-Auswahl

Unter dem Begriff Feature-Auswahl wird innerhalb dieser Arbeit ein Prozess bezeichnet, der sich damit beschäftigt in einem Feature-Raum $\mathbb{F}$ die relevantesten Features ausfindig zu machen. Eine Feature-Auswahl ist sinnvoll, da es in Autorschaftsanalyse-Szenarien oftmals vorkommt, dass nur ein Bruchteil der Features aus $\mathbb{F}$ tatsächlich dazu beitragen eine verlässliche Diskriminierung der Autorenstile zu ermöglichen. Zahlreiche andere Features sind dagegen (je nach Szenario) nicht geeignet, weil sie entweder keinerlei Aussagekraft besitzen oder Redundanz aufweisen. Zudem können derartige irrelevante Features, Autorschaftsanalyse-Verfahren enorm verlangsamen. Dafür gibt es zwei Gründe:

1. **Feature Entnahme:** Es existieren leider zahlreiche Features, die nicht aus der reinen Textform eines Dokuments $\mathcal{D}$ entnommen werden können. Stattdessen müssen diese aus Feature-Ebenen entnommen oder auch teils in umfangreichen Wortlisten gesucht werden, was je nach Anzahl der Dokumente und deren Dokumentlängen, einen erheblichen Aufwand mit sich bringen kann. Da im Voraus leider nicht bekannt ist, ob sich diese Features für eine Analyse eignen, wird für die Entnahme irrelevanter Features unnötig (Lauf-)Zeit vergeudet.

2. **Vergleich der Autorenstile:** Um eine Autorschaftsanalyse durchzuführen, müssen Feature-Vektoren (bzw. die dadurch repräsentierten Autorenstile) gegeneinander, anhand von Metriken oder Klassifikatoren, verglichen werden. Irrelevante Features fließen in diesem Vergleich mit ein und verlangsamen unnötig dessen Berechnung.

Die Konsequenz ist, dass Techniken benötigt werden, um $\mathbb{F}$ auf eine kleinere Teilmenge von Features $\mathbb{F}'$ zu reduzieren, sodass dadurch eine Autorschaftsanalyse beschleunigt und optimiert werden kann. Voraussetzung für alle Techniken ist dabei stets eine vorliegende Feature-Matrix F_{matrix}, die im Kapitel 2.1 eingeführt wurde. Die Techniken für die Auswahl von Features können dabei entweder manuell oder automatisiert erfolgen. In den folgenden Unterabschnitten werden für beide Varianten Ansätze vorgestellt, die allesamt in den Experimenten (Kapitel 10) verwendet wurden.

5.8.1 Automatisierte Verfahren für die Feature Auswahl

Automatisierte Verfahren gliedern sich in die beiden Kategorien Feature-Selektion sowie Feature-Extraktion[1] auf:

Feature-Selektion: Unter dem Begriff Feature-Selektion[2] wird ein Prozess bezeichnet, welcher ausgehend von F_{matrix} und dem zugrundeliegenden Feature-Raum $\mathbb{F}$ eine Teilmenge $\mathbb{F}'$ selektiert, deren Features ungefähr genauso aussagekräftig sind, wie alle Features in $\mathbb{F}$ zusammen. In manchen Fällen (so wie auch in dieser Arbeit) kann es sogar vorkommen, dass die Features in $\mathbb{F}'$ sogar zu einem deutlich besseren Ergebniss führen, als der ganze Feature Raum $\mathbb{F}$. Diese Aussage wird durch ein Experiment in Kapitel 10 belegt.

Schematisch kann die Feature-Selektion beispielsweise wie folgt ausgedrückt werden:

$$\boxed{\text{Feature-Selektion}} \rightsquigarrow \Big(F_{matrix} \text{ mit } \mathbb{F} = \{ f_1, f_2, \ldots, f_n \} \Big) = \Big(F_{matrix} \text{ mit } \mathbb{F}' = \{ f_1, f_2, f_3 \} \Big)$$

Feature-Selektions Algorithmen lassen sich nach Dash et al. [20] in insgesamt drei Arten aufteilen:

[1] Dieser Begriff sollte nicht mit der Feature-Entnahme verwechselt werden. In der Feature-Entnahme werden zwar Features extrahiert, jedoch ausschließlich aus dem Text. Die Feature-Extraktion extrahiert diese dagegen aus dem Feature Raum.

[2] In der Literatur auch bekannt als *Feature Subset Selection*.

1. **Vollständig:** Hierbei wird F_{matrix} vollständig nach den besten Features durchsucht. Die optimale Teilmenge $\mathbb{F}'$ wird dabei durch eine Evaluierungsfunktion bestimmt, die die Beste aller möglichen Feature Kombinationen auswählt, [20].

2. **Heuristisch:** Hier wird F_{matrix} wiederum iterativ nach den besten Features durchsucht. Dazu werden nach Han, [41, Seiten: 75–77] unter anderem die folgenden zwei Strategien verwendet:

   (a) **Schrittweise Vorwärtsselektion:** Hierbei wird die Teilmenge $\mathbb{F}'$ zunächst mit $\mathbb{F}' = \{\,\}$ initialisiert. Anschließend werden die besten Features die in F_{matrix} vorkommen, anhand eines Maßes (z.B. Information Gain) ermittelt und nach Relevanz sortiert. Diejenigen Features, die die höchsten Werte besitzen, werden anschließend zu der Teilmenge $\mathbb{F}'$ nach und nach hinzugefügt.

   (b) **Schrittweise Rückwärtsselektion:** Bei dieser Methode wird umgekehrt vorgegangen. Die Menge $\mathbb{F}'$ wird mit $\mathbb{F}' = \{\, f_1, f_2, \ldots, f_n \,\}$ initialisiert. Anschließend wird in jedem Iterationsschritt der schlechteste Feature, der in F_{matrix} vorkommt eliminiert, sodass $\mathbb{F}'$ zum Schluß nur noch aus Features besteht, die die höchsten (Information Gain-)Werte aufweisen, [41, Seiten: 75–77].

   Für beide Strategien wird in jedem Iterationsschritt eine Kreuzvalidierung durchgeführt, um nachvollziehen zu können, ob die Hinzunahme/Entfernung von Features zu einer signifikanten Verbesserung geführt haben. Ist dies nicht der Fall, so terminiert der heuristische Selektions-Algorithmus und liefert damit die optimale Teilmenge $\mathbb{F}'$, [20].

3. **Zufällig:** Hierbei werden aus F_{matrix} zufällig k Feature Teilmengen gebildet, auf die dann verschiedene ML-Algorithmen (z.B. genetische oder heuristische Verfahren) angewendet werden, um diese zu optimieren, [20]. Die Idee von zufälligen Feature-Selektions-Algorithmen ist es lokale Maxima zu umgehen, die bei rein heuristischen Verfahren vorkommen können. Die Hoffnung ist dabei, irgendwann zu einem globalen Maximum und damit zu einer optimalen Teilmenge $\mathbb{F}'$ zu gelangen. Da diese Verfahren hinsichtlich ihres Aufbaus sehr komplex und bzgl. der Laufzeit extrem aufwändig sein können und in der Praxis bisher nur selten anzutreffen sind, wird an dieser Stelle von einer detailierten Beschreibung abgesehen.

Feature-Extraktion: Unter diesem Begriff wird die Erzeugung einer neuen (niedrig dimensionierten) Feature-Matrix F'_{matrix} verstanden, die mit Hilfe einer Transformation $\psi(\,\cdot\,)$ bzgl. der Features in F'_{matrix} resultiert. Schematisch kann die Feature-Extraktion beispielsweise wie folgt ausgedrückt werden:

$$\boxed{\text{Feature-Extraktion}} \rightsquigarrow \Big(F_{matrix} \text{ mit } \mathbb{F} = \{\, f_1, f_2, \ldots, f_n \,\} \Big) \;=\; \Big(F'_{matrix} \text{ mit } \mathbb{F}' = \{\, \lambda_1, \lambda_2, \lambda_3 \,\} \Big)$$

und der komprimierten Form:

$$\{\, \lambda_1, \lambda_2, \lambda_3 \,\} = \psi\Big(\{\, f_1, f_2, \ldots, f_n \,\} \Big)$$

In der Praxis existieren zahlreiche Methoden für die Feature-Extraktion, wie beispielsweise die Hauptachsentransformation (engl: *Principal Components Analysis*, kurz PCA), Lineare Diskriminanzanalyse (engl: *Linear Discriminant Analysis*, kurz LDA), Multidimensionale Skalierung (engl. *Multidimensional Scaling*, kurz MDS) oder auch die Wavelet Transformation. Um den Rahmen dieses Abschnitts nicht zu sprengen, wird im Folgenden nur die PCA Methode nach Han, [41, Seiten: 79–80] erläutert.

Ausgehend von F_{matrix} mit m Zeilen (Feature-Vektoren) und n Spalten (Feature-Instanz Vektoren), sucht die PCA nach den k othogonalen Feature-Instanz Vektoren, die in der Lage sind die Autorenstile am besten unterscheiden zu können, dabei gilt $k \leq n$. Anders als bei der Feature-Selektion, die eine Teilmenge von Features aus $\mathbb{F}$ liefert, „kombiniert“ die PCA die wesentlichen Features indem mit Hilfe einer Transformation F_{matrix} in einen anderen Vektorraum mit neuer Basis überführt wird. Durch diese Kombination kann die PCA Zusammenhänge von Features enthüllen, die in F_{matrix} möglichweise nur implizit gegeben waren. Nachfolgend werden die einzelnen Schritte der PCA beschrieben:

1. Der erste Schritt besteht darin die Features zu normalisieren[1], sodass alle Werte in einem einheitlichen Intervall liegen, [41, Seiten: 79–80].

2. Anschließend werden die k othogonalen Feature-Instanz Vektoren berechnet, die die Basis von F_{matrix} bilden. Hierbei handelt es sich um Einheitsvektoren, die senkrecht zueinander stehen und im ML-Fachjargon als *Hauptkomponenten* bezeichnet werden. Eine Linearkombination dieser Hauptkomponenten entspricht F_{matrix}, [41, Seiten: 79–80].

3. Im nächsten Schritt werden die Hauptkomponenten $\lambda_1, \lambda_2, \ldots, \lambda_k$ (und damit die Koordinatenachsen) nach ihrer aufsteigender Signifikanz sortiert: $\lambda_q > \lambda_{q+1} > \lambda_{q+2} > \ldots$, sodass λ_q die höchste Varianz in F_{matrix} enthält, λ_{q+1} die Zweithöchste, etc., [41, Seiten: 79–80].

4. Im letzten Schritt werden diejenigen Hauptkomponenten, die eine niedrige Varianz aufweisen, eliminiert. Dies kann z.B. dadurch realisiert werden, indem diese bei einer Unterschreitung eines bestimmten Varianz-Schwellwerts entfernt bzw. ignoriert werden. „Weniger relevant“ bedeutet dabei, dass diese Hauptkomponenten nur noch Informationen über das Rauschen enthalten (also nicht brauchbare Features) und sich somit für ein Autorschaftsanalyse-Szenario nicht eignen. Hauptkomponenten mit einer höheren Varianz enthalten dagegen jene Features, die sich eher eignen Autorenstile zu diskriminieren, [41, Seiten: 79–80].

Beobachtung: Sowohl die Feature-Selektion als auch die Feature-Extraktion haben ihre Vor- und Nachteile. Bei der Feature-Extraktion (z.B. PCA) ist der Berechnungsaufwand mit einer kubischen Laufzeit relativ gering, während bei der Feature-Selektion (z.B. die vollständige Suche) eine erschöpfende Suche nach der besten Teilmenge von Features erfolgen muss. Eine derartige Suche ist mit einer exponentiellen Laufzeit verbunden und eignet sich daher nicht für höherdimensionierte Feature-Matrizen. Allerdings haben Feature-Extraktions-Verfahren wie die PCA auch einige Nachteile. So ist diese z.B. nicht parametrisierbar und kann nur in einer Black-Box Manier verwendet werden [93], während Feature-Selektions-Algorithmen über viele Einstellungsmöglichkeiten verfügen, um die Suche nach relevanten Festures zu optimieren.

Ein weiterer Nachteil von Feature-Extraktions-Verfahren wie die der PCA ist die Verborgenheit von einzelnen relevanten Features. Wie oben erwähnt, werden zwar relevante Features ermittelt, jedoch in einer kombinierten Form, aus der zunächst nicht hervorgeht, welche Features genau die beste Diskriminierungseigenschaft besitzen. Bei der Feature-Selektion wird dagegen keine Transformation mit anschließender Gruppierung durchgeführt, sondern stattdessen relevante Features mit Hilfe von Gütefunktionen (z.B. Information Gain) direkt aus F_{matrix} ermittelt. Dies birgt den Vorteil, dass die Selektion transparent bleibt, sodass dadurch ersichtlich wird, welche einzelnen Features zu einer Verbesserung (oder Verschlechterung) bzgl. des Diskriminierungsergebnisses führen.

5.8.2 Manuelle Verfahren für die Feature Auswahl

Eine Auswahl von relevanten Features muss nicht immer automatisch durch Algorithmen erfolgen. Vor allem für kleinere Autorschaftsanalyse Szenarien (z.B. 8-15 Autoren und ca. 400 Features) kann eine manuelle Feature-Selektion schnell, transparent und erfolgsversprechend durchgeführt werden. In diesem Unterabschnitt wird eine effiziente Möglichkeit gezeigt, wie eine solche Feature-Selektion realisiert werden kann.

Visual Feature-Selection: Hierunter wird in dieser Arbeit eine visuell gestützte Methode verstanden, um Features mit kognitiven Fähigkeiten ausfindig zu machen. Hierfür werden die folgenden Voraussetzungen benötigt:

1. Eine Feature-Matrix F_{matrix} aus der die relevanten Features ermittelt werden sollen.

[1] Innerhalb dieser Arbeit fällt dieser Schritt weg, da dieser bereits in der Feature-Normalisierung (Kapitel 5.7) abgedeckt wird.

2. Ein Tabellenkalkulationsprogramm (z.B. „Microsoft Excel“ oder „OpenOffice Calc“), welches F_{matrix} darstellen kann und in der Lage ist, die darin befindlichen Feature-Instanz Vektoren $\mathcal{I}(f_1), \mathcal{I}(f_2), \ldots, \mathcal{I}(f_n)$ zu visualisieren.

3. Ein Benutzer der mit dem Tabellenkalkulationsprogramm umgehen kann.

Die Idee dieser Methode besteht darin sämtliche Spalten derart zu visualisieren, dass Features die eine starke Varianz aufweisen, sichtbar und vom Benutzer schnell erfasst werden können. Hierfür müssen zunächst Teilintervalle definiert werden, die mit festgelegten Farben oder Symbolen (oder beiden) assoziiert werden.

Beispiel: Sei $Q = [0; 1]$ das einheitliche Intervall in dem sämtliche Feature Werte $f_k(\mathcal{D}_{i\mathcal{A}_j})$ liegen. Q wird zunächst in q disjunkte Teilintervalle $T_1, T_2, \ldots, T_q$ aufgeteilt. Anschließend werden q Symbole $Sym_1, Sym_2, \ldots, Sym_q$ ausgewählt, sodass jedes T_i auf ein Sym_i visuell abgebildet wird. Die folgende Tabelle verdeutlicht dies anhand von $q = 5$ Teilintervalle:

$T_1 = [0; 0.19\ldots]$	$T_2 = [0.20; 0.39\ldots]$	$T_3 = [0.40; 0.59\ldots]$	$T_4 = [0.60; 0.79\ldots]$	$T_5 = [0.80; 1]$
○	◔	◑	◕	●

Der nächste Schritt besteht darin dem Tabellenkalkulationsprogramm die Aufteilung des Intervalls als auch die visuelle Abbildung(en) mitzuteilen. Dabei muss dieser Schritt nicht zwingend manuell erfolgen, da die meisten Programme bereits über viele Visualisierungsmöglichkeiten verfügen und vorgefertigte Intervall-Aufteilungen enthalten, die der Benutzer beliebig modifizieren kann.

Als nächstes müssen die Zellen (also die einzelnen $f_k(\mathcal{D}_{i\mathcal{A}_j})$) farblich hervorgehoben werden. Dies erfolgt automatisch durch das Tabellenkalkulationsprogramm. Nachfolgend wird ein Ausschnitt aus einer resultierenden Visualisierung gezeigt:

Hyphens	Functionword(werden)	SentenceEnd(VVINF)	SentenceBegin(ART)	Letter:(g)
0	0,0217	0,5	0,1667	0,0279
0,0143	0,0152	0,5	0,1667	0,0346
0	0,0395	0	0	0,0279
0	0,025	0,25	0	0,0225
0	0	0	0,1111	0,0242
0,0072	0,0156	0,0789	0,1842	0,0286
0,0072	0,0143	0,1573	0,2022	0,0297
0,0019	0,008	0,2619	0,1429	0,0302
0,001	0,0041	0,35	0,05	0,0253
0	0,0435	0,1429	0,1429	0,0294
0,002	0,0345	0,2963	0	0,0318
0	0	0,0952	0,0476	0,0203
0,0131	0,0096	0,2535	0,1972	0,0238
0,0092	0	0,3333	0	0,0259
0,0114	0,0279	0,53	0,1875	0,0219
0,0116	0,0253	0,1333	0,2333	0,0205
0,01	0,0099	0,25	0,2188	0,0252
0,0174	0,02	0,2143	0,4	0,0236
0,0082	0,0088	0,692	0,3333	0,0215
0	0,025	0,1429	0,4286	0,0309
0	0,019	0,1667	0,4167	0,0194
0,0022	0,0192	0,1923	0,1923	0,0252
0	0	0,2	0,2	0,0269
0	0	0,3333	0,3333	0,0277

Abbildung 18: Visual Feature-Selection: Entdeckung von zwei relevanten Features.

Ausgehend von dieser Visualisierung, müssen nun relevante Features durch den Benutzer ausfindig gemacht werden. Dafür muss dieser die Spalten anhand der farblichen/symbolischen Abbildungen analysieren. Im konkreten Beispiel fallen dabei schnell die folgenden beiden Features auf:

1. f_i = `Satzende(VVINF)` mit $f_i \in F_9$.
2. f_{i+1} = `Satzanfang(ART)` mit $f_{i+1} \in F_9$.

Die beiden Features f_i und f_{i+1} unterscheiden sich deutlich, anhand der Zellenfarben sowie den Tortendiagramm-Symbolen, von den anderen Features. Die Zellenfarben sind dabei kontinuierlich auf jedes $f_k(\mathcal{D}_{i_{\mathcal{A}_j}})$ abgestimmt (rot = **niedrigster Wert**, grün = **höchster Wert**), während die Symbole diskret, wie oben aufgführt, die fünf Teilintervalle hinsichtlich der Abbildung repräsentieren.

5.9 Feature-Gewichtung

Neben der Feature Auswahl gibt es ein weiteres Konzept, um die Genauigkeit eines Autorschaftsanalyse-Systems zu steigern. Hierfür werden die Werte innerhalb einer Feature-Matrix F_{matrix}, oder genauer, innerhalb der Spalten $\mathcal{I}(f_1), \mathcal{I}(f_2), \ldots, \mathcal{I}(f_n)$ modifiziert, um somit eine Varianz zu verstärken, die ursprünglich nur in einer schwächeren Form vorlag. Hierfür sind verschiedene Feature-Gewichtungsstrategien denkbar, die eine solche „künstliche“ Varianz erzeugen können. In diesem Abschnitt wird eine eigene Gewichtungsstrategie vorgeschlagen, die im Folgenden erläutert wird.

5.9.1 Lowest Scores Elimination

Unter dem Begriff Lowest Scores Elimination wird in dieser Arbeit eine Feature Gewichtungsstrategie verstanden, die ausgehend von F_{matrix} zunächst Spalten sucht, deren Einträge großteils von Null verschieden sind, z.B.

$$\mathcal{I}(f_1) = \Big(0.00,\ 0.24,\ 0.02,\ 0.19,\ 0.41,\ \ldots,\ 0.79 \Big)$$

Im nächsten Schritt werden die Werte innerhalb einer solchen Spalte modifiziert. Dieser Schritt kann dabei durch zwei Ansätze realisiert werden:

1. **Maximum Value Survives:** Hierbei wird der maximale Wert einer Spalte $\mathcal{I}(f_i)$ behalten, während alle anderen Werte auf Null gesetzt werden. Diese Herangehensweise ist dann sinnvoll, wenn $\mathcal{I}(f_i)$ ohnehin schon einen (maximalen) Wert enthält, der sich signifikant von allen anderen unterscheidet, z.B.:

   $$\mathcal{I}(f_2) = \Big(0.00,\ 0.03,\ 0.17,\ 0.04,\ 0.12,\ 0.82,\ 0.09 \Big)$$

   Hierbei fällt auf, dass der Wert 0.82 sich extrem von den anderen distanziert, sodass diese eliminiert bzw. auf Null gesetzt werden:

   $$\mathcal{I}'(f_2) = \Big(0.00,\ 0.00,\ 0.00,\ 0.00,\ 0.00,\ 0.82,\ 0.00 \Big)$$

2. **Upper Median Values Survive:** Bei diesem Ansatz werden zunächst die Werte einer Spalte $\mathcal{I}(f_i)$ sortiert, um daraus den Median zu ermitteln. Anschließend werden alle Werte die darunter liegen eliminiert, während die restlichen Werte verbleiben. Diese Herangehensweise ist angebracht, falls die Werte in $\mathcal{I}(f_i)$ linear ansteigen bzw. gestreut sind. Das folgende Beispiel verdeutlicht die Idee dieses Ansatzes. Sei dazu eine geeignete Spalte wie folgt gegeben:

   $$\mathcal{I}(f_3) = \Big(0.19,\ 0.37,\ 0.05,\ 0.49,\ 0.57,\ 0.28,\ 0.61 \Big)$$

   Zunächst gilt es die Folge zu sortieren, um den Median bestimmen zu können. Es gilt hierbei $\text{Median}(\mathcal{I}(f_3)) = 0.37$, sodass alle Werte unterhalb dieses Werts auf Null gesetzt werden:

   $$\mathcal{I}'(f_3) = \Big(0.0,\ 0.37,\ 0.0,\ 0.49,\ 0.57,\ 0.0,\ 0.61 \Big)$$

   Das Resultat der Gewichtung ergibt für $\mathcal{I}'(f_3)$ eine höhere Varianz gegenüber $\mathcal{I}(f_i)$:

   $$\text{Variance}(\mathcal{I}(f_3)) = \frac{1}{n} \sum_{k=1}^{n} \Big(\mathcal{I}(f_3)_k - \text{Mean}(\mathcal{I}(f_3)) \Big)^2 \approx 0.036$$

$$\text{Variance}(\mathcal{I}'(f_3)) = \frac{1}{n}\sum_{k=1}^{n}\Big(\mathcal{I}'(f_3)_k - \text{Mean}(\mathcal{I}'(f_3))\Big)^2 \approx 0.067$$

Um die Varianz zusätzlich zu verstärken, können des Weiteren sämtliche Nicht-Null-Werte in $\mathcal{I}'(f_i)$ mittels Addition/Subtraktion mit einer Konstante modifiziert werden.

6 Einführung in die Autorschaftsanalyse

Dieses Kapitel führt in die Autorschaftsanalyse und die darunter fallenden Unterdisziplinen ein. Im Vordergrund stehen dabei die drei Kerndisziplinen: Autorschafts-Attribution, Autorschafts-Verifikation sowie die intrinsische Exploration. Hierfür werden zunächst die einzelnen Disziplinen beschrieben und mit Hilfe der eingeführten Notation aus Kapitel 2.1 (soweit wie möglich) formalisiert, um dadurch die Funktions- bzw. Vorgehensweise verdeutlichen zu können. Anschließend werden einige verwandte Disziplinen erläutert sowie deren Querbezüge zu den drei Kerndisziplinen hergestellt. Zum Ende des Kapitels werden schließlich die wichtigsten Herausforderungen diskutiert sowie einige Anwendungsgebiete der Autorschaftsanalyse aufgelistet.

6.1 Autorschaftsanalyse

Der Begriff der Autorschaftsanalyse wird in der Literatur des Öfteren unterschiedlich aufgefasst. Während Stein et al. [102, Seite: 4] beispielsweise unter diesem Begriff lediglich die beiden Disziplinen Autorschafts-Attribution sowie Autorschafts-Verifikation einordnen, setzt Stamatatos [99, Seite: 2] die Liste fort und fügt die Disziplinen des Sprachprofilings, Plagiarismus-Detektion, Autor Charakterisierung sowie die Erkennung von Stil-Inkonsistenzen hinzu. Innerhalb dieser Arbeit wird der Begriff der Autorschaftsanalyse, in Anlehnung an die Beschreibung von Stamatatos, wie folgt definiert:

Definition 6.1. ***Autorschaftsanalyse:*** *Die Autorschaftsanalyse stellt ein Oberbegriff für sämtliche Disziplinen dar, die je nach Kontext (Attribution, Verifikation, Aufspüren von Stil-Inkonsistenzen, etc.) Autorenstile untersuchen.*

6.2 Einführung in die Autorschafts-Attribution

Die Autorschafts-Attribution ist eine intensiv erforschte Disziplin, die viele Forscher aus den unterschiedlichsten Gebieten seit mehr als einem Jahrhundert beschäftigt. Bis heute lässt es sich schwer einschätzen, wieviele Forschungsarbeiten zu dieser Disziplin existieren. Dies liegt daran, dass viele Forschungsarbeiten weder frei zugänglich, noch elektronisch erfasst oder in einer einheitlichen Sprache (z.B. Englisch) verfasst worden sind. Zudem hat diese Disziplin aufgrund ihrer Popularität damit zu kämpfen, dass im Laufe der Jahre zahlreiche Synonyme entstanden sind, die das Auffinden entsprechender Arbeiten deutlich erschweren. Zu den bekannten Synonymen zählen unter anderem die „Autorenbestimmung“, „Autorzuordnung“, „Autorenschaftsattribution“, „Autor-Identifikation“ und Ähnliche. Um dennoch ein grobes Bild davon zu erhalten, wieviele Veröffentlichungen für diese Disziplin (zumindest) elektronisch verfügbar sind, wurden im Rahmen einer Internet-Recherche, Suchanfragen mit den Stichworten α = „Authorship Attribution“ sowie β = „Authorship Identification“ an bekannte Wissenschaftsportale gerichtet. Dabei wurden die Anfragen in Anführungszeichen gesetzt, um ausschließlich Treffer zu erhalten, die im Titel α und β beinhalteten. Das Resultat der Suchanfragen führte zu den folgenden Ergebnissen[1]:

[1] Bei den Treffern bzgl. der Suchanfragen fielen insbesondere vier Personen auf, die sowohl viele wissenschaftliche Beiträge veröffentlicht haben als auch in zahlreiche Arbeiten referenziert worden sind. Im Anhang (Seite: 163) findet sich eine kleine Auflistung zu diesen Personen.

Portal	**Anzahl** (α)	**Anzahl** (β)	**Quelle**
CiteSeer^x	306	76	[13]
ACLWeb	156	19	[2]
ACM	66	216	[3]
Elsevier	5169	1481	[26]
IEEE Xplore	281	59	[64]
SpringerLink	1418	38	[97]

Tabelle 16: Veröffentlichungen in der Disziplin der Autorschafts-Attribution.

Beobachtungen: Es fällt auf, dass das Wissenschaftsportal Elsevier die höchste Anzahl an Veröffentlichungen vorzuweisen hat. Eine genauere Untersuchung der Ergebnisse zeigte, das ein Großteil der wissenschaftlichen Beiträge dort aus dem Bereich der Humanwissenschaften stammt (insbesondere Linguistik, Philosophie und Psychologie). Ein Großteil der Veröffentlichungen in den anderen Portalen stammt dagegen aus dem Bereich der Naturwissenschaften (hier vor allem Informatik und Mathematik), sodass die Autorschafts-Attribution damit als eine interdisziplinäre Disziplin betrachtet werden kann. In einer weiteren Beobachtung fiel außerdem auf, dass einige wissenschaftliche Beiträge sich mit sehr abwechslungsreichen Themen auseinandersetzen. Neben der Attribution von klassischer Literatur [115] oder von Kapitelabschnitten aus dem alten Testament [57], finden sich unter anderem auch Forschungsarbeiten für die Attribution von Twitter-Nachrichten [61] sowie Forenbeiträge von extremistischen Gruppen [1].

6.2.1 Ziel der Autorschafts-Attribution

Das primäre Ziel der Autorschafts-Attribution ist es zu einem gegebenen anonymen Dokument den stilistisch ähnlichsten Autor zuzuordnen. Die Hoffnung dabei ist, dass dieser auch tatsächlich den wahren Autoren des anonymen Dokuments wiederspiegelt. Im folgenden werden die einzelnen Komponenten der Autorschafts-Attribution aufgelistet:

- **Anonymes Dokument:** Zunächst wird ein gegebenes anonymens Dokument $\mathcal{D}_\varepsilon$ benötigt, dessen Autorschaft attribuiert werden soll. Der unbekannte Autor wird dabei als ε bezeichnet.

- **Trainingsmenge:** Neben dem anonymen Dokument wird eine Trainingsmenge $\mathbb{D}_{train}$ benötigt, um darin den zu ε stilistisch ähnlichsten Autor $\mathcal{A}_{max}$ ausfindig zu machen. Die Grundannahme[1] hierbei ist, dass der wahre Autor des anonymen Dokuments (ausgedrückt durch $\mathcal{A}_{true}$) innerhalb der Trainingsmenge bekannt sein muss. Diese Grundannahme ist dabei genau dann erfüllt, falls $\mathcal{D}_{\mathcal{A}_{true}} \in \mathbb{D}_{train}$ bzw. $\mathbb{D}_{\mathcal{A}_{true}} \subseteq \mathbb{D}_{train}$ gilt.

- **Features:** Eine weitere wichtige Komponente stellen die Features $\mathbb{F} = \{ f_1, f_2, \ldots, f_n \}$ dar, mit deren Hilfe sich die Autorenstile repräsentieren lassen. Jedes Dokument $\mathcal{D}_{i\mathcal{A}_j}$ wird dafür in eine Feature-Vektor[2] Darstellung $\mathcal{F}_{i\mathcal{A}_j}$ überführt, welche wiederum $k \leq n$ Stilmerkmale $\{ f_1, f_2, \ldots, f_k \}$ enthält, die auf $\mathcal{D}_{i\mathcal{A}_j}$ angewendet wurden. Es gilt somit:

  $$\mathcal{F}_{i\mathcal{A}_j} = \Big(f_1(\mathcal{D}_{i\mathcal{A}_j}),\, f_2(\mathcal{D}_{i\mathcal{A}_j}),\, \ldots,\, f_k(\mathcal{D}_{i\mathcal{A}_j}) \Big)$$

  Die Stilmerkmale innerhalb der Feature-Vektoren sollten dabei so gewählt werden, dass ihre Diskriminierungseigenschaft möglichst hoch ist, um dadurch die Autorenstile zuverlässig unterscheiden zu können. Hierfür können die beschriebenen Strategien in Kapitel 5.8 verwendet werden.

[1] In der Literatur existieren dazu einige Arbeiten (z.B. [58]), die diese Grundannahme aufheben. Innerhalb dieser Arbeit wird diese jedoch gefordert.

[2] Siehe dazu Kapitel 5.1.

- **Attributions-Modell:** Die wichtigste Komponente stellt das Attributions-Modell dar, welches aus einem Klassifikator, einer Metrik oder einer Kombination aus beidem besteht. Mit Hilfe des Modells wird der Stil von ε mit denen aller Autoren $\mathcal{A}_j$ in $\mathbb{D}_{train}$ verglichen, um dadurch den zu ε stilistisch ähnlichsten Autor $\mathcal{A}_{max}$ bestimmen zu können. Eine erfolgreiche Attribution liegt dabei genau dann vor, wenn die folgende Konjunktion gilt:

  $$(\varepsilon = \mathcal{A}_{max}) \wedge (\mathcal{A}_{max} = \mathcal{A}_{true})$$

  Informell: Der stilistisch ähnlichste Autor ist auch gleichzeitig der wahre Autor des anonymen Dokuments.

- **Schwellwert:** Mit Hilfe eines Schwellwerts θ kann eine Trennschärfe zwischen $\mathcal{A}_{max}$ und allen anderen $\mathcal{A}_j$ innerhalb $\mathbb{D}_{train}$ definiert werden. In der Praxis zeigt sich jedoch, dass ein derartiges θ nur schwer festzulegen ist, da dieser von vielen Faktoren abhängt. Zu den Faktoren zählen z.B. verwendete Metrik/Klassifikator, Anzahl von Trainingsdokumenten, Features, Dokumentenlängen, etc. Eine weitere Problematik besteht darin, dass viele unterschiedliche Autorenstile sich nur durch minimale Abweichungen voneinander unterscheiden, sodass durch ein zu hoch (bzw. zu niedrig) gesetztes θ eine möglicherweise korrekte Attribution fehlschlagen könnte. Aus diesem Grund sollte θ nur als ein optionaler Parameter betrachtet werden.

Die folgende Abbildung fasst sämtliche Komponenten zu einem abstrakten Modell der Autorschafts-Attribution zusammen:

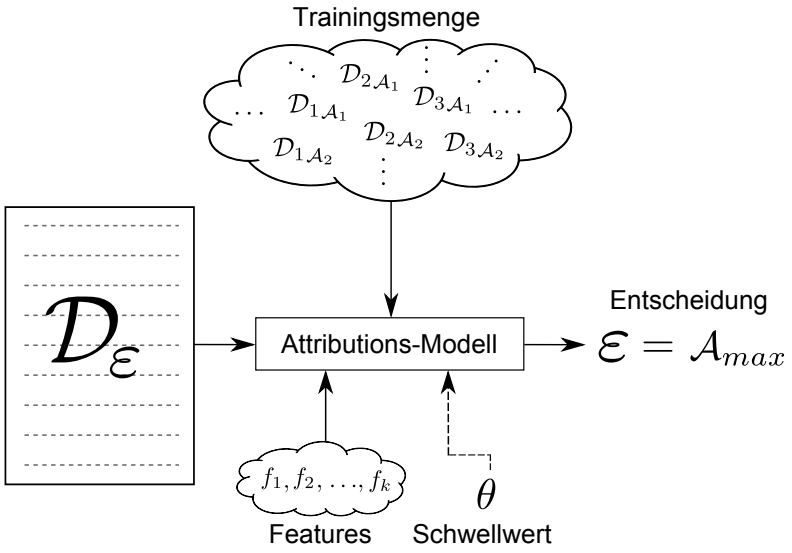


Abbildung 19: Abstraktes Modell der Autorschafts-Attribution.

6.3 Einführung in die Autorschafts-Verifikation

Im Gegensatz zu der Autorschafts-Attribution ist die Autorschafts-Verifikation eine deutlich weniger erforschte Disziplin. In den gleichen Portalen, die in Tabelle 16 aufgelistet sind, konnten für das Stichwort „Authorship Verification“ nur die folgende Anzahl an wissenschaftlichen Beiträgen ermittelt werden:

Portal	Anzahl	Quelle
CiteSeer^x	20	[13]
ACLWeb	5	[2]
ACM	4	[3]
Elsevier	8	[26]
IEEE Xplore	23	[64]
SpringerLink	11	[97]

Tabelle 17: Veröffentlichungen in der Disziplin der Autorschafts-Verifikation.

Die Ergebnisse lassen vermuten, dass die Autorschafts-Verifikation nicht nur eine seltene Disziplin darstellt, sondern gleichzeitig ein schwieriges Problem aufwirft, dass im folgenden Unterabschnitt genauer erläutert wird.

6.3.1 Ziel der Autorschafts-Verifikation

Die Disziplin der Autorschafts-Verifikation kann als ein Spezialfall der Autorschafts-Attribution verstanden werden, bei der die Attribution der Autorschaft lediglich auf einen Autor beschränkt ist. Das primäre Ziel hierbei ist, ausgehend von einem gegebenen (nicht-anonymen) Dokument $\mathcal{D}_{\mathcal{A}_j}$ sowie einer Menge mit Beispieltexten von $\mathcal{A}_j$ urteilen zu können, ob es sich bei dem ausgewiesenen Autor $\mathcal{A}_j$ tatsächlich um diesen handelt oder nicht. Im Kontext des ML's (maschinellen Lernens) handelt es sich bei der Autorschafts-Verifikation um eine sogenannte Ein-Klassen-Klassifikation[1], da hier die Entscheidung für eine Klasse ($\mathcal{A}_j$) oder gegen sie ($\neg\mathcal{A}_j$) getroffen werden muss.

Nachgolgend werden zunächst die einzelnen Komponenten der Autorschafts-Verifikation aufgeführt, wobei unterschiedliche Eigenschaften im Bezug zur Autorschafts-Attribution hervorgehoben werden. Anschließend wird das abstrakte Modell der Autorschafts-Verifikation schematisch dargestellt, bevor geklärt wird, was die Kompliziertheit dieser Disziplin ausmacht:

- **Bekanntes Dokument:** Im Gegensatz zu der Disziplin der Autorschafts-Attribution wird bei der Autorschafts-Verifikation ein Dokument $\mathcal{D}_{\mathcal{A}_j}$ benötigt, welches keine anonyme Form aufweist, sondern stattdessen die Autorschaft eines vermeintlichen Autors $\mathcal{A}_j$ vorgibt, die nun auf ihre Authentizität verifiziert werden soll.

- **Trainingsmenge:** Analog zu der Autorschafts-Attribution wird auch hier eine Trainingsmenge $\mathbb{D}_{train}$ benötigt, die jedoch eine andere Form aufweist:

  $$\mathbb{D}_{train} = \mathbb{D}_{\mathcal{A}j} = \{\, \mathcal{D}_{1\mathcal{A}_j}, \mathcal{D}_{2\mathcal{A}_j}, \ldots, \mathcal{D}_{\ell\mathcal{A}_j} \,\}$$

  Informell: Sämtliche Beispieldokumente gehören nur einem einzigen Autor $\mathcal{A}_j$ an.

- **Features:** Auch hier wird wieder eine Menge von Features benötigt. Dabei können die gleichen Features wie bei der Autorschafts-Attribution verwendet werden.

- **Verifikations-Modell:** Dies ist die zentrale Komponente der Autorschafts-Verifikation. Das Modell hat die Aufgabe die Entscheidung hinsichtlich der Authentizität von $\mathcal{A}_j$ zu treffen. Im Gegensatz zu der Autorschafts-Attribution ist hier das Resultat nicht die Identität (bzw. Autorschaft) von $\mathcal{D}_{\mathcal{A}_j}$, sondern vielmehr die Entscheidung, ob $\mathcal{A}_j$ tatsächlich als der wahre Autor von $\mathcal{D}_{\mathcal{A}_j}$ in Frage kommt oder nicht.

- **Schwellwert:** Innerhalb der Autorschafts-Verifikation wird ein Schwellwert θ zwingend benötigt (im Gegensatz zu der Autorschafts-Attribution), um die Entscheidung für oder gegen $\mathcal{A}_j$ zu

[1] Siehe dazu Kapitel 4.3.2.

treffen. Wie und vor allem zu welchem Zeitpunkt θ genau definiert werden kann, hängt dabei vom verwendeten Verifikations-Modell ab. Mit „Zeitpunkt“ ist hierbei der Unterschied gemeint, ob θ statisch (also fest) vorgegeben oder jedoch dynamisch (während des Stilvergleichs) erlernt wird. Im Rahmen dieser Arbeit werden beide Varianten für die Festlegung von θ in Betracht gezogen.

Anmerkung: Anstelle eines numerischen Schwellwerts, kann θ alternativ auch als eine Bedingung formuliert werden. Falls θ dabei zutrifft, wird die Entscheidung für $\mathcal{A}_j$ getroffen, andernfalls für $\neg\mathcal{A}_j$.

Die folgende Abbildung stellt das abstrakte Modell der Autorschafts-Verifikation dar:

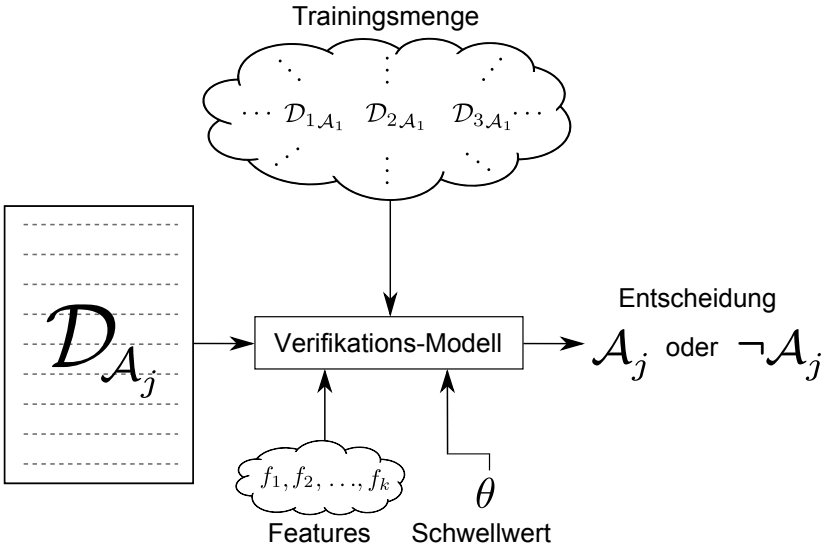


Abbildung 20: Abstraktes Modell der Autorschafts-Verifikation.

Beobachtung: Es entsteht eventuell der Eindruck, als wäre diese Disziplin einfacher zu handhaben, als die der Autorschafts-Attribution. Schließlich besteht $\mathbb{D}_{train}$ lediglich aus Instanzen eines einzigen Autors, sodass hier stilistische Ähnlichkeiten zwischen dem gegebenen Dokument $\mathcal{D}_{\mathcal{A}_j}$ und allen Beispieldokumenten in $\mathbb{D}_{train}$ berechnet werden müssen, um so zu beurteilen, ob es sich bei dem vermeintlichen Autor $\mathcal{A}_j$ auch wirklich um diesen handelt. Tatsächlich trügt jedoch der Schein, da der wahre Autor $\mathcal{A}_{true}$ von $\mathcal{D}_{\mathcal{A}_j}$ nur im Idealfall wirklich $\mathcal{A}_j$ ist, sodass das Verifikations-Modell mit der Aufgabe konfrontiert ist, alle möglichen Autoren abzuweisen und nur für $\mathcal{A}_j$ eine positive Entscheidung abzugeben. Die Frage die hierbei aufkommt ist, wie alle erdenklichen Autoren (außer $\mathcal{A}_j$) abgewiesen werden können, ohne das von ihnen entsprechende Beispieldokumente vorhanden sind? Diese Fragestellung ist die zentrale Problematik der Autorschafts-Verifikation, die dafür sorgt, dass diese Disziplin ein (noch) offenes Problem in der Forschung darstellt.

6.4 Einführung in die Intrinsische Exploration

Der Begriff der intrinsischen Exploration als solcher wurde in der Literatur nicht vorgefunden und wird im Rahmen dieser Arbeit als eine Teildisziplin neu eingeführt. Bei dem Begriff handelt es sich um eine Umformulierung der Phrase *„Die Erkundung vom Innen heraus“*, die diese Disziplin am ehesten beschreibt.

6.4.1 Ziel der intrinsischen Exploration

Unter dem Begriff der intrinsischen Exploration werden stilprüfende Verfahren verstanden, welche die Zielsetzung haben, ein gegebenes Dokument $\mathcal{D}$ von innen heraus nach möglichen Stil-Inkonsistenzen zu untersuchen. Derartige Stil-Inkonsistenzen können darauf hindeuten, dass $\mathcal{D}$ nicht von einem Autor verfasst wurde, sondern stattdessen auf eine kollaborative Zusammenarbeit zurückzuführen ist. Dies stellt insbesondere für die Autorschafts-Attribution bzw. Verifikation ein Problem dar, da Dokumente die unterschiedliche Autorenstile gleichzeitig enthalten und nun attribuiert bzw. verifiziert werden sollen, in der Forschung[1] noch nicht erprobt worden sind.

Im folgenden werden einzelnen Komponenten der intrinsischen Exploration aufgeführt:

- **Dokument:** Analog zu der Autorschafts-Attribution bzw. Verifikation wird auch hier ein Dokument $\mathcal{D}$ benötigt, welches jedoch nicht hinsichtlich der Autorschaft bzw. Authentizität, sondern nur auf mögliche Stil-Inkonsistenzen analysiert werden soll. Je nachdem ob $\mathcal{D}$ in einer weiterführenden Attribution oder Verifikation verwendet wird, muss dieses in einer entsprechenden Form (anonym/nicht-anonym) vorliegen.

- **Zerlegungsstrategie:** Mit Hilfe einer Zerlegungsstrategie wird $\mathcal{D}$ in k Fragmente[2] $\xi_1, \xi_2, \ldots, \xi_k$ aufgeteilt. Diese Fragmente stellen essentielle Bestandteile der intrinsischen Exploration dar, die dafür benötigt werden stilistische Ausreißer in $\mathcal{D}$ identifizieren zu können.

- **Features:** Neben $\mathcal{D}$ wird für die Analyse zusätzlich eine Menge von Features benötigt, die auf $\mathcal{D}$ sowie dessen Fragmente angewendet werden.

- **Stil-Konsistenzmodell:** Diese zentrale Komponente hat die Aufgabe stilistische Ausreißer in $\mathcal{D}$ ausfindig zu machen. Dafür werden die einzelnen Fragmente $\xi_1, \xi_2, \ldots, \xi_k$ und das Gesamtdokument $\mathcal{D}$ anhand von Features sowie einem Klassifikator bzw. einer Metrik verglichen. Werden keine Stil-Inkonsistenzen gefunden, so kann mit einer anschließenden Attribution oder Verifikation bzgl. $\mathcal{D}$ verfahren werden. Liegen jedoch Stil-Inkonsistenzen vor, so wird zunächst die Entscheidung gefällt, dass $\mathcal{D}$ sich in dieser Form nicht für eine weitere Untersuchung (Attribution/Verifikation) eignet. Um diese dennoch zu ermöglichen, ist ein entsprechendes Postprocessing nötig.

- **Schwellwert:** Wie bei der Autorschafts-Verifikation wird auch hier ein Schwellwert θ vorausgesetzt. Mit diesem wird die Entscheidung innerhalb des Stil-Konsistenzmodells getroffen, ob ein Fragment ξ_i (gegenüber anderen Fragmenten) stilistisch in $\mathcal{D}$ auffällt.

- **Postprocessing:** Für den Fall das $\mathcal{D}$ stilistische Ausreißer enthält, kann in dieser Komponente eine spezifische Aufbereitung bzgl. $\mathcal{D}$ vorgenommen werden, um damit eine Attribution/Verifikation durchzuführen, die vorher nicht möglich war. Hierbei bieten sich z.B. die folgenden zwei Möglichkeiten an:

  1. Eliminierung derjenigen Textfragmente $\{\xi_i, \xi_j, \ldots\}$ die Stil-Inkonsistenzen enthalten, um eine Attribution des modifizierten Dokuments $\mathcal{D}'$ dennoch zu ermöglichen.

  2. Bündelung derjenigen Textfragmente $\{\xi_i, \xi_j, \ldots\}$ die Stil-Inkonsistenzen enthalten zu sogenannten Stil-Einheiten $\{E_1, E_2, \ldots\}$, um dadurch eine gesonderte Attribution für jede Einheit E_i zu ermöglichen.

[1] Eine größer angelegte Recherche konnte hierfür keinerlei wissenschaftliche Ausarbeitungen ausfindig machen (weder für die deutsche noch für die englische Sprache).

[2] Siehe dazu die Notation in Kapitel 2.1.

Die zweite Möglichkeit ist dahingehend interessant, weil dadurch eine multiple Attribution[1] ermöglicht werden kann. Dies bedeutet, wenn $\mathcal{D}$ von mehreren Autoren $\mathcal{A}_1, \mathcal{A}_2, \ldots, A_\ell$ verfasst worden ist und die intrinsische Exploration in der Lage war, die einzelnen Stile korrekt in ℓ Stil-Einheiten zu segmentieren, sodass für jeden Autor $\mathcal{A}_j$ eine korrespondierende Stil-Einheit E_j existiert, dann kann damit eine $\ell-$fache Attribution realisiert werden.

Anmerkung: Innerhalb dieser Arbeit wird ein Verfahren für die intrinsische Exploration vorgestellt, welches eine Postprocessing-Komponente enthält. Hierbei werden diejenigen Slices ausfindig gemacht, die sich stilistisch von den anderen Slices unterscheiden. Wie genau mit diesen Slices weiter verfahren wird (Eliminierung oder Bündelung), kann dabei durch den Benutzer festgelegt werden.

Die folgende Abbildung zeigt zusammenfassend ein abstraktes Modell der intrinsischen Exploration:

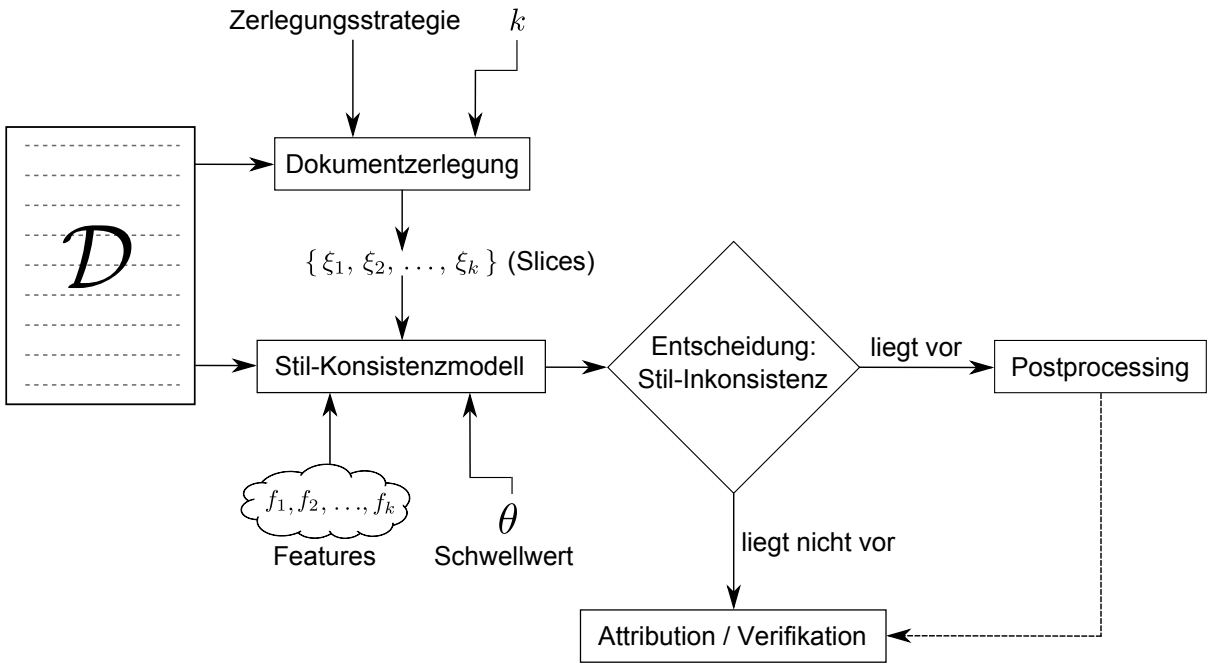


Abbildung 21: Abstraktes Modell der intrinsischen Exploration.

6.5 Verwandte Disziplinen

Neben der Autorschafts-Attribution, Autorschafts-Verifikation und der intrinsischen Exploration, existieren weitere engverwandte Disziplinen, die ebenfalls (zumindest teilweise) unter dem Begriff der Autorschaftsanalyse eingeordnet werden können. Bei einigen der verwandten Disziplinen herrscht dabei Unklarheit darüber, in wie weit diese von der Autorschafts-Attribution bzw. Verifikation abgegrenzt werden können. Dennoch werden sie, der Vollständigkeit halber, in den folgenden Unterabschnitten kurz erläutert.

6.5.1 Textklassifikation

Bei der Textklassifikation[2] handelt es sich um ein Anwendungsgebiet des ML's, dessen primäres Ziel es ist, Texte in zuvor festgelegten Klassen (z.B. „Sport“, „Politik“, „Wirtschaft“, etc.) auto-

[1] In Kapitel 7.3.4 wird auf diese Möglichkeit näher eingegangen.

[2] In der Literatur auch unter dem Begriff Textkategorisierung bekannt.

matisch einzuteilen. Damit kann die Textklassifikation als eine spezielle Art der Klassifikation[1] verstanden werden, [51]. Es besteht ein Zusammenhang zwischen dieser Disziplin und der Autorschafts-Attribution. Letztere kann dabei als eine nicht-inhaltsbasierte Instanz der Textklassifikation angesehen werden, [109]. Mit anderen Worten: In der Textklassifikation erfolgt die Einteilung der Texte anhand der behandelten Thematik oder Genres, während in der Autorschafts-Attribution die Einteilung der Texte anhand des darin enthaltenen Stils[2] vorgenommen wird.

6.5.2 Stilistik

Im Allgemeinen kann die Disziplin der Stilistik als „die Lehre des Stils“ verstanden werden. Nach Radka [105], befasst sich die Stilistik mit der Gestaltung von sprachlichen Äußerungen in unterschiedlichen Kommunikationsbereichen, wobei sich diese in drei folgende Hauptaspekte einteilen lassen:

1. **Der semantische Aspekt:** Verwendung synonymer Ausdrücke in einem Text, um diesen abwechslungsreicher (z.B. in Hinsicht auf die emotionale Ebene) darstellen zu können, [105].
2. **Der formale Aspekt:** Syntaktische „Ausbesserung“ eines Textes, um bestimmte Wortarten wie Verben oder Nomen innerhalb von Phrasen (Verbal- bzw. Nominalphrasen) hervorheben zu können, [105].
3. **Der psychologische Aspekt:** Beachtung des Entstehungskontextes von einem Text, um die darin enthaltene Botschaft besser verstehen zu können, [105].

Den Kern der Stilistik bilden sogenannte rhetorische Stilmittel[3], die vor allem in der Literatur umfangreich eingesetzt werden.

6.5.3 Stilometrie

Golcher [38] beschreibt den Begriff der Stilometrie als eine Art der Textklassifikation, welche jedoch weder Thematik noch Genre eines Textes klassifiziert, sondern vielmehr dessen stilistische Kriterien. Die Stilometrie kann nach seinen Worten nahezu immer mit der Autorenbestimmung gleichgesetzt werden: „*Stilometrie bedeutet fast immer Autorenbestimmung*“, [37].

Stamatatos et al. [46] beschreiben wiederum den gleichen Begriff etwas anders. Laut ihrer Aussage handelt es sich bei der Stilometrie um eine Teilaufgabe der Autorschafts-Attribution, deren Ziel es ist, Autorenstile durch geeignete Features repräsentieren zu können.

6.5.4 Forensische Linguistik

Der Begriff „Forensische Linguistik“ ist in der Literatur leider ebenfalls nicht einheitlich definiert. Nach Fischer, untersucht diese Disziplin *„Sprache auf einen kriminologischen Aspekt“*, [30] um so beispielsweise Urheber von juristisch-relevanten Texten ermitteln zu können. Dies deckt sich ungefähr mit dem gleichen Wortlaut von Liedke: *„Die forensische Linguistik beschäftigt sich als Teilgebiet der Angewandten Sprachwissenschaft mit juristischen Aspekten von Sprache“*, [66, Seite: 12]. Schall[4] verwendet dagegen diesen Begriff synonym zu dem Begriff der Autorenerkennung, [101, Seite: 315], während Heine wiederum den Begriff noch weiter ausdehnt. Laut ihren Worten ist die forensische Linguistik *„im Bereich der Kriminaltechnik angesiedelt und beschäftigt sich mit der Identifikation von Urhebern sprachlicher Äußerungen sowohl im Bereich der forensischen Phonetik als auch im Bereich*

[1] Die Klassifikation in ML ist nicht nur auf Texte beschränkt, sondern kann auch auf Bilder, Videos, Musikstücke, Webseiten, etc. angewendet werden.

[2] Zur Erinnerung: Stil ist ein Phänomen, welches sowohl vom Thema als auch vom Genre unabhängig ist. Siehe dazu die Aussagen von Golcher [37] in Kapitel 2.4.

[3] Im Anhang (Seite: 165) befindet sich eine Liste der wichtigsten rhetorischen Stilmittel für die deutsche Sprache.

[4] Fr. Schall ist Co-Leiterin der Abteilung für forensische Linguistik des Bundeskriminalamts in Wiesbaden, [110].

der Autorenerkennung“, [42, Seiten: 77–78].

Aus all diesen Äußerungen kann geschlossen werden, dass die forensische Linguistik eine eng verwandte Disziplin der Autorschafts-Attribution darstellt, die jedoch explizit auf gerichtsrelevante Texte ausgerichtet ist, während dies bei der Autorschafts-Attribution nicht zwingend verlangt wird. Neben der angedeuteten Disziplin der forensischen Phonetik, existieren zudem weitere Disziplinen innerhalb der forensischen Linguistik:

- **Forensische Stilistik:** Nach einer freien Übersetzung aus [80], handelt es sich bei diesem Begriff um eine Unterdisziplin der forensischen Linguistik. Diese zielt darauf hin Stilistik auf den Kontext der Autorschafts-Verifikation anzuwenden.
- **Forensische Semantik:** Diese junge (und leider kaum erforschte) Disziplin verwendet nach [85] NLP-Komponenten um Hinweise auf Betrügereien/Täuschungen innerhalb natürlichsprachlicher Texte zu identifizieren. Realisiert wird diese Disziplin dabei durch unterschiedliche Wissensressourcen/Wissensdatenbanken, wie beispielsweise Ontologien, spezielle Lexika, Fakten-Datenbanken, Inferenzsysteme (z.B. Cyc[1]) und mehr. Die Autoren aus [85] erklären dabei, dass jene Hinweise dazu verwendet werden sollen, um (kriminelle) Ereignisse zu rekonstruieren, nachdem diese bereits eingetreten sind.

  Der Literatur [85] zufolge kann nicht explizit beurteilt werden, wo diese Disziplin genau einzuordnen ist. Es wird daher vermutet, dass diese neben der forensischen Stilistik als eine weitere Unterdisziplin der forensischen Linguistik zu betrachten ist.

6.5.5 Intrinsische Plagiaterkennung

Die Disziplin der intrinsischen Plagiaterkennung wurde vor allem von Stein et. al [102] geprägt. Sie verfolgt dabei das Ziel potentiell plagiierte Textstellen innerhalb eines Dokuments $\mathcal{D}$ ausfindig zu machen, die vom Gesamtstil des gleichen Dokuments abweichen. Stein et. al erläutern, dass die intrinsische Plagiaterkennung eng mit der Disziplin der Autorschafts-Verifikation verwandt ist. Bei beiden handelt es sich um sogenannte Ein-Klassen-Klassifikations Probleme[2]. Hierbei wird eine Zielklasse definiert, für die (Trainings-)Instanzen existieren. Alle weiteren Instanzen die außerhalb dieser Klasse liegen, werden als Ausreißer bezeichnet, sodass ein Klassifikator die Aufgabe hat diese von der eigentlichen Zielklasse zu unterscheiden. Die Ausreißer stellen dabei die plagiierten Textstellen dar, während die Zielklasse wiederum das Gesamtdokument repräsentiert (bzw. dessen Features), [102].

Beobachtung: Die Disziplin der intrinsischen Plagiaterkennung sollte nicht mit der intrinsischen Exploration verwechselt werden. Technisch gesehen sind beide Disziplinen ähnlich, unterscheiden sich jedoch in ihrem semantischen Kontext. Bei der intrinsischen Plagiaterkennung wird versucht potentiell verdächtige Textstellen ausfindig zu machen, um so beurteilen zu können ob $\mathcal{D}$ plagiiert wurde. In der intrinsischen Exploration wird $\mathcal{D}$ dagegen nach Stil-Inkonsistenzen analysiert, um so festzustellen, ob sich dieser für eine weiterführende Attribution bzw. Verifikation eignet.

6.6 Herausforderungen in der Autorschaftsanalyse

Die Disziplin der Autorschaftsanalyse birgt zahlreiche Herausforderungen, die dazu beigetragen haben, dass diese ein bis heute noch ungelöstes Problem in der Forschung darstellt. In diesem Abschnitt werden die wichtigsten Herausforderungen beschrieben und - zumindest teilweise - mögliche Lösungsansätze vorgeschlagen. Einige dieser Ansätze werden anschließend im weiteren Verlauf der Arbeit genauer erläutert.

[1] Siehe nähere Details dazu unter: `http://www.cyc.com`

[2] Siehe in diesem Zusammenhang Kapitel 4.3.2.

Hinweis: In den folgenden Unterabschnitten wird der Begriff „Autorschaftsanalyse-System“ des Öfteren verwendet und wird daher aus Lesbarkeitsgründen durch AAS abgekürzt.

6.6.1 Sprachabhängigkeit

Ein AAS kann in der Regel nur auf eine Ausgangssprache angewendet werden, da viele der darin befindlichen Komponenten sprachspezifisch beschränkt sind. Die wichtigsten Elemente eines AAS stellen dabei Features als auch NLP-Komponenten dar, mit deren Hilfe Features aus Texten gewonnen werden können.

Was die NLP-Komponenten anbelangt, so besteht eine Abhängigkeit hinsichtlich des zugrundeliegenden Sprachmodells, sodass z.B. ein Parser (und damit auch ein POS-Tagger, Chunker, etc.) der auf englischsprachige Texte trainiert wurde, nicht ohne zusätzliches Training auf deutschsprachige Texte angewendet werden kann, [109]. Um mit mehreren Sprachen umgehen zu können, bedarf es daher trainierte Sprachmodelle, die wiederum zu einer Sprachabhängigkeit führen.

Was die Features betrifft, so existiert hier ebenfalls eine Abhängigkeit hinsichtlich der Texte (und damit der vorliegenden Sprache), da in vielen Fällen Buchstaben, Buchstaben $n-$Gramme, Wörter oder Phrasen die Features darstellen, sodass auch hier eine direkte Sprachabhängigkeit hervorgerufen wird. Es existieren zwar vereinzelte Ansätze[1], die mit Hilfe von sogenannten $n-$Gramm Profilen auf der sprachneutralen Byte-Ebene versuchen eine Autor-Attribution auf multilinguale Texte zu ermöglichen. Jedoch lässt sich schwer sagen, ob diese Verfahren als praxistauglich einzustufen sind, da dazu nur wenig Forschungsarbeit erbracht wurde.

6.6.2 Verrauschte Trainigsdaten

Die Qualität der Dokumente in $\mathbb{D}_{train}$ spiegelt sich in der Gewinnung von Features und damit auch in der Erkennungsgenauigkeit des AAS 1:1 dar. Verrauschte Trainigsdaten[2] zeichnen sich unter anderem durch das folgende Prinzip von Ursache und Wirkung aus:

Rechtschreibfehler	⇒	Falsche Wortart Klassifikation seitens eines POS-Taggers.
Grammatikfehler	⇒	Falsche Zerlegung durch einen Chunker.
Viele Interpunktionszeichen	⇒	Inkorrekte Satzende Erkennung durch ein Sentence Tokenizer.
Fremdsprachige Textstellen	⇒	Inkorrekte morphologische Zerlegung seitens eines Stemmers.

Diese Herausforderungen lassen sich (zumindest teilweise) mit Text-Normalisierungsmethoden gut bewerkstelligen, die bereits im Kapitel 2.6 erläutert wurden.

6.6.3 Kollaborativ erstellte Trainigsdokumente

Befinden sich in einer Trainigsmenge $\mathbb{D}_{train}$ kollaborativ erstellte Dokumente, so können diese die Erkennungsgenauigkeit eines AAS negativ beeinflussen. Aus diesem Grund sollte sichergestellt werden, dass $\mathbb{D}_{train}$ nur solche Dokumente enthält, die eine eindeutige Autorschaft aufweisen. In der Praxis kann jedoch diese Eigenschaft nicht immer erfüllt werden. So wäre es z.B. undenkbar eine Trainigsmenge mit der Größenordnung von 1000 Dokumenten manuell nach entsprechenden Dokumenten zu filtern, sodass hier eine andere Strategie gefunden werden muss.

---

[1] Siehe dazu beispielsweise den Ansatz von Keselj et al. in [108].

[2] Mit starken Rauschen ist insbesonders bei Nachrichten in sozialen Netzwerken, E-Mails, Forenbeiträge oder Webblogs zu rechnen.

Eine mögliche Lösung wäre z.B. der Einsatz der intrinsischen Exploration, um Dokumente die ein stilistisches Rauschen enthalten für eine Attribution oder Verifikation entsprechend aufzubereiten. Dabei ist es jedoch wichtig zu wissen, in welchem Ausmaß Rauschen in den Texten vorkommt. Falls z.B. ein Dokument $\mathcal{D} \in \mathbb{D}_{train}$ von mehreren Autoren verfasst wurde, sodass jeder Abschnitt einen einzigen Autorenstil darstellt, so würde sich die Möglichkeit[1] anbieten $\mathcal{D}$ in entsprechende Fragmente $\xi_1, \xi_2, \ldots$ aufzuteilen, um diese anschließend einzeln zu analysieren. Handelt es sich bei $\mathcal{D}$ jedoch um ein Dokument mit vermischten Stilen, bei dem z.B. Sätze von unterschiedlichen Autoren paraphrasiert worden sind, so könnte eine andere Methode angewendet werden, um solche Sätze zu finden und anschließend aus $\mathcal{D}$ zu entfernen. Die Methode wird jedoch fehlschlagen, falls die Mehrheit der Sätze in $\mathcal{D}$ solche Paraphrasierungen aufweisen.

6.6.4 Nicht zutreffende Grundannahme

Die Grundannahme der Autorschafts-Attribution besagt, dass der unbekannte Autor ε von $\mathcal{D}_\varepsilon$ innerhalb der Trainingsmenge $\mathbb{D}_{train}$ bekannt sein muss. In der Praxis trifft jedoch diese Grundannahme in vielen Fällen nicht zu, sodass hier eine Lösung gefunden werden muss, wie das AAS in solchen Fällen verfahren soll.

Eine Möglichkeit die sich hier anbietet ist einen Schwellwert θ zu definieren, welcher bei einer Über- bzw. Unterschreitung keine Attribution zulässt. Diese Lösung erscheint zwar plausibel, erfordert jedoch eine ausgeklügelte Strategie um θ dynamisch (also zur Laufzeit des AAS) bestimmen zu können. Eine statische Festlegung von θ ist dagegen nur in Ausnahmefällen sinnvoll, da das AAS hinsichtlich Parametern wie verwendete Metrik/Klassifikator, Features, Dokumentlängen, etc. nicht bzw. kaum skalieren könnte.

Eine andere Möglichkeit besteht darin, anstelle der Identität des unbekannten Autors, zumindest alternative Anhaltspunkte bestimmen zu können, um so ein ungefähres Bild von diesen zu erhalten. Möglich wären hier unter anderem Hinweise auf das Geschlecht, Alter, Bildung oder gar die Verwendung von Deutsch als Zweitsprache. Im nächsten Abschnitt wird dazu näheres erläutert.

6.6.5 Attribution anhand einer Trainingsmenge mit zahlreichen Autoren

Eine weitere Herausforderung der Autorschafts-Attribution stellt eine Trainingsmenge $\mathbb{D}_{train}$ mit sehr vielen Autoren dar (z.B. $|\mathbb{A}| \gtrapprox 10000$). In einem solchen Szenario würde jedes heutzutage verfügbare AAS mit großer Wahrscheinlichkeit scheitern, da $\mathbb{D}_{train}$ zwangsläufig Dokumente mit kaum unterscheidbaren Autorenstilen enthalten würde. Daher wäre hier eine spezielle Filterungsstrategie denkbar, die mit stilunabhängigen Parametern versucht den Umfang von $\mathbb{D}_{train}$ derart zu reduzieren, sodass nicht in Frage kommende Autoren (bzw. deren Dokumente) von der eigentlichen Attribution anhand des Stilvergleichs ausgeschlossen werden können.

Ein möglicher Parameter der eine solche Filterung realisieren könnte, wäre beispielsweise das Geschlecht der Autoren die in $\mathbb{D}_{train}$ vorkommen. Wenn z.B. $\mathcal{D}_\varepsilon$ von einer weiblichen Person verfasst wurde und $\mathbb{D}_{train}$ überwiegend aus Dokumenten $\mathcal{D}_{i_{\mathcal{A}_j}}$ besteht, die von männlichen Personen verfasst worden sind, dann könnte mit Hilfe einer Geschlechtsklassifizierung[2] (engl. *Gender Classification*) der Umfang von $\mathbb{D}_{train}$ entsprechend reduziert werden, was sich gleichzeitig auf die Laufzeit und Erkennungsgenauigkeit des AAS positiv auswirken würde. Neben der Klassifikation des Geschlechts können zudem weitere Parameter (z.B. Alter, Dialekt, Bildungsgrad, Sozialschicht, etc.) auf die verbliebenen $\mathcal{D}_{i_{\mathcal{A}_j}}$ angewendet werden, um dadurch $\mathbb{D}_{train}$ und damit Autorenmenge $\mathbb{A}$ noch weiter zu verkleinern. Auch wenn derartige Filterungsstrategien zunächst sinnvoll erscheinen mögen, so haben diese auch einige nennenswerte Nachteile. Einer davon ist, dass für jeden Parameter eigens ein Modell benötigt wird, welches anhand eines Klassifikators im Vorfeld erlernt werden muss und dadurch mit einem

[1] Ein Ansatz der sich dieser Problematik annimmt wird in Kapitel: 7.3 präsentiert.

[2] Siehe dazu die folgende Literaturempfehlung: [77, 83, 84].

aufwändigen Training verbunden ist. Zudem sind solche Klassifikationen[1] in der Praxis bisher nicht ausreichend erprobt, sodass hier mit (möglicherweise) hohen Fehlerraten zu rechnen ist.

Alternativ dazu kann ein anderer Ansatz[2] verwendet werden, der innerhalb dieser Arbeit entwickelt wurde. Der Ansatz basiert dabei auf der Kombination zweier Attributions-Verfahren. Das erste Verfahren filtert dabei zunächst diejenigen Autoren in $\mathbb{D}_{train}$ raus, deren Beispieltexte kaum gemeinsame $n-$Gramme mit $\mathcal{D}_{\varepsilon}$ aufweisen, während das zweite Verfahren im Anschluß daran die Attribution auf die verbliebene Trainingsmenge $\mathbb{D}'_{train}$ ausführt. Die Besonderheit dieses Ansatzes liegt darin, dass sich beim ersten Verfahren die Anzahl der zu filternden Autoren prozentual regulieren lässt. Dadurch reduziert sich die Laufzeit des zweiten Verfahrens, während gleichzeitig die Qualität hinsichtlich der Attribution gesteigert werden kann, da hier weniger Autoren in $\mathbb{D}'_{train}$ enthalten sind und dadurch eine Fehlentscheidung gemindert wird.

6.6.6 Textsortenunabhängige Features

Features sind der essentielle Bestandteil eines jeden Autorschaftsanalyse-Szenarios. Aus diesem Grund sollten sie die Eigenschaft besitzen, Autorenstile möglichst eindeutig und über Textsorten hinweg, unterscheiden zu können. Leider weisen viele Features eine eindeutige Diskriminierungseigenschaft nicht aus (erst recht nicht textsortenübergreifend), sodass aus diesem Grund Forscher in der Vergangenheit zahlreiche Features vorgeschlagen haben, die zumindest für eine Großzahl von Textsorten brauchbare Ergebnisse erzielen konnten. Unter anderem wurden hierbei Funktionswörter[3] als mögliche Features empfohlen, da diese einige interessante Eigenschaften bergen, die sich für textsortenübergreifende Analyseszenarien gut eignen. Die wichtigste Eigenschaft ist dabei, dass Funktionswörter keine Semantik enthalten (anders als z.B. bei Verben oder Nomen) und somit von der Textsorte unabhängig sind. Des Weiteren sind Funktionswörter praktisch in jedem Text enthalten und machen prozentual gesehen nahezu die Hälfte dessen aus. Um diese Annahme zu verifizieren wird im Folgenden eine Untersuchung erläutert, bei der fünf Korpora hinsichtlich 763 Funktionswörtern analysiert wurden. Der Dokumentumfang von jedem Korpus betrug dabei exakt 50 KByte ($\approx$ 12 DIN-A4-Seiten). Die Korpora die dabei verwendet wurden sind wie folgt beschrieben:

1. **Romane Korpus:** Dieser Korpus wurde aus Textfragmenten von insgesamt 10 Romanen zusammengestellt. Die Romane wurden dabei aus der Webseite des Guttenberg-Projekts herruntergeladen (`Projekt.Gutenberg.de`) . Bei sämtlichen Romanen handelt es sich um knapp einem Jahrhundert alte Werke.
2. **Nachrichten Korpus:** Für die Zusammenstellung von diesem Korpus wurden 449 Sätze aus dem Wortschatz-Leipzig Korpus (`Wortschatz.Uni-Leipzig.de`) entnommen. Es handelte sich hierbei um einen satz-tokenisierten Korpus, dessen Sätze überwiegend aus Nachrichtentexten (genauer Onlineversionen von Zeitungen) entnommen wurden.
3. **Dissertationen Korpus:** Dieser Korpus wurde aus Textfragmenten von fünf Dissertationen erstellt, die von Doktoranden der TU Darmstadt verfasst wurden (eine davon stammt von einem Betreuer dieser Arbeit).
4. **Forenbeiträge Korpus:** Für die Erstellung von diesem Korpus wurden Beiträge aus zwei Foren (`Gesundheit.de`) und (`Reise-Forum.org`) verwendet.
5. **Internet-Blogs Korpus:** Hierfür wurden Textfragmente aus 10 Internet-Blogs entnommen (`Bloggerei.de`). Dabei handelte es sich vorwiegend um musikbezogene Themen.

Die Untersuchung bzgl. der Verteilung von Funktionswörtern auf die Korpora ergab das folgende Resultat:

---

[1] In Kapitel 7.4.1 wird dazu ein therotischer Ansatz erläutert.

[2] In Kapitel 7.1.3.3 wird dieser Ansatz im Detail erläutert.

[3] Siehe dazu unter anderem: [33, 94, 116].

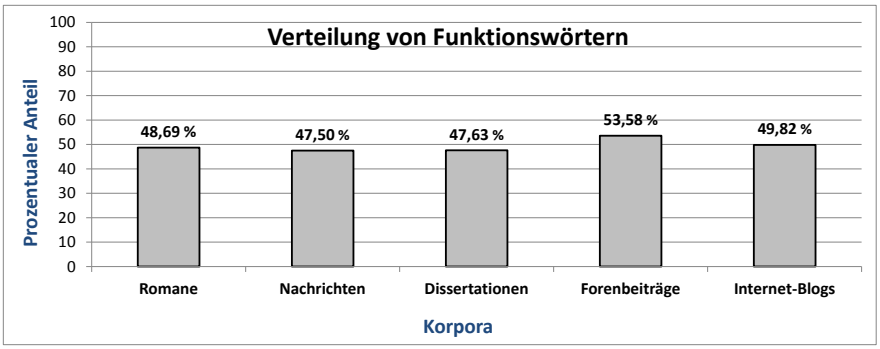


Abbildung 22: Verteilung von Funktionswörtern in deutschen Texten.

Hieraus lässt sich erkennen, dass die Verteilung von Funktionswörtern in jedem Korpus (und damit der Textsorte) nahezu gleich ist. Nicht ersichtlich ist jedoch die Tatsache, dass die Häufigkeitsverteilung einzelner Wörter sich von Autor zu Autor unterscheidet. Ein Experiment in Kapitel 10 wird dies genauer erläutern.

6.6.7 Dokumentlänge

Die Textlänge von $\mathcal{D}_\varepsilon$ geht mit der Erkennungsgenauigkeit des AAS einher, da diese die Anzahl der Features direkt beeinflusst. In der Forschung existieren dazu zahlreiche Kennzahlen mit denen erfolgreiche Autorschaftsanalyse-Szenarien durchgeführt wurden. In [99] erwähnt Stamatatos z.B. Attributions-Szenarien, die mit weniger als 1000 Wörter ausgekommen sind. Gleichzeitig mahnt er jedoch auch an, dass eine generelle Größe nicht angegeben werden kann bzw. darf. Entscheidend für ein erfolgreiches Analyse-Szenario ist anstelle der Wortanzahl vielmehr die Wortvielfalt die einen Text prägen. Schließlich bringt es wenig, wenn $\mathcal{D}_\varepsilon$ 10^6 Wörter enthält, die allesamt identisch sind.

Im Rahmen dieser Arbeit konnten brauchbare Ergebnisse ($> 80\%$ Genauigkeit) mit ca. 2 KByte Textumfang erzielt werden, wobei an erster Stelle entscheidend war, welche Features dabei verwendet wurden. In der Regel sollten für kurze Dokumente nur Features aus der niedrigsten Feature Ebene (also die Zeichen Ebene) benutzt werden, da der Recall für zeichenbasierte Stilmerkmale stets am höchsten ist.

Ein anderer wichtiger Aspekt betrifft die Größe der Trainingsdokumente die gegen $\mathcal{D}_\varepsilon$ hinsichtlich des Stils verglichen werden müssen. Diese sollten dabei stets größer als $\mathcal{D}_\varepsilon$ gewählt werden. Interne Tests haben für ein Testdokument $\mathcal{D}_\varepsilon$ mit $length(\mathcal{D}_\varepsilon) \approx 2$ bis 4 KByte ergeben, dass sich Längen von Trainingsdokumente mit $\lambda \cdot length(\mathcal{D}_\varepsilon)$ KByte für $\lambda \geq 4$ als nützlich erwiesen haben.

6.6.8 Stilistische Varietäten eines Autoren

Die stilistische Varietät eines Autors stellt die wohl schwierigste Herausforderung innerhalb der Autorschaftsanalyse dar. Es existieren sprachlich versierte Autoren, die in der Lage sind zwei Texte $\mathcal{D}_1$ und $\mathcal{D}_2$ derart zu verfassen, dass selbst ein geübter Experte Schwierigkeiten haben würde zu urteilen, ob $\mathcal{D}_1$ und $\mathcal{D}_2$ von ein und demselben Autor geschrieben worden sind. In der Forschung wird dies als das grundsätzliche Problem der Autorschafts-Attribution bezeichnet (engl. *„The Fundamental Problem of Authorship Attribution“*). Koppel et al. erläutern in ihrem gleichnamigen wissenschaftlichen Beitrag [59], dass die Bewältigung dieser Herausforderung gleichzeitig die Lösung eines jeden Autorschafts-Attributions Problem zur Folge hätte. Für die Lösung dieses Problems schlagen sie dabei einen komplexen und leider nicht ganz nachvollziehbaren Ansatz vor, der hauptsächlich auf ihre frühere Arbeit in [76] basiert.

Zusammengefasst geht es in dem Ansatz aus [59, Seite: 289] darum, ausgehend von $\mathcal{D}_{\mathcal{A}x}$ und $\mathcal{D}_{\mathcal{A}y}$ zunächst eine Reihe von künstlichen Kandidaten[1] $\mathbb{A} = \{ \mathcal{A}_{y_1}, \mathcal{A}_{y_2}, \ldots, \mathcal{A}_{y_n} \}$ zu erzeugen. Anschließend wird mit Hilfe der Methode aus [76] versucht zu bestimmen, ob $\mathcal{D}_1$ vom selben Autor von $\mathcal{D}_2$ oder vom $\mathcal{A}_{y_i} \in \mathbb{A}$ oder stattdessen von keinem der beiden verfasst wurde. Falls und nur falls ein signifikanter Wert bestimmt werden konnte, der angibt das $\mathcal{D}_{\mathcal{A}x}$ und $\mathcal{D}_{\mathcal{A}y}$ vom selben Autor verfasst worden ist, dann gilt $\mathcal{A}_x = \mathcal{A}_y$. Als Performanzmaße geben Koppel et al. einen Recall von 83% und eine Precision von 90% an, [59, Seite: 289].

6.6.9 Stileinfluss durch Internet/Software-Lösungen

Dank des Internets und zahlreicher Software-Lösungen aus dem Office-Bereich, haben Autoren heutzutage Unmengen an Möglichkeiten, um ihre Texte anhand von Parametern wie z.B. Rechtschreibung, Grammatik, Synonymauswahl, Redewendungen oder gar einer Stilanalyse-Prüfung, zu optimieren. Im Internet finden sich beispielsweise die folgenden Portale, mit deren Hilfe Texte „perfektioniert“ werden können:

`Woxikon.de`	Wörterbücher für Antonyme, Synonyme sowie Fremdwörter.
`Redensarten-Index.de`	Nachschlagewerk für Redensarten, Redewendungen, idiomatische Ausdrücke und feste Wortverbindungen.
`Wortschatz.Uni-Leipzig.de`	Enzyklopädie für Wörter mit zahlreichen assoziierten Daten, z.B. morphologische Zerlegung, Grammatik- bzw. Pragmatikangaben, semantische Relationen (Synonyme, Antonyme, Meronyme), Kookkurrenzen und vieles mehr.
`Schreiblabor.com`	Enthält nützliche Werkzeuge wie z.B. Füllwörter-Überprüfung oder Text-Analyse.
`Duden.de`	Zahlreiche Services wie z.B. Textprüfung (Rechtschreibung, Grammatik), Wendungen, Redensarten, Sprichwörter, (visuelle) Semantik-Netze, etc.

Bei einem übermäßigen Gebrauch dieser Werkzeuge bzw. Ressourcen, können ursprüngliche Autorenstile verzerrt werden und damit zu falschen Ergebnissen seitens des AAS führen. Um dieser Problematik entgegenzutreten kann unter anderem die intrinsiche Exploration eingesetzt werden. Hier wäre es z.B. denkbar Textstellen, die bedingt durch die oben genanten Werkzeuge/Ressourcen modifiziert wurden, zunächst aus einem gegebenen Dokument $\mathcal{D}$ zu entfernen und dann eine weiterführende Analyse (Attribution/Verifikation) auf dem bereinigten Dokument $\mathcal{D}'$ anzuwenden. Der Ansatz macht jedoch nur dann Sinn, falls $\mathcal{D}$ verhältnismäßig wenig modifizierte Textstellen enthält (ca. 5 bis 10%). Sind dagegen zu viele solcher Modifizierungen in $\mathcal{D}$ präsent, die dazu noch gleichmäßig über das gesamte Dokument verteilt sind, so wird der Ansatz höchstwahrscheinlich scheitern.

6.6.10 Zeitliche Stiländerung eines Autoren

Im Laufe ihres Lebens ändern die meisten Autoren ihren Schreibstil in einem unbestimmten Ausmaß. Diejenigen Faktoren, die eine Stiländerung bewirken, können hierbei aus der Abbildung 2.4 (Seite: 23) entnommen werden. In manchen Autorschaftsanalyse-Szenarien kann[2] es nun vorkommen, dass ein zeitlich aktuelles Dokument $\mathcal{D}$ analysiert werden soll, aber die zugrundeliegende Trainingsmenge dabei nur ältere Beispieldokumente enthält, die sich aufgrund des veränderten Stils für die Analyse nicht eignen.

Um eine Analyse von $\mathcal{D}$ dennoch zu ermöglichen, müssen daher Features gefunden werden, die gegenüber dem zeitlichen Verlauf zwischen $\mathcal{D}$ und den Trainingsdokumenten resistent geblieben sind.

[1] In [59] wird dabei der Begriff „Betrüger“ (engl. *Imposters*) verwendet.

[2] Anmerkung: Im Rahmen einer Recherche konnte keine wissenschaftliche Studie (im deutschsprachigen Raum) ermittelt werden, die eine Stiländerung von Autoren über viele Jahre hinweg untersucht hat.

Derartige Features sind jedoch schwer zu finden, da Autoren während ihrer Entwicklung nicht nur ihren Stil sondern auch ihre grammatikalischen Fähigkeiten verändern und dadurch beispielsweise zeichen- oder wortbasierte Features unbrauchbar werden können. Es wird jedoch vermutet, dass eine spezifische Kategorie von Features existiert, die in Texten aus unterschiedlichen Zeitabschnitten vorkommen, wobei jedoch ausreichend lange Texte vorliegen müssen. In Kapitel 7.4.5 wird ein theoretischer Ansatz vorgestellt, der diese Herausforderung zu lösen versucht.

6.6.11 Schwache Generalisierungsfähigkeit eingesetzter NLP-Werkzeuge

Eine Autorschaftsanalyse basiert unter anderem auf einer Reihe von Werkzeugen aus dem NLP Umfeld. Dazu zählen unter anderem Tokenizer, Sentence-Tokenizer, POS-Tagger, Chunker, etc. Es ist anzumerken, dass die meisten dieser Werkzeuge statistischen Modellen zugrunde liegen, die somit bzgl. ihrer Entscheidungen nicht immer unfehlbar sind. Die Konsequenz daraus ist, dass die fertigen „Produkte“ der NLP-Werkzeuge (z.B. resultierende Wortarten bei einem POS-Tagger) die Qualität eines AAS direkt beeinflussen können.

Auf dem Gebiet der Autorschaftsanalyse finden sich einige wissenschaftliche Beiträge, die ältere Texte (z.B. aus dem Projekt Gutenberg) untersuchen und dabei NLP-Werkzeuge anwenden, die auf moderne Texte[1] trainiert worden sind. In solchen Fällen werden die eingesetzten NLP-Werkzeuge sehr wahrscheinlich Fehler produzieren, da sowohl lexikalische als auch syntaktische Eigenenarten der damaligen Sprache, in der heutigen nicht mehr vorkommen. Um dieses Problem zu umgehen, existieren unterschiedliche Ansätze. So können NLP-Werkzeuge z.B. auf entsprechende Domänen (Biologie, Medizin, Soziologie, etc.) trainiert werden, um so zumindest mit Texten umzugehen, die ein spezielles Vokabular aufweisen. Dieser Ansatz erfordert jedoch einen großen Aufwand, da die Daten mit denen die NLP-Werkzeuge trainiert werden, manuell von Experten annotiert werden müssen und es hierfür in der Praxis öfters an zeitlichen, fachlichen und finanziellen Ressourcen mangelt. Fatal wäre es hierbei auf Experten zu verzichten, da die Trainingsdaten der NLP-Werkzeuge (die nicht mit korrekten linguistischen Annotationen versehen werden) automatisch zu Fehlentscheidungen der statistischen Modelle führen werden und dies den Einsatz der NLP-Werkzeuge wertlos machen würde.

Eine andere Möglichkeit um dieses Problem anzugehen ist der Einsatz einer sogenannten *Domain Adaptation*[2], welche eine dynamische Anpassung an eine beliebige Domäne verspricht. Diese Technik stellt jedoch einen aktuellen Forschungsgegenstand dar, sodass ein praxistauglicher Einsatz (noch) nicht erprobt wurde.

6.6.12 Evaluierungsschwierigkeiten

Auf dem Gebiet der Autorschaftsanalyse wurden zahlreiche Forschungsarbeiten veröffentlicht. Viele von ihnen sind jedoch untereinander nicht vergleichbar. Die Ursachen dafür liegen zahlreichen Faktoren zugrunde, wie z.B. verwendete Ausgangssprache, Korpora, NLP-Werkzeuge, Features, Klassifikatoren sowie die Autorschaftsanalyse-Algorithmen selbst, inklusive deren Parametrisierung (Text-Normalisierung, Anzahl von Trainingsdokumenten, Dokumentlängen, etc.).

Um diese Problematik in den Griff zu bekommen, bedarf es daher zwingend festgelegte Standards und einige Voraussetzungen. Denkbar wären beispielsweise:

- Entwicklung einheitlicher und öffentlich zugänglicher Korpora (inklusive einer wohldefinierten Annotation).
- Festlegung von State of the Art Algorithmen, welche in eigenen Experimenten als Baselines benutzt werden können. Hierbei sollte vor allem die Parametrisierung lückenlos offengelegt werden.

[1] Zahlreiche NLP-Werkzeuge werden auf (moderne) Zeitungstexte trainiert, siehe dazu z.B. [10].

[2] Siehe dazu die Literaturempfehlung [49, 52]

- Verwendung einheitlicher Werkzeuge inklusive deren Konfigurartion (z.B. Trainingsdaten von NLP-Werkzeugen oder die Parametrisierung der eingesetzten Klassifikatoren/Metriken).

In den letzten Jahren wird ein Bestreben für die Einführung solcher Standards beobachtet. So ist beispielsweise im Jahre 2007 eine sogenannte SIGIR[1] Gruppe mit der Bezeichnung PAN („Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection“) entstanden, die sich zum Ziel gemacht haben, Forscher aus dem Gebiet der Autorschaftsanalyse zusammenzuführen, [98]. Unter anderem wurden von dem PAN-Komitee frei zugängliche Korpora veröffentlicht, mit deren Hilfe sich implementierte Verfahren aus der Autorschaftsanalyse als auch aus verwandten Disziplinen untereinander vergleichen lassen. Ursprünglich fing PAN als ein Workshop an und hat sich mittlerweile zu einem der bedeutendsten Organisationen im Bereich der Autorschaftsanalyse entwickelt.

Anmerkung: Im Rahmen dieser Arbeit konnten die PAN Korpora leider nicht eingesetzt werden, da die meisten Dokumente darin englischsprachig sind.

6.7 Anwendungsgebiete der Autorschaftsanalyse

Die Autorschaftsanalyse hat in der Praxis vielfältige Anwendungsmöglichkeiten. In diesem Abschnitt werden dazu einige bekannte sowie weniger bekannte Einsatzgebiete vorgestellt.

6.7.1 Erkennung von Plagiatsvergehen

Ein bekanntes Anwendungsgebiet der Autorschaftsanalyse ist die Aufdeckung von Plagiatsvergehen, [109]. Hierbei kann die Aufdeckung unterschiedlich realisiert werden. So kann beispielsweise eine intrinsische Plagiatserkennung auf ein gegebenes Dokument $\mathcal{D}_{\mathcal{A}}$ durchgeführt werden, um somit (ohne externe Vergleichsmöglichkeiten) beurteilen zu können, ob auffällige Textpassagen enthalten sind, die sich vom Gesamtstil von $\mathcal{D}_{\mathcal{A}}$ unterscheiden. Werden Abschnitte gefunden, die stilistische Ausreißer darstellen, so kann dies häufig ein Indiz[2] dafür sein, dass $\mathcal{D}_{\mathcal{A}}$ plagiiert wurde. Sind Vergleichsmöglichkeiten vorhanden, so kann stattdessen auch ein Attributions-Verfahren für die Plagiatsaufdeckung eingesetzt werden. Das folgende Beispiel beschreibt dazu eine Möglichkeit.

Sei $\mathcal{D}_{\mathcal{A}_1}$ ein gegebenes Dokument, welches von einem Autoren $\mathcal{A}_1$ verfasst wurde und $\mathbb{D}_{train}$ eine festgelegte Trainingsmenge, die Beispieldokumente von $\mathcal{A}_1$ und anderen Autoren $\mathcal{A}_2$, $\mathcal{A}_3$, ... enthält sowie m Feature-Kategorien F_1, F_2, ..., F_m gegeben. Wenn nun ein Attributions-Verfahren in m Iterationen[3] nicht in der Lage ist $\mathcal{A}_1$ bezüglich $\mathcal{D}_{\mathcal{A}_1}$ richtig zuzuordnen, sondern stattdessen immer einen anderen festen Autor $\mathcal{A}_j$ bestimmt, so könnte dies darauf deuten das $\mathcal{A}_1$ Textabschnitte von $\mathcal{A}_j$ übernommen hat. Im schlimmsten Fall könnte es sich bei $\mathcal{A}_1$ sogar um $\mathcal{A}_j$ handelt (Ghostwriting[4]).

6.7.2 Demaskierung von Pseudonymen

Pseudonyme spielen im Internet eine bedeutende Rolle. Dort sind diese häufig in Foren, Blogs oder Kommentierungen[5] anzutreffen. Die Benutzung von Pseudonymen erfolgt dabei nicht ohne Grund, da in der Regel nur wenige Nutzer ihre wahre Identität im Internet preisgeben wollen. In der Praxis kann es hierbei häufig vorkommen, dass durch eine solche Identitätsverschleierung Missbrauch betrieben werden kann. Dieser äußert sich z.B. durch Beleidigungen, rassistische Äußerungen, sexuelle Belästigungen oder Gewaltdrohungen seitens der anonymen Nutzer. Foren-Administratoren können in vielen

[1] SIGIR ist ein Akronym für: „Special Interest Group on Information Retrieval“.

[2] Anmerkung: Ein Indiz ist kein Beweis sondern vielmehr ein Anhaltspunkt (bzw. Hinweis.)

[3] Jede $i-$te Iteration entspricht hierbei der Anwendung der Feature-Kategorie F_i auf $\mathcal{D}_{\mathcal{A}_1}$ sowie den Trainingsdokumenten.

[4] Bei einem Text, welcher durch Ghostwriting entstanden ist, handelt es sich nicht um ein Plagiat sondern um eine Arbeit, die im Auftrag, durch eine andere Person angefertigt wurde.

[5] Hierzu zählen (unter anderem) Kommentare in sozialen Netzwerken, Film- bzw. Bücherrezensionen, Meinungen über Produkte/Restaurants, etc.

Fällen nur bedingt weiterhelfen, wenn es um die Offenlegung der Identität der betroffenen Nutzer geht. Die Gründe dafür sind beispielsweise:

- **Falsche Angaben:** Viele Benutzer tragen bei Forum-Registrierungen falsche Daten ein, die zumeist von Administratoren oder entsprechenden Systemen nicht auf ihre Gültigkeit überprüft werden. Hierzu zählen unter anderem Alter, Geschlecht, Ortsangaben, etc.
- **Unzureichende Angaben:** Um eine Zugangsberechtigung für die Nutzung eines Forums zu erhalten, sind oft nur wenige Daten tatsächlich erforderlich. In der Regel wird lediglich ein Benutzername sowie eine gültige E-Mail Adresse zwingend vorausgesetzt, während andere identitätsbezogene Daten optional sind und von Benutzern kaum vollständig ausgefüllt werden.
- **Unbrauchbare Web-Logdateien:** In Web-Logdateien sind zahlreiche personenbezogene Daten vorhanden, wie z.B. wann und von wo eine Webseite aufgerufen wurde, welche Dateien zu welcher Zeit heruntergeladen wurden oder wieviel Zeit ein Benutzer auf einer bestimmten Seite verbracht hat. Das größte Problem dabei ist, dass es sich hierbei um indirekte personenbezogene Daten handelt, sodass die Bestimmung von wahren Benutzeridentitäten seitens der Foren-Administratoren hierdurch nicht ohne Weiteres ermöglicht werden kann.

In solchen Fällen können Attributions-Verfahren eingesetzt werden um Identitäten, die sich hinter Benutzernamen (bzw. Pseudonyme) verschleiern, zu demaskieren. Der Gedanke dabei ist, dass eine Stil-Anonymisierung mit deutlich mehr Aufwand verbunden ist, als das Umgehen bzw. Eintragen von falschen Registrierungsdaten. Voraussetzung hierbi ist jedoch, dass die wahren Identitäten bzgl. der anonymen Nutzer vorhanden sein müssen (Grundannahme). Diese Voraussetzung lässt sich in der Praxis bis zu einem gewissen Grad erfüllen. So können als Trainingsdokumente verschiedene Ressourcen dienen, wie etwa Blogs, (Erfahrungs-)Berichte oder Bewertungen[1]. Im Gegensatz zu Foren kommt es hier häufiger vor, dass Nutzer ihre Vor- und Nachnamen preisgeben, da sie auf Feedback bzw. Gegenbewertungen seitens anderer Nutzer hoffen und sich daher keine Gedanken darüber machen, dass sie ihre wahren Identitäten mitsamt ihres Schreibstils unbewusst assoziiert haben.

Wenn genügend solcher Ressourcen gesammelt werden, so lassen sich Autorschafts-Attributionen auf unbekannte Quellen anwenden, um dadurch die Demaskierung von Pseudonymen zu ermöglichen. Um im Internet gezielt an diese Ressourcen zu gelangen, können unter anderem fokussierte Crawler[2] eingesetzt werden.

6.7.3 Unterstüzung in der forensischen Analyse

In der forensischen Analyse kommen Autorschaftsanalyse-Verfahren in unterschiedlichen Szenarien verstärkt zum Einsatz[3]. So werden z.B. Attributions-Verfahren auf SMS [48] oder Instant Messaging Nachrichten [78] bzw. auch Verifikations-Verfahren auf Office-Dokumente oder E-Mails [47] angewendet, um Autorschaften demaskieren bzw. verifizieren zu können. Da beide Varianten auf Trainingsdokumente angewiesen sind, kommt hier die Frage auf, wie Forensiker an entsprechende Beispieltexte gelangen können. Wenn beispielsweise Rechner oder mobile Geräte (Laptops, Smartphones, etc.) beschlagnahmt werden, so finden sich darauf in der Regel mehrere textuelle Daten des Täters, die als Trainingsdokumente fungieren können. Dazu zählen insbesondere E-Mails oder Messenger Nachrichten (z.B. ICQ/MSN, XING, Facebook, etc.), da diese geräteübergreifend abgerufen werden können und dabei öfters offline in Form von lokal gecachten Dateien verfügbar gemacht werden. Anders dagegen sieht es in solchen Fällen aus, bei denen nur ein Text vorliegt (z.B. Drohbrief, Erpresserbrief, etc.), sodass hier weder eine Attribution noch Verifikation sinnvoll erscheinen würde. Stattdessen kann hier der Text (je nach Kontext/Fall) mit Hilfe einer intrinsischen Exploration nach möglichen Stil-Inkonsistenzen untersucht werden. Liegen solche vor, so könnte dies ein Hinweis sein, dass der Text aus

[1] Beispielsweise in Portale wie `Amazon.de`, `Ciao.de` und Ähnliche.

[2] Siehe dazu die Literaturempfehlung [6].

[3] Dies könnte im Rahmen der Recherche in Kapitel 6.2, bzw. Kapitel 6.3 beobachtet werden.

Fragmenten zusamengesetzt wurde, um so z.B. falsche Tatsachen vorzutäuschen. Testamente können beispielsweise auf eine solche Art manipuliert werden, um den Stil des wahren Verfassers bis zu einem gewissen Grad beizubehalten und gleichzeitig falsche Angaben zu Erbschaften vorzugeben.

Da in der Forensik (je nach Szenario) Textlängen sowie die Textanzahl stark variieren können, kommen sehr oft Features aus der Zeichen-Ebene zur Anwendung, da hier der Recall am höchsten ist. Häufig werden z.B. Buchstaben $n-$Gramme, Interpunktionszeichen oder Abkürzungen als Features eingesetzt. So wurde beispielsweise in [78] eine Attributions-Genauigkeit von $97,85\%$ erzielt, indem für kurze Texte Abkürzungen als Features verwendet wurden.

Anmerkung: Aufgrund der Tatsache, dass automatisierte Autorschaftsanalyse-Verfahren nicht in der Lage sind stichhaltige Beweise, sondern nur statistische Aussagen zu erbringen, sollten diese nur als unterstützende Werkzeuge für Forensik-Experten betrachtet werden. Mit anderen Worten: Forensik-Experten sollten durch derartige Werkzeuge unter keinen Umständen gänzlich ersetzt werden.

6.7.4 Attribution von Quelltexte

Die Autorschafts-Attribution ist nicht nur auf natürlichsprachige Texte beschränkt. So lassen sich unter anderem Quelltexte, die in einer Programmiersprache verfasst worden sind, ebenfalls hinsichtlich ihrer Autorschaft zuordnen. Burrows verwendet beispielsweise in [12] sechs verschiedene Feature Klassen, um dadurch Programmierstile (analog zu den Autorenstilen in natürlichsprachigen Texten) zu approximieren. Zu den Feature-Klassen zählen unter anderem reservierte Schlüsselwörter, Operatoren, Funktionbezeichner, oder auch Leerzeichen/Einrückungen, [12, Seite: 124].

Die Attribution von Quelltexte wird nicht nur innerhalb der Lehre und Industrie, sondern auch in der IT-Sicherheitsbranche eingesetzt. Hier wird unter anderem versucht Autorschaften von böswilligem Code (Viren, Würmer, Trojaner, etc.), verdächtigen Tätern zuzuordnen.

6.7.5 Bestimmung der Güte von Natural Language Watermarking Verfahren

Unter dem Begriff des Natural Language Watermarkings (kurz NLW) wird eine Disziplin verstanden, die das Ziel verfolgt, versteckte Wasserzeichennachrichten in natürlichsprachlichen Texten einzubetten, sodass diese beim Lesen nicht wahrgenommen werden, [106]. Die Einbettung einer Wasserzeichennachricht $\mathcal{W}$ erfolgt dabei durch eine Manipulation auf die unterschiedlichsten sprachlichen Ebenen (Morphologie, Syntax, Semantik, etc.), wobei sich die Einbettungsmethoden unter anderem auf Paraphrasierungen (z.B. Umordnung von Wörtern/Konstituenten) oder die Substitution von Wörtern stützen, [106]. Somit resultiert ein Dokument $\mathcal{D}$ nach der Einbettung von $\mathcal{W}$ in Verbindung mit einem geheimen Schlüssel in ein modifiziertes Dokument $\mathcal{D}^{'}$. Das Ziel ist dabei die semantische Gleichheit zu bewahren, sodass stets $\mathcal{D}^{'} \approx \mathcal{D}$ gilt.

Es existieren zahlreiche Einbettungsmethoden, die in der Lage sind $\mathcal{D}$ nach $\mathcal{D}^{'}$ zu überführen, ohne dabei die Semantik zu verletzen. Viele von ihnen weisen jedoch eine Schwäche bzgl. des Stils auf, der durch die Einbettung von $\mathcal{W}$ stets in einem gewissen Grad verzerrt wird. Hier könnte z.B. eine intrinsische Exploration angewendet werden, um so stilistische Ausreißer in $\mathcal{D}^{'}$ zu entdecken. Diese Ausreißer repräsentieren dabei diejenigen Textstellen in $\mathcal{D}$, die durch die Einbettungsmethoden modifiziert worden sind, sodass sich die Güte[1] eines NLW Algorithmus dadurch bestimmen lässt. Werden dagegen keine bzw. nur schwache Ausreißer in $\mathcal{D}^{'}$ gefunden, kann die Einbettung als erfolgreich angesehen werden, da sich $\mathcal{D}^{'}$ neben der Semantik, vom Stil gegenüber $\mathcal{D}$ wenig unterscheidet.

[1] Eine Internet-Recherche in ACM, IEEE, Springer und anderen Portalen ergab für eine solche Gütebestimmung bisher keine wissenschaftliche Beiträge.

6.7.6 Weitere Anwendungsgebiete

Die Autorschaftsanalyse hat viele weitere Einsatzmöglichkeiten. Um den Rahmen des Kapitels nicht zu sprengen, werden nachfolgend weitere drei Anwendungsgebiete kurz aufgezählt:

- **Stil-Optimierung:** Die intrinsische Exploration kann dazu verwendet werden, ein Dokument auf Stilbrüche hin zu untersuchen. Denkbar sind z.B. außergewöhnlich komplexe Sätze oder auch ein unverhältnismäßiger Gebrauch von komplexen Worten, die nur in bestimmten Absätzen vorkommen. Solche grammatischen Eigenarten können von einem Autor aufgegriffen werden, um dessen Text stilistisch zu „glätten“.
- **Stilbasiertes Information Retrieval:** Hier könnten z.B. Suchanfragen ermöglicht werden, um Dokumente anhand stilistischer Ähnlichkeiten zu finden. Herkömmliche Suchmaschinen (wie etwa Google) können bisher nur thematische ähnliche Dokumente aufspüren.
- **Stilbasierte E-Mail Filterung:** Denkbar ist die Erkennung von ausgeklügelten Spam, Phishing oder Scamming-Nachrichten anhand der darin befindlichen Stile, um anschließend diese zu entfernen. Anstelle böswilliger Nachrichten können stattdessen auch relevante Mails anhand ihrer Autoren kategorisiert werden, unabhängig von Mail-Inhalte bzw. assoziierten Metadaten (z.B. Absender, Betreffzeile, Signatur, etc.).

7 Ansätze für die Autorschaftsanalyse

In diesem Kapitel werden Ansätze für die Autorschafts-Attribution, Autorschafts-Verifikation und die intrinsische Exploration vorgestellt. Die ersten drei Unterkapitel erläutern dabei diejenigen Verfahren, die im Rahmen des Frameworks (Kapitel 8) implementiert wurden. Im vierten Unterkapitel werden darüber hinaus theoretische Ansätze vorgeschlagen, die aufgrund der technischen Machbarkeit aktuell noch nicht realisiert werden können.

Bei allen Verfahren handelt es sich um eigene Ansätze, die teilweise auf anderen Verfahren bzw. existierenden Konzepte basieren. Eine Ausnahme bilden hier die instanzbasierten Autorschafts-Attributions-Verfahren im Unterkapitel 7.1.4, die innerhalb dieser Arbeit implementiert, jedoch nicht selbst entworfen wurden. Da diese Verfahren für einige Experimente im Kapitel 10 als Baselines dienen, werden sie der Vollständigkeit halber hier mit aufgeführt.

7.1 Autorschafts-Attributions-Verfahren

In diesem Unterkapitel werden insgesamt sechs Autorschafts-Attributions-Verfahren (kurz *Attributions-Verfahren*) präsentiert. Diese basieren auf existierenden Verfahren bzw. Konzepten, die Stamatatos in seiner Studie über moderne Attributions-Verfahren [99] beschreibt. Die Erweiterungen bzw. Unterschiede zwischen den bestehenden und den eigenen Verfahren werden dabei an den jeweiligen Stellen hervorgehoben.

7.1.1 Einordnung von Attributions-Verfahren

Nach Stamatatos [99, Seiten: 13–18] lassen sich Attributions-Verfahren in profil- und instanzbasierte Ansätze aufteilen. Diese werden wie folgt zusammengefasst:

1.) Profilbasierte Ansätze: Diese Ansätze erfordern, dass sämtliche Dokumente in einer Trainingsmenge zunächst zu einem großen Dokument zusammengeführt werden, welches dabei unterschiedlich repräsentiert werden kann. So kann z.B. die reine Textform[1], die komprimierte Textform oder auch die Byte-Repräsentation eines Textes betrachtet werden. Im nächsten Schritt werden aus dem Dokument Features entnommen, um daraus ein *Autor-Profil* erstellen zu können. Ein Autor-Profil entspricht

[1] Sämtliche (eigene) Verfahren die in dieser Arbeit vorgestellt werden, betrachten neben der reinen Form, zusätzlich die sprachlichen Ebenen der Texte.

dabei (in der Regel) einem Feature-Vektor, der wiederum einen Autorenstil wiederspiegelt. Neben dem Profil wird des Weiteren ein Attributions-Modell benötigt, mit dessen Hilfe die Autorschaft eines anonymen Dokuments attribuiert werden soll. In der Literatur existieren dazu viele verschiedene Modelle, so zählt Stamatatos in [99, Seite: 14] beispielsweise die Folgenden auf:

- Wahrscheinlichkeitsmodelle: Diese Modelle basieren zumeist auf stochastischen Prozessen (z.B. Markow-Ketten) oder auch probabilistische Klassifikatoren (z.B. Naive Bayes), die die Autorschaft des anonymen Dokuments mittels Häufigkeitsverteilungen von Features zu attribuieren versuchen. Die wichtigste Eigenschaft dieser Modelle ist, dass die Features hier eine qualitative Form aufweisen, [99, Seite: 14]. Unter anderem sind hierbei Wörter, Lexeme, Types, Stems oder auch Buchstaben bzw. Token $n-$Gramme als Features zulässig.

- Kompressionsmodelle: Bei diesen Modellen werden die Beispieltexte zunächst als große Dokumente zusammengeführt. Zu jedem großen Dokument wird anschließend das anonyme Dokument hinzugefügt. Das resultierende Dokument wird dann mit Hilfe eines entsprechenden Kompressionsalgorithmus (z.B. RAR, LZW, GZIP, etc.) zu einer Datei komprimiert. Anschließend werden mit Hilfe einer speziellen Metrik Entropieunterschiede zwischen den komprimierten Dateien sowie dem komprimierten großen Dokument gesucht. Die Attribution erfolgt hierbei anhand des kleinsten bitweisen Größenunterschieds zwischen jedem Paar, [99, Seite: 14].

- Häufigste $n-$Gramme Modelle: Diese zählen zu den mächtigsten Attributions-Modellen und basieren auf einfache Metriken mit deren Hilfe Stilähnlichkeiten bzw. Stilabweichungen zwischen dem anonymen Dokument und allen Traingsdokumenten berechnet werden. Letztere werden dabei zuvor zu großen Dokumenten zusammengeführt. Aus diesen großen Dokumenten sowie dem anonymen Dokument, werden anschließend die $k-$häufigsten $n-$Gramme entnommen und als Profile gespeichert. Anhand der extrahierten $n-$Gramme wird im nächsten Schritt ein Stilvergleich zwischen den Profilen und dem anonymen Dokument durchgeführt. Die Attribution erfolgt hierbei anhand des Profils, der die ähnlichste Häufigkeitsverteilung zum anonymen Dokument aufweist.

2.) Instanzbasierte Ansätze: Bei diesen Ansätzen werden anstelle von Profilen, die Dokumente der Autoren einzeln betrachtet. Diese werden dabei stets in eine Feature-Vektor Darstellung überführt, sodass jeder Vektor als eine stilistische Instanz eines dazugehörigen Autors betrachtet werden kann. Analog zu den profilbasierten Verfahren wird auch hier ein Attributions-Modell benötigt, welches in der Regel auf (mindestens) einem Klassifikator beruht. Anhand des Modells wird die Autorschaft des anonymen Dokuments vorhergesagt (bzw. *klassifiziert*). Stamatatos zählt dazu in [99, Seite: 17] die folgenden Modelle auf:

- Vektorraum Modelle: Ausgehend von den Feature-Vektoren wird zunächst ein $n-$dimensionaler Feature-Raum definiert, in dem die Vektoren angeordnet sind. Mit Hilfe eines Klassifikators wird anschließend ein Modell erlernt, dass die Aufgabe hat ein anonymes Dokument (welches auch als Feature-Vektor vorliegt) zu einer passenden Autor-Klasse zuzuordnen. Hierbei können alle möglichen Klassifikatoren eingesetzt werden, wie beispielsweise Naive Bayes, Support Vector Maschine, $k-$Nearest Neighbour, Neuronale Netze, etc. Vereinfacht ausgedrückt: Bei Vektorraum Modellen gleicht die Attribution einer Klassifikation[1].

- Ähnlichkeitsbasierte Modelle: Die Idee dieser Modelle liegt darin, paarweise Ähnlichkeiten zwischen dem anonymen Dokument und allen Trainingsdokumenten (in ihrer Feature-Vektor Darstellung) zu berechnen. Die Attribution bzgl. des stilistisch ähnlichsten Autors erfolgt mit Hilfe eines Nearest Neighbour Klassifikator, [99, Seite: 17].

[1] Siehe in diesem Zusammenhang Kapitel 4.3.

7.1.2 Grundlegende Komponenten

In diesem Abschnitt werden einige grundlegende Komponenten beschrieben, auf die sowohl profil- als auch instanzbasierte Ansätze angewiesen sind:

- Ein anonymes Dokument $\mathcal{D}_{\varepsilon}$, dessen Autorschaft attribuiert werden soll.
- Eine Trainingsmenge $\mathbb{D}_{train}$ die r Trainingsdokumente von insgesamt m Autoren enthält:

$$\mathbb{D}_{train} = \{\, \mathcal{D}_{1\mathcal{A}_1}, \mathcal{D}_{2\mathcal{A}_1}, \ldots, \mathcal{D}_{1\mathcal{A}_2}, \mathcal{D}_{2\mathcal{A}_2}, \ldots, \mathcal{D}_{r-1\mathcal{A}_m}, \mathcal{D}_{r\mathcal{A}_m} \,\}, \text{ mit } r > m \tag{1}$$

  Jedes $\mathcal{D}_{i\mathcal{A}_j}$ repräsentiert das $i-$te Dokument eines zugehörigen $j-$ten Autors $\mathcal{A}_j$, wobei von jedem $\mathcal{A}_j$ eine bestimmte Anzahl ℓ an Dokumenten existiert. Hierbei werden hinsichtlich aller Dokumente die folgenden zwei Annahmen vorausgesetzt:

  1. Jedes $\mathcal{D}_{i\mathcal{A}_j}$ ist von einem Autor verfasst worden (Annahme der eindeutigen Autorschaft).
  2. Der wahre Autor $\mathcal{A}_{true}$ von $\mathcal{D}_{\varepsilon}$ ist innerhalb $\mathbb{D}_{train}$ bekannt (Grundannahme[1]).

- Eine Menge von Features $\mathbb{F} = \{\, f_1, f_2, \ldots \}$ mit deren Hilfe Feature-Vektoren generiert werden können. Ein Feature-Vektor $\mathcal{F}_{i\mathcal{A}_j}$ bzgl. eines Dokuments $\mathcal{D}_{i\mathcal{A}_j}$ hat dabei stets die Form:

$$\mathcal{F}_{i\mathcal{A}_j} = (\, f_1(\mathcal{D}_{i\mathcal{A}_j}), f_2(\mathcal{D}_{i\mathcal{A}_j}), \ldots, f_n(\mathcal{D}_{i\mathcal{A}_j}) \,)$$

  wobei $f_k(\mathcal{D}_{i\mathcal{A}_j})$ die Anwendung eines Features f_k auf das Dokument $\mathcal{D}_{i\mathcal{A}_j}$ beschreibt.

Anhand dieser Komponenten werden in den nächsten Unterkapiteln profil- und instanzbasierte Attributions-Verfahren vorgestellt.

7.1.3 Profilbasierte Attributions-Verfahren

Im Folgenden werden profilbasierte Verfahren, anhand der eingeführten Notation aus Kapitel 2.1 erläutert. Wie eingangs erwähnt, erfordern diese, dass sämtliche Trainingsdokumente aller Autoren, zu Autor-Profilen konkateniert (aneinandergefügt) werden. Dazu werden die r Dokumente zunächst nach ihren m Autoren wie folgt sortiert:

$$\begin{aligned}
\mathbb{D}_{\mathcal{A}_1} &= \{\, \mathcal{D}_{1\mathcal{A}_1}, \mathcal{D}_{2\mathcal{A}_1}, \ldots, \,\} \\
\mathbb{D}_{\mathcal{A}_2} &= \{\, \mathcal{D}_{1\mathcal{A}_2}, \mathcal{D}_{2\mathcal{A}_2}, \ldots, \,\} \\
&\vdots \\
\mathbb{D}_{\mathcal{A}_m} &= \{\, \mathcal{D}_{1\mathcal{A}_m}, \mathcal{D}_{2\mathcal{A}_m}, \ldots, \,\}
\end{aligned}$$

Dadurch ensteht für jeden Autor $\mathcal{A}_j$ eine Dokumentenmenge $\mathbb{D}_{\mathcal{A}j}$, die dessen sämtliche Dokumente enthält. Damit kann $\mathbb{D}_{train}$ wie folgt umgeformt werden:

$$\mathbb{D}_{train} = \mathbb{D}_{\mathcal{A}_1} \cup \mathbb{D}_{\mathcal{A}_2} \cup \ldots \cup \mathbb{D}_{\mathcal{A}_m}$$

Im nächsten Schritt werden sämtliche $\mathcal{D}_{i\mathcal{A}_j} \in \mathbb{D}_{\mathcal{A}j}$ zu einem Autor-Profil $\mathcal{D}_{BIG\mathcal{A}_j}$ konkateniert:

$$\begin{aligned}
\mathcal{D}_{BIG\mathcal{A}_1} &= \mathcal{D}_{1\mathcal{A}_1} \circ \mathcal{D}_{2\mathcal{A}_1} \circ \ldots \circ \mathcal{D}_{\ell_1\mathcal{A}_1} && \text{, mit } \ell_1 = |\mathbb{D}_{\mathcal{A}_1}| \\
\mathcal{D}_{BIG\mathcal{A}_2} &= \mathcal{D}_{1\mathcal{A}_2} \circ \mathcal{D}_{2\mathcal{A}_2} \circ \ldots \circ \mathcal{D}_{\ell_2\mathcal{A}_2} && \text{, mit } \ell_2 = |\mathbb{D}_{\mathcal{A}_2}| \\
&\vdots && \vdots \\
\mathcal{D}_{BIG\mathcal{A}_m} &= \mathcal{D}_{1\mathcal{A}_m} \circ \mathcal{D}_{2\mathcal{A}_m} \circ \ldots \circ \mathcal{D}_{\ell_m\mathcal{A}_m} && \text{, mit } \ell_m = |\mathbb{D}_{\mathcal{A}_m}|
\end{aligned}$$

[1] Siehe dazu Kapitel 6.2.1.

Ausgehend von diesen Profilen ändert sich $\mathbb{D}_{train}$ in:

$$\mathbb{D}'_{train} = \{ \mathcal{D}_{BIG\mathcal{A}_1}, \mathcal{D}_{BIG\mathcal{A}_2}, \ldots, \mathcal{D}_{BIG\mathcal{A}_m} \}$$

Aus dem anonymen Dokument $\mathcal{D}_\varepsilon$ und sämtlichen $\mathcal{D}_{BIG\mathcal{A}_j} \in \mathbb{D}'_{train}$ werden anschließend Feature-Vektoren generiert[1]. Für $\mathcal{D}_\varepsilon$ entsteht dadurch ein Vektor mit der Form $\mathcal{F}_\varepsilon$ und für die $\mathcal{D}_{BIG\mathcal{A}_j}$ die folgende Menge:

$$\mathbb{F}'_{train} = \{ \mathcal{F}_{BIG\mathcal{A}_1}, \mathcal{F}_{BIG\mathcal{A}_1}, \ldots, \mathcal{F}_{BIG\mathcal{A}_m} \}$$

Anmerkung: Die einzelnen Autor-Profile $\mathcal{D}_{BIG\mathcal{A}_j}$ bieten gegenüber einzelnen $\mathcal{D}_{i\mathcal{A}_j}$ den interessanten Vorteil, dass stilistische Varietäten eines Autors in einem einzigen Dokument zusammengefasst werden können. Dadurch kann dieses als eine Art *Stil-Repertoire* verstanden werden, welches zahlreiche Stilmerkmale (z.B. Kollokationen, Anglizismen, Redewendungen oder Ähnliche) enthält, auf die ein Autor mehr oder weniger zurückgreift, sobald dieser einen neuen Text verfasst[2].

7.1.3.1 Pairwise-Similarity Verfahren

Beim ersten profilbasierten Attributions-Verfahren, dass nun vorgestellt wird, handelt es sich um das sogenannte Pairwise-Similarity Verfahren, welches auf eine konzeptuelle Beschreibung von Stamatatos in [99, Seite: 13] basiert. Während Stamatatos' Beschreibung als Modell für thematisch ähnliche Verfahren gedacht war, wurde die Gelegenheit ergriffen, aus diesem Modell ein eigenständiges Verfahren umzusetzen. Im Folgendem wird das Verfahren anhand der Komponenten aus Kapitel 7.1.3 genauer erläutert.

Ausgehend von $\mathcal{F}_\varepsilon$ und $\mathbb{F}'_{train}$, gilt es zunächst eine beliebige Metrik auszuwählen, um dadurch Stilvergleiche zwischen $\mathcal{F}_\varepsilon$ und jedem $\mathcal{F}_{BIG\mathcal{A}_j} \in \mathbb{F}'_{train}$ durchführen zu können. Als gewählte Metrik wird hierbei stellvertreted eine Distanzfunktion[3] $dist(\cdot, \cdot)$ verwendet. Anhand dieser werden Distanzen zwischen $\mathcal{F}_\varepsilon$ und jedem $\mathcal{F}_{BIG\mathcal{A}_j}$ berechnet, die in diesem Kontext als Stilabweichungen interpretiert werden können. Die resultierenden Distanzen werden anschließend in einer Folge gespeichert:

$$Distances = (\, s_1, s_2, \ldots, s_m \,) \quad \text{, mit } s_j = dist(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_j})$$

Im nächsten Schritt wird *Distances* aufsteigend sortiert, was einer Permutation der Folge entspricht:

$$Distances = (\, s_{x_1}, s_{x_2}, \ldots, s_{x_m} \,) \quad \text{, mit } x_i \in \{\, 1, 2, \ldots, m \,\}$$

Hierdurch entsteht ein Problem, dass nun nicht ersichtlich ist, zu welchem Autor-Profil die jeweilige Distanz gehört, da durch die Sortierung, die Indizes umgeordnet werden können. Daher werden die einzelnen s_{x_i} mit einer Kennung assoziiert, die angibt gegen welches $\mathcal{F}_{BIG\mathcal{A}_j}$ (und damit gegen welchen Autor $\mathcal{A}_j$) $\mathcal{F}_\varepsilon$ verglichen wurde. Dadurch ändert sich *Distances* in:

$$Distances' = \Big((s_{x_1}, \mathcal{A}_{x_1}), (s_{x_2}, \mathcal{A}_{x_2}), \ldots, (s_{x_m}, \mathcal{A}_{x_m}) \Big)$$

Die Attribution bzgl. der Autorschaft des anonymen Dokuments erfolgt, indem derjenige Autor $\mathcal{A}_{min}$, dessen Stilabweichung am kleinsten zu dem von ε ist, aus dieser Folge entnommen wird. Da $Distances'$ aufsteigend sortiert ist, stellt $\mathcal{A}_{x_1}$ innerhalb des ersten Tupels, den stilistisch ähnlichsten Autor $\mathcal{A}_{min}$ dar. Damit wäre die Attribution abgeschlossen, die als korrekt betrachtet wird, sofern die folgende Konjunktion gilt:

$$(\, \mathcal{A}_{min} = \varepsilon \,) \wedge (\, \mathcal{A}_{min} = \mathcal{A}_{true} \,)$$

Hierbei drückt $\mathcal{A}_{true}$ den wahren Autoren des anonymen Dokuments $\mathcal{D}_\varepsilon$ aus.

---

[1] Zur Erinnerung: Diese Vorgehensweise wurde bereits in Kapitel 5.1 beschrieben.

[2] Siehe in diesem Zusammenhang die Aussage von Drommel auf der Seite: 23.

[3] Anstelle einer Distanzfunktion kann hier auch eine Ähnlichkeitsfunktion angegeben werden.

7.1.3.2 Fragmentwise Intersection Verfahren

Das Fragmentwise Intersection Verfahren ist ein weiteres profilbasiertes Attributions-Verfahren, welches auf das sogenannte SCAP (*Source Code Author Profile*) Verfahren von Stamatatos et. al, [31] basiert. SCAP stellt selbst eine Weiterentwicklung der CLLM (*Character Level Language Model*) Methode von Keselj et al. [109] dar, die in Kapitel 6.6.1 kurz erwähnt wurde. CLLM wurde ursprünglich konzipiert, um eine sprachunabhängige Autorschafts-Attribution zu ermöglichen. Realisiert wurde dies mit $n-$Gramme Profilen, die aus der sprachneutralen Byte-Repräsentation der Texte entnommen wurden. Im Jahre 2007 wurde CLLM von Stamatatos et. al [31] weiterentwickelt, um neben natürlichen Sprachen auch Quelltexte hinsichtlich ihrer Autoren (bzw. Programmierer) attribuieren zu können. Das SCAP Verfahren wird im Folgenden kurz zusammengefasst.

Ausgehend von einer Menge von Quelltexten gilt es zunächst eine Aufteilung in einer Trainings- und Testmenge vorzunehmen. Aus der Trainingsmenge werden anschließend sämtliche darin befindliche Quelltexte (eines jeden Autors) zu einem großen Dokument konkateniert. Im nächsten Schritt werden aus diesem Dokument, Features in Form von $n-$Gramme entnommen. Die Features können dabei aus Buchstaben, Leerzeichen, Symbolen oder auch Steuerzeichen (nicht darstellbare Zeichen) bestehen. Zu jedem Dokument wird anschließend ein Trainingsprofil generiert, welches aus einer absteigend sortierten Liste der $k-$häufigsten $n-$Gramme besteht.

Die Attribution erfolgt hier anhand einer einfachen Schnittbildung $|SP_A \cap SP_B|$, wobei SP_A dem Trainingsprofil und SP_B dem Testprofil entspricht. Alle Profile sind dabei hinsichtlich ihrer Größe einheitlich durch das k beschränkt. Das Resultat der Attribution ergibt dasjenige Trainingsprofil, dessen Schnitt mit SP_B die meisten gemeinsamen Elementen aufweist. Im Folgenden wird das Fragmentwise Intersection Verfahren beschrieben, bevor im Anschluß die Unterschiede zu SCAP hervorgehoben werden.

Ausgehend von $\mathcal{D}_\varepsilon$ und $\mathbb{D}'_{train}$ werden zunächst aus sämtlichen Dokumenten die $k-$häufigsten Textfragmente entnommen. Bei diesen kann es sich z.B. um Tokens, Wörter, Präfixe/Suffixe, $n-$Gramme, Wortarten, Kollokationen, Konstituenten und ähnliche Textfragmente handeln. Der nächste Schritt besteht darin, die entnommenen Textfragmente in entsprechenden Mengen M_ε bzw. $M_{\mathcal{A}_j}$ zu speichern. Anschließend werden für alle Mengenpaare $(M_\varepsilon, M_{\mathcal{A}_1}), (M_\varepsilon, M_{\mathcal{A}_2}), \ldots, (M_\varepsilon, M_{\mathcal{A}_m})$, Ähnlichkeitswerte mit Hilfe einer mengenbasierten Ähnlichkeitsfunktion[1] berechnet und in einer Folge *Similarities* gespeichert:

$$Similarities = (\, s_1,\, s_2,\, \ldots,\, s_m\,) \quad \text{mit } s_j = sim(M_\varepsilon, M_{\mathcal{A}_j})$$

Im nächsten Schritt wird *Similarities* absteigend sortiert und mit einer entsprechenden Autoren-Kennung assoziiert, sodass dadurch eine geänderte Folge entsteht:

$$Similarities' = \Big(\, (s_{x_1}, \mathcal{A}_{x_1}),\, (s_{x_2}, \mathcal{A}_{x_2}),\, \ldots,\, (s_{x_m}, \mathcal{A}_{x_m})\,\Big)$$

Die Attribution bzgl. der Autorschaft von $\mathcal{D}_\varepsilon$ erfolgt nach dem gleichen Prinzip, wie beim Pairwise-Similarity Verfahren. Hier wird jedoch nicht der Autor mit der kleinsten Stilabweichung, sondern derjenige mit dem größten Ähnlichkeitswert in $Similarities'$ gesucht. Da $Similarities'$ absteigend sortiert ist, lautet der gesuchte Autor $\mathcal{A}_{x_1}$. Dieser stellt den zu ε stilistisch ähnlichsten Autor $\mathcal{A}_{max}$ dar. Damit wäre die Attribution abgeschlossen, die hier als korrekt betrachtet wird, sofern die folgende Bedingung gilt:

$$(\, \mathcal{A}_{max} = \varepsilon\,) \wedge (\, \mathcal{A}_{max} = \mathcal{A}_{true}\,)$$

Im Folgenden werden nun die Unterschiede zwischen dem Fragmentwise Intersection Verfahren und SCAP aufgelistet:

- Als Features können beim Fragmentwise Intersection Verfahren neben $n-$Gramme, beliebige Textfragmente (siehe oben) verwendet werden. Die Features werden dabei nicht aus der Byte-Repräsentierung, sondern stattdessen aus den unterschiedlichen sprachlichen Ebenen (wie etwa

[1] Einige solcher Ähnlichkeitsfunktionen wurden im Kapitel 4.2.4 vorgestellt.

Morphologie, Syntax oder Semantik) der Texte gewonnen. Die Überführung der Texte in die sprachlichen Ebenen, wird mit Hilfe der NLP-Werkzeuge aus Kapitel 3.2 realisiert.

- Werden nur $n-$Gramme als Features verwendet, so muss n nicht wie beim SCAP fest angegeben werden. Das bedeutet, dass hier mehrere $n-$Gramm Größen kombiniert werden können, z.B. $\{\texttt{Unigramme}\} \cup \{\texttt{Bigramme}\} \cup \{\texttt{Trigramme}\} \cup \ldots$
- Anstelle einer einfachen Schnittbildung zwischen den Trainings- und Testprofilen in SCAP, können beim Fragmentwise Intersection Verfahren beliebige (mengenbasierte) Ähnlichkeitsfunktionen verwendet werden. Vergleiche können damit besser gehandhabt werden, da die Resultate der Ähnlichkeitsfunktionen intervallskaliert sind.

Anmerkung: Das Fragmentwise Intersection Verfahren hat gegenüber dem Pairwise-Similarity Verfahren einige Vorteile, die an dieser Stelle erwähnt werden sollten:

- **Qualitative Features:** Sämtliche Features, die in dieser Methode verwendet werden, stellen Textfragmente dar und sind damit qualitativ. Dies birgt den Vorteil, dass dadurch eine Transparenz hinsichtlich der Features gegeben ist, die eine bessere Interpretation ermöglicht, welche Features genau für eine erfolgreiche Attribution ausschlaggebend waren. Dies ist beim Pairwise-Similarity Verfahren, aufgrund der quantitativen (bzw. numerischen) Features nicht der Fall. Hier muss stattdessen ein Mehraufwand (z.B. mit Hilfe von Feature-Selektionsalgorithmen) betrieben werden, um feststellen zu können, welche Features für ein erfolgreich durchgeführtes Attributions-Szenario relevant waren.
- **Laufzeit:** Neben der Transparenz von Features ist die Laufzeit ein weiterer wichtiger Vorteil des Fragmentwise Intersection Verfahrens. Da hier keine Feature-Vektoren aus den Dokumenten generiert werden, entfällt unter anderem die aufwändige Suche von wörterbuchbasierten Features innerhalb der Dokumente. Die Komplexität des Verfahrens ist im Wesentlichen auf die paarweise (mengenbasierte) Ähnlichkeitsberechnung, anhand der $k-$häufigsten (gemeinsamen) Textfragmenten beschränkt. Hinzu kommt noch die Tatsache, dass k in der Regel deutlich kleiner ist als die Anzahl der Features, die beim Pairwise-Similarity Verfahren verwendet werden, was sich wiederum positiv auf die Laufzeit des Fragmentwise Intersection Verfahrens auswirkt.
- **Dokumentbasierte Features:** Sämtliche Features die beim Fragmentwise Intersection Verfahren zur Anwendung kommen, werden direkt aus $\mathcal{D}_\varepsilon$ und sämtlichen $\mathcal{D}_{BIG_{\mathcal{A}_1}} \in \mathbb{D}'_{train}$ bzw. deren sprachlichen Ebenen entnommen. Beim Pairwise-Similarity Verfahren werden dagegen allgemeine Features verwendet, die (je nach eingesetzter Feature-Kategorie) nicht zwingend in den Dokumenten vorkommen. Damit kann beim Letzteren der Fall eintreten, dass die Feature-Vektoren überwiegend aus Nullelementen bestehen und dadurch keine Differenzierung der Autorenstile ermöglicht wird.

7.1.3.3 Two-Stage Pairwise-Similarity Verfahren

Das Two-Stage Pairwise-Similarity Verfahren stellt eine Kombination der beiden Verfahren aus Kapitel 7.1.3.2 und Kapitel 7.1.3.1 dar. Ausgehend von $\mathcal{D}_\varepsilon$ und $\mathbb{D}'_{train}$ werden die Verfahren in zwei Stufen eingeteilt, deren Vorgehensweise im Folgenden erläutert wird:

- **Erste Stufe:** In der ersten Stufe des Verfahrens wird zunächst das Fragmentwise Intersection Verfahren auf $\mathcal{D}_\varepsilon$ und sämtlichen $\mathcal{D}_{BIG_{\mathcal{A}_j}} \in \mathbb{D}'_{train}$ angewendet. Das Resultat dieser Anwendung ergibt die gleiche Folge, die in Kapitel 7.1.3.2 aufgeführt ist:

$$Similarities' = \Big((s_{x_1}, \mathcal{A}_{x_1}),\, (s_{x_2}, \mathcal{A}_{x_2}),\, \ldots,\, (s_{x_m}, \mathcal{A}_{x_m}) \Big)$$

  Wichtig hierbei ist die Indexvariable m, die die Anzahl aller Autoren innerhalb $\mathbb{D}'_{train}$ darstellt. Im nächsten Schritt wird $Similarities'$ anhand des Parameters `SPLIT` in zwei Teilfolgen

Sub_1, Sub_2 aufgeteilt, welcher dabei die prozentuale Größe von Sub_1 angibt. Wichtig für das Verfahren ist nur Sub_1, sodass Sub_2, welches nicht repräsentative Autoren enthält, verworfen wird. Aus Sub_1 werden anschließend die Namen der Autoren entnommen, die potentielle Autoren von $\mathcal{D}_\varepsilon$ darstellen und in der folgenden Menge gespeichert:

$$Candidates = \{\, \mathcal{A}_{x_1}, \mathcal{A}_{x_2}, \ldots, \mathcal{A}_{x_k} \}\ \text{, mit } x_i \in \{\, 1, 2, \ldots, m \,\} \text{ und } k = \left\lceil \frac{\texttt{SPLIT} \cdot m}{100} \right\rceil$$

Im letzten Schritt der ersten Stufe werden sämtliche $\mathcal{A}_{x_i} \in Candidates$ mit ihren Autor-Profilen assoziiert. Das Ergebniss der ersten Stufe ergibt die folgende (gefilterte) Trainingsmenge:

$$\mathbb{D}''_{train} = \{\, \mathcal{D}_{BIG\mathcal{A}_{x_1}}, \mathcal{D}_{BIG\mathcal{A}_{x_2}}, \ldots, \mathcal{D}_{BIG\mathcal{A}_{x_k}} \,\}$$

- **Zweite Stufe:** In der zweiten Stufe wird das Pairwise Similarity Verfahren aus Kapitel 7.1.3.1 auf $\mathcal{D}_\varepsilon$ und die gefilterte Trainingsmenge $\mathbb{D}''_{train}$ angewendet. Das Resultat der Anwendung ergibt eine Folge, die die Distanzen (bzw. Stilabweichungen) zwischen $\mathcal{D}_\varepsilon$ und sämtlichen $\mathcal{D}_{BIG\mathcal{A}_{x_i}} \in \mathbb{D}''_{train}$ wiederspiegelt:

$$Distances' = \Big((s_{x_1}, \mathcal{A}_{x_1}), (s_{x_2}, \mathcal{A}_{x_2}), \ldots, (s_{x_m}, \mathcal{A}_{x_m}) \Big)$$

Die Folge ist dabei anhand der Distanzen aufsteigend sortiert, sodass s_{x_1} die niedrigste Stilabweichung zwischen dem anonymen Dokument und dem Autor-Profil $\mathcal{D}_{BIG\mathcal{A}_{x_1}}$ aufweist. Dies ist gleichbedeutend damit, dass $\mathcal{A}_{x_1}$ den zu ε stilistisch ähnlichsten Autor $\mathcal{A}_{min}$ darstellt. Die abschließende Attribution erfolgt analog zum Pairwise Similarity Verfahren, wobei auch hier diese als korrekt betrachtet werden kann, sofern die folgende Bedingung gilt:

$$(\, \mathcal{A}_{min} = \varepsilon \,) \wedge (\, \mathcal{A}_{min} = \mathcal{A}_{true} \,)$$

Beobachtung: Das Two-Stage Pairwise-Similarity Verfahren birgt einige interessante Eigenschaften, die hauptsächlich vom Parameter `SPLIT` abhängen. Je nachdem wie dieser gewählt wird, verkleinert sich dementsprechend die neue Trainingsmenge. Für `SPLIT` $= 50\%$ halbiert sich beispielsweise $\mathbb{D}'_{train}$ hinsichtlich der Autorenanzahl und damit auch annähernd die Laufzeit in der zweiten Stufe, da hier weniger Feature-Vektoren generiert werden müssen und daher auch weniger Stilvergleiche durchzuführen sind. Damit eignet sich das Verfahren insbesondere für Trainingsmengen mit einer größeren Autorenanzahl. Neben der verminderten Laufzeit steigert sich zudem die Erkennungsrate des Pairwise-Similarity Verfahrens in der zweiten Stufe, da hier aufgrund der kleineren Anzahl an Autoren, weniger Fehlentscheidungen möglich sind. Allerdings hängt dies zum einen davon ab, ob die gefilterte Trainingsmenge in der ersten Stufe tatsächlich potentielle Autoren enthält und zum anderen, ob in der zweiten Stufe diskriminierende Features für die Attribution verwendet werden.

In den späteren Experimenten (Kapitel 10) wird sich zeigen, dass dieses Verfahren im Durchschnitt allen Attributions-Verfahren in dieser Arbeit überlegen ist, sofern eine geeignete Parametrisierung vorgenommen wird.

7.1.4 Instanzbasierte Attributions-Verfahren

In diesem Abschnitt werden die instanzbasierten Attributions-Verfahren erläutert. Wie eingangs erwähnt, handelt es sich hier nicht um eigene Verfahren, sondern stattdessen um die Klassifikatoren $k-$Nearest Neighbor ($k-$NN), Naive Bayes (NB) sowie die Support Vector Machine (SVM), die bereits in Kapitel 4.3 ausführlich beschrieben wurden. Im Folgenden wird daher nicht die Funktionsweise der Klassifikatoren erläutert, sondern, wie diese auf Attributions-Szenarien angewendet werden können.

Ausgehend von $\mathbb{D}_{train}$ gilt es zunächst eine Feature-Matrix F_{matrix} zu generieren, um dadurch einen gewählten Klassifikator trainieren zu können. Der Aufbau[1] von F_{matrix} ist dabei wie folgt gegeben:

[1] Der genauere Aufbau kann in Kapitel 2.1 betrachtet werden.

- **Zeilen:** Jede Zeile von F_{matrix} repräsentiert ein Feature-Vektor $\mathcal{F}_{i\mathcal{A}_j}$, der durch die Anwendung von n Features aus einem Beispiel-Dokument $\mathcal{D}_{i\mathcal{A}_j}$ entsteht. Es gilt:

$$\mathcal{F}_{i\mathcal{A}_j} = (\, f_1(\mathcal{D}_{i\mathcal{A}_j}),\, f_2(\mathcal{D}_{i\mathcal{A}_j}),\, \ldots,\, f_n(\mathcal{D}_{i\mathcal{A}_j})\,)$$

- **Spalten:** Jede Spalte von F_{matrix} repräsentiert einen Feature-Instanz-Vektor $\mathcal{I}(f_k)$, welcher durch die Anwendung eines Features f_k auf sämtliche $\mathcal{D}_{i\mathcal{A}_j} \in \mathbb{D}_{train}$ entsteht. Es gilt:

$$\mathcal{I}(f_k) = (\, f_1(\mathcal{D}_{\mathcal{A}1}),\, f_1(\mathcal{D}_{\mathcal{A}2}),\, \ldots,\, f_1(\mathcal{D}_{\mathcal{A}m})\,)$$

Somit entspricht F_{matrix} der Trainingsmenge $\mathbb{D}_{train}$ in ihrer Feature-Vektor (bzw. Feature-Instanz-Vektor) Darstellung. Im nächsten Schritt wird der Klassifikator trainiert, um ein Vorhersage-Modell $\mathcal{M}$ zu erlernen, der dazu dient das anonyme Dokument $\mathcal{D}_\varepsilon$ zu attribuieren (bzw. klassifizieren).

Hinweis: Falls es sich bei dem gewählten Klassifikator um $k-$NN handelt, so wird der Schritt übersprungen, da dieser kein Modell erzeugt (siehe dazu die Anmerkungen im Kapitel 4.3.3).

Nachdem der Klassifikator anhand von F_{matrix} ein entsprechendes $\mathcal{M}$ generiert hat, kann anschließend das anonyme Dokument $\mathcal{D}_\varepsilon$ klassifiziert werden. Dieses muss jedoch zuvor in dessen Feature-Vektor Darstellung $\mathcal{F}_\varepsilon$ überführt werden. Die folgende Abbildung fasst schematisch die einzelnen Schritte des instanzbasierten Ansatzes zusammen:

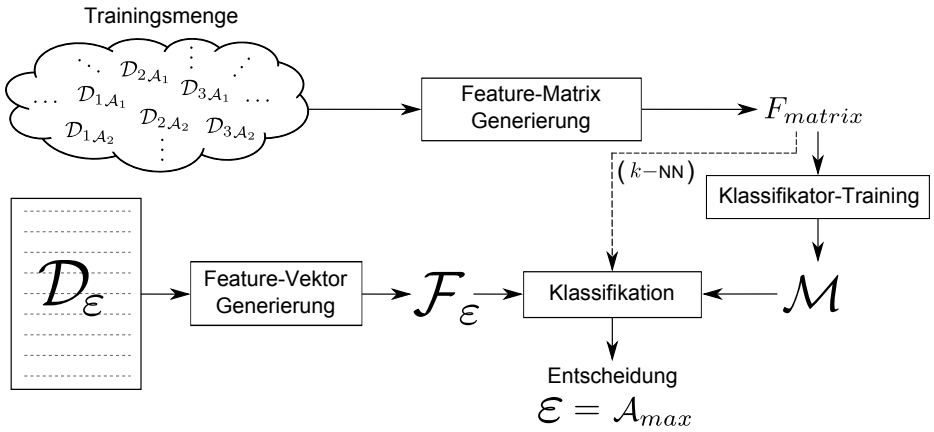


Abbildung 23: Abstraktes Modell eines instanzbasierten Attributions-Verfahrens.

Anmerkung: Um ein verlässliches Modell $\mathcal{M}$ zu erhalten, empfiehlt es sich das Training mehrfach durchzuführen und die Klassifikationsergebnisse entsprechend zu mitteln. Hierfür wurden im Kapitel 4.5.2 jeweils unterschiedliche Strategien beschrieben. Für den Fall, dass für einen Autor nur ein einzelner langer Text existiert, rät Stamatatos in [99] diesen in mehrere (möglichst gleich große) Segmente aufzuteilen. Ähnlich sieht es aus, falls mehrere Trainings-Beispiele existieren, welche jedoch variable Textlängen aufweisen. Hier bietet es sich an, die Beispieltexte zunächst zu einem großen Dokument zusammenzuführen, um diesen anschließend in gleich große Segmente zu unterteilen. Die resultierenden Segmente bzw. Dokumente können als stilistische Instanzen (oder auch als Stil-Beispiele) eines Autors verstanden werden. Daher auch die Bezeichnung *instanzbasierte Attribution.*

7.2 Ansätze der Autorschafts-Verifikation

In diesem Unterkapitel werden zwei Ansätze für die Autorschafts-Verifikation vorgestellt. Das erste Verfahren basiert auf Erläuterungen aus einer Studie über Ein-Klassen-Klassifikation. Beim zweiten Verfahren handelt es sich dagegen um einen eigenen Ansatz, welcher zweistufig aufgebaut ist und den Einsatz von ML- als auch Attributions-Verfahren erfordert, die in vorherigen Kapiteln erwähnt wurden. Im nächsten Unterkapitel werden diejenigen Komponenten aufgelistet, die beide Verfahren benötigen.

7.2.1 Grundlegende Komponenten

Beide Verifikations-Verfahren, die in den nächsten Unterkapiteln präsentiert werden, setzen die folgenden Komponenten voraus:

- Ein Dokument (eines nicht verifizierten Autors), welches durch $\mathcal{D}_{\mathcal{A}_{mask}}$ symbolisiert wird.
- Eine Trainingsmenge mit der Form: $\mathbb{D}_{train} = \mathbb{D}_{\mathcal{A}j} = \{ \mathcal{D}_{1\mathcal{A}_j}, \mathcal{D}_{2\mathcal{A}_j}, \ldots, \mathcal{D}_{m\mathcal{A}_j} \}$.
- Eine Feature-Vektor Darstellung von $\mathcal{D}_{\mathcal{A}_{mask}}$, die durch $\mathcal{F}_{\mathcal{A}_{mask}}$ symbolisiert wird.
- Eine Menge von Feature-Vektoren, die die Trainingsmenge $\mathbb{D}_{train}$ repräsentiert und die Form $\mathbb{F}_{train} = \{ \mathcal{F}_{1\mathcal{A}_j}, \mathcal{F}_{2\mathcal{A}_j}, \ldots, \mathcal{F}_{m\mathcal{A}_j} \}$ aufweist.

7.2.2 Local Density Verfahren

Bei diesem Ansatz handelt es sich um ein einfaches Verifikations-Verfahren, welches auf einem mathematischen Konzept basiert, dass Tax in [104] beschreibt. Im Folgenden wird zunächst das generische Modell von Tax in Kontext der Autorschafts-Verifikation erläutert, bevor im Anschluß einige kleine Modifikationen genannt werden, die daran vorgenommen wurden.

Tax beschreibt in [104, Seite: 69] ein Ein-Klassen-Klassifikationsverfahren[1], dessen Kern ein $k-$NN Klassifikator darstellt und die Bezeichnung *Nearest Neighbor Method* trägt. Ausgehend von $\mathbb{F}_{train}$ wird zunächst ein $n-$dimensionaler Feature-Raum betrachtet, in dem sämtliche $\mathcal{F}_{i\mathcal{A}_j} \in \mathbb{F}_{train}$ angeordnet sind und zusammen eine Zelle[2] $\mathcal{C}_{\mathcal{A}_j}$ darstellen. Diese gibt an, dass jeder Feature-Vektor, der sich darin befindet zu der Klasse $\mathcal{A}_j$ (und damit zum Autor) angehört. Feature-Vektoren die außerhalb dieser Zelle liegen und dazu noch weit von ihr entfernt sind, gehören dagegen einer anderen Klasse an (die durch $\neg\mathcal{A}_j$ gekenntzeichnet wird). Die grundlegende Frage, die hier zu beantworten ist lautet, ob das zu verifizierende Dokument (welches durch $\mathcal{F}_{\mathcal{A}_{mask}}$ repräsentieret wird) zu $\mathcal{C}_{\mathcal{A}_j}$ gehört oder nicht. Dazu wird eine Grenze um $\mathcal{F}_{\mathcal{A}_{mask}}$ gelegt, sodass auch hier eine Zelle entsteht, die Tax in seiner Beschreibung als lokale Dichte bezeichnet und wie folgt definiert:

$$density(\mathcal{F}_{\mathcal{A}_{mask}}) = \frac{k}{m \cdot vol(\|\mathcal{F}_{\mathcal{A}_{mask}} - kNN(\mathcal{F}_{\mathcal{A}_{mask}})\|)}$$

Hierbei repräsentiert $kNN(\mathcal{F}_{\mathcal{A}_{mask}})$ den $k-$nächsten Nachbarn von $\mathcal{F}_{\mathcal{A}_{mask}}$ und $vol(\cdot)$ das Volumen des Bereichs, indem dieser liegt. Um feststellen zu können, ob $\mathcal{F}_{\mathcal{A}_{mask}}$ zu $\mathcal{C}_{\mathcal{A}_j}$ stilistisch „passt“ (was gleichbedeutend ist, dass es sich bei $\mathcal{A}_{mask}$ um $\mathcal{A}_j$ handeln könnte), wird die Zelle in der $\mathcal{F}_{\mathcal{A}_{mask}}$ liegt, solange vergrößert bis damit die $k-$nächsten Nachbarn von $\mathcal{F}_{\mathcal{A}_{mask}}$ eingefangen werden, die sich wiederum in $\mathcal{C}_{\mathcal{A}_j}$ befinden.

Wenn es sich bei $\mathcal{A}_{mask}$ um $\mathcal{A}_j$ handelt, so gilt die Annahme das $\mathcal{F}_{\mathcal{A}_{mask}}$ eine signifikante Ähnlichkeit zu den Feature-Fektoren innerhalb $\mathcal{C}_{\mathcal{A}_j}$ aufweisen muss. Die Ähnlichkeit wird dabei durch einen Abstand zwischen $\mathcal{F}_{\mathcal{A}_{mask}}$ und dessen nächsten Nachbarn $kNN(\mathcal{F}_{\mathcal{A}_{mask}})$ gemessen. Um zu prüfen, ob hier tatsächlich eine signifikante Ähnlichkeit besteht und $\mathcal{F}_{\mathcal{A}_{mask}}$ damit $\mathcal{C}_{\mathcal{A}_j}$ zugeordnet werden kann, muss die lokale Dichte von $\mathcal{F}_{\mathcal{A}_{mask}}$ größer oder gleich der Dichte von seinem (ersten) nächsten Nachbarn in $\mathcal{C}_{\mathcal{A}_j}$ sein. Laut Tax [104, Seiten: 69–71] ist die signifikante Ähnlichkeit (für $k = 1$) gegeben durch:

$$\frac{1}{m \cdot vol(\|\mathcal{F}_{\mathcal{A}_{mask}} - kNN(\mathcal{F}_{\mathcal{A}_{mask}})\|)} \geq \frac{1}{m \cdot vol(\|kNN(\mathcal{F}_{\mathcal{A}_{mask}}) - kNN(kNN(\mathcal{F}_{\mathcal{A}_{mask}}))\|)} \quad (2)$$

Tax wählt hier bewusst die Anzahl der $k-$nächsten Nachbarn mit $k = 1$, da dadurch nur der Abstand zwischen $\mathcal{F}_{\mathcal{A}_{mask}}$ und $\mathcal{C}_{\mathcal{A}_j}$ betimmt wird, anstatt die Abstände zu allen $\mathcal{F}_{i\mathcal{A}_j}$ innerhalb $\mathcal{C}_{\mathcal{A}_j}$ explizit

[1] Zur Erinnerung: Im ML-Kontext stellt die Autorschafts-Verifikation eine Ein-Klassen-Klassifikation dar.

[2] Mathematisch gesprochen handelt es sich hier konkret um eine *Sphäre* im $n-$dimensionalen Raum.

berechnen zu müssen. Eine Umformung von (2) ergibt die folgende vereinfachte Ungleichung:

$$\frac{vol(\|\mathcal{F}_{\mathcal{A}_{mask}} - kNN(\mathcal{F}_{\mathcal{A}_{mask}})\|)}{vol(\|kNN(\mathcal{F}_{\mathcal{A}_{mask}}) - kNN(kNN(\mathcal{F}_{\mathcal{A}_{mask}}))\|)} \leq 1 \tag{3}$$

Um die Volumina in (3) elliminieren zu können, wird die (etwas komplexe) Formel (3.4) aus [104, Seite: 59] angewendet, sodass nach weiteren Umformungen schließlich die folgende Ungleichung resultiert:

$$\frac{\|\mathcal{F}_{\mathcal{A}_{mask}} - kNN(\mathcal{F}_{\mathcal{A}_{mask}})\|}{\|kNN(\mathcal{F}_{\mathcal{A}_{mask}}) - kNN(kNN(\mathcal{F}_{\mathcal{A}_{mask}}))\|} \leq 1$$

Die Interpretation dieser Ungleichung entspricht dem Vergleich zwischen der Distanz von $\mathcal{F}_{\mathcal{A}_{mask}}$ und dessen nächsten Nachbarn $kNN(\mathcal{F}_{\mathcal{A}_{mask}})$ sowie der Distanz von $kNN(\mathcal{F}_{\mathcal{A}_{mask}})$ und dessen nächsten Nachbarn $kNN(kNN(\mathcal{F}_{\mathcal{A}_{mask}}))$ innerhalb $\mathcal{C}_{\mathcal{A}_j}$. Damit ist die Beschreibung des ursprünglichen Modells von Tax abgeschlossen.

Das Modell wird nun modifiziert, indem die folgenden Parameter darin frei eingestellt werden:

- Da die euklidische Norm $\|\,.\,\|$ im ursprünglichen Modell einen Abstand bzw. eine Distanz wiederspiegelt, wird diese im modifizierten Modell durch eine frei wählbare Distanzfunktion $dist(\,\cdot\,,\,\cdot\,)$ substituiert. So kann z.B. $\|\mathcal{F}_{\mathcal{A}_{mask}} - kNN(\mathcal{F}_{\mathcal{A}_{mask}})\|$ zu $d_1 = dist(\mathcal{F}_{\mathcal{A}_{mask}}, kNN(\mathcal{F}_{\mathcal{A}_{mask}}))$ und analog dazu $\|kNN(\mathcal{F}_{\mathcal{A}_{mask}}) - kNN(kNN(\mathcal{F}_{\mathcal{A}_{mask}}))\|$ zu d_2 verallgemeinert werden.

- Im oberen Modell wurde $k = 1$ gewählt, sodass der Abstand zwischen $\mathcal{F}_{\mathcal{A}_{mask}}$ und $kNN(\mathcal{F}_{\mathcal{A}_{mask}})$ im Verhältnis zum Abstand zwischen $kNN(\mathcal{F}_{\mathcal{A}_{mask}})$ und dessen (ersten) direkten Nachbarn $kNN(kNN(\mathcal{F}_{\mathcal{A}_{mask}}))$ gemessen wurde. Im modifizierten Modell kann k frei angegeben werden, solange $k \leq m$ gilt. Dadurch ändert sich die Berechnung des Nenners, indem ein Durchschnittsvektor K_{avg} aus den $k-$nächsten Nachbarn gebildet wird, sodass der Nenner anhand von $\|kNN(\mathcal{F}_{\mathcal{A}_{mask}}) - K_{avg}\|$ bzw. $dist(\mathcal{F}_{\mathcal{A}_{mask}}, K_{avg})$ berechnet werden kann. Die folgende Abbildung illustriert die Idee für unterschiedliche k's:

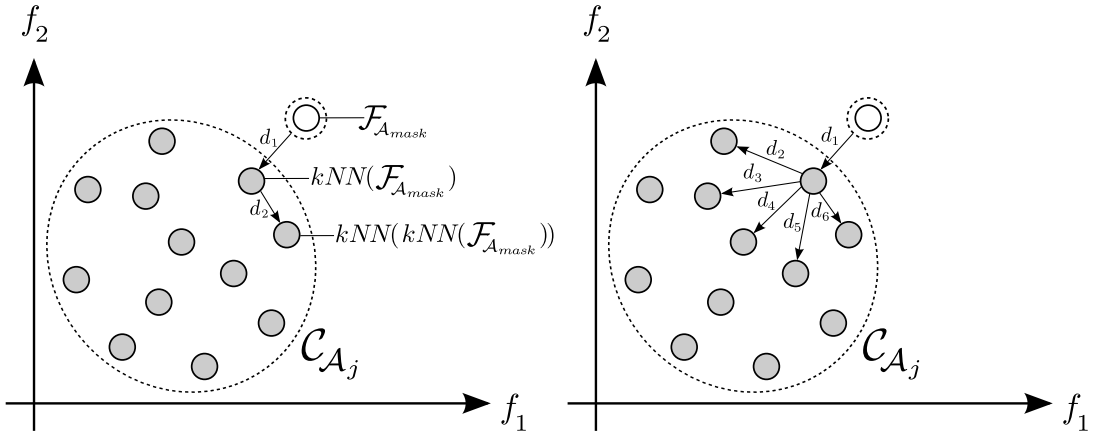


Abbildung 24: Parametereinstellung: $k = 1$ (links) und $k = 5$ (rechts).

- Durch die Umformung in (3) entstand ein fester Schwellwert θ mit dem Wert 1. Dieser kann im modifizierten Modell ebenfalls frei gewählt bzw. dynamisch aus den $\mathcal{F}_{i\mathcal{A}_j}$ innerhalb $\mathcal{C}_{\mathcal{A}_j}$ erlernt werden.

Das modifizierte Modell lautet damit:

$$\frac{d_1}{d_2} \leq \theta \Rightarrow (\,\mathcal{A}_{mask} = \mathcal{A}_j\,) \quad \text{bzw.} \quad \frac{d_1}{d_2} > \theta \Rightarrow (\,\mathcal{A}_{mask} \neq \mathcal{A}_j\,)$$

7.2.3 Cluster-Based Maximum Similarity

Bei diesem Ansatz handelt es sich um die Zusammenführung eines partitionierenden Clustering-Verfahrens (im konkreten Fall $k-$Means) und des Fragmentwise Intersection Verfahrens aus Kapitel 7.1.3.2. Die Idee hierbei beruht darauf, unterschiedliche Formen von Stilähnlichkeiten miteinander zu kombinieren. Ausgehend von $\mathcal{D}_{\mathcal{A}_{mask}}$, $\mathbb{D}_{train}$, $\mathcal{F}_{\mathcal{A}_{mask}}$ und $\mathbb{F}_{train}$ wird die genaue Arbeitsweise des Verfahrens (bzw. Algorithmus) nachfolgend in Form von Pseudocode beschrieben.

1. Definiere zunächst eine Menge $M = \mathbb{D}_{train} \times \mathbb{F}_{train}$, die die folgende Form aufweist:

$$M = \{\, (\mathcal{D}_{1\mathcal{A}_j}, \mathcal{F}_{1\mathcal{A}_j}),\, (\mathcal{D}_{2\mathcal{A}_j}, \mathcal{F}_{2\mathcal{A}_j}),\, \ldots,\, (\mathcal{D}_{m\mathcal{A}_j}, \mathcal{F}_{m\mathcal{A}_j}) \,\}$$

   Füge anschließend das zu verifizierende Dokument dieser Menge hinzu $M \cup \{\, (\mathcal{D}_{\mathcal{A}_{mask}},\, \mathcal{F}_{\mathcal{A}_{mask}}) \,\}$ und bezeichne die resultierende Menge als M'.

2. Partitioniere M' mit Hilfe des $k-$Means Verfahrens (wobei als Input nur die Feature-Vektoren $\mathcal{F}_{\mathcal{A}_{mask}}$ und sämtliche $\mathcal{F}_{i\mathcal{A}_j} \in \mathbb{F}_{train}$ verwendet werden), sodass dadurch eine Aufteilung von M' entsteht, die eine Menge von disjunkten Clustern darstellt:

$$\mathbb{C} = \{\, \mathcal{C}_1,\, \mathcal{C}_2,\, \ldots,\, \mathcal{C}_k \,\},\ \text{mit}\ \ k \leq |\mathbb{F}_{train}|$$

3. Überprüfe für jeden $\mathcal{C}_i \in \mathbb{C}$ ob hier Cluster mit $|\mathcal{C}_i| = 1$ vorhanden sind. Terminiere, falls ein Cluster mit der Form $\mathcal{C}_i = \{\, \mathcal{F}_{\mathcal{A}_{mask}} \,\}$ präsent ist. Liefere dabei die Entscheidung $\mathcal{A}_{mask} \neq \mathcal{A}_j$. Überspringe diesen Schritt, falls für alle Cluster $|\mathcal{C}_i| > 1$ gilt.

4. Entferne $\mathcal{F}_{\mathcal{A}_{mask}}$ aus demjenigen Cluster $\mathcal{C}_x$ zu dem dieser zugeordnet wurde und speichere dessen Kennung x in die Variable `Cluster-Mask`.

5. Erstelle für jeden Cluster $\mathcal{C}_i$ eine Menge $\mathcal{M}_i$ mit denjenigen Dokumenten $\mathcal{D}_{i\mathcal{A}_j}$, deren ID's zu den Feature-Vektoren innerhalb der $\mathcal{C}_i$ korrespondieren. Beispiel: Für $\mathcal{C}_1 = \{\, \mathcal{F}_{3\mathcal{A}_j},\, \mathcal{F}_{6\mathcal{A}_j},\, \mathcal{F}_{10\mathcal{A}_j} \,\}$ wird die Menge $\mathcal{M}_1 = \{\, \mathcal{D}_{3\mathcal{A}_j},\, \mathcal{D}_{6\mathcal{A}_j},\, \mathcal{D}_{10\mathcal{A}_j} \,\}$ erstellt.

   Wende nun das Fragmentwise Intersection Verfahren auf $\mathcal{D}_{\mathcal{A}_{mask}}$ und sämtlichen Dokumenten innerhalb der Mengen $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k$ an, um dadurch Ähnlichkeitswerte zu erhalten. Das Resultat des Fragmentwise Intersection Verfahrens liefert dazu insgesamt k Folgen:

$$\begin{aligned}
\mathit{Similarities}_1 &= (\, S_{1,1},\, S_{1,2},\, \ldots,\, S_{1,|\mathcal{C}_1|} \,),\\
\mathit{Similarities}_2 &= (\, S_{2,1},\, S_{2,2},\, \ldots,\, S_{2,|\mathcal{C}_2|} \,),\\
&\vdots\\
\mathit{Similarities}_k &= (\, S_{k,1},\, S_{k,2},\, \ldots,\, S_{k,|\mathcal{C}_k|} \,)
\end{aligned}$$

   Aus jeder Folge $\mathit{Similarities}_i$ wird nun der höchste Ähnlichkeitswert $S_{i,max}$ ermittelt und in einer weiteren Folge gespeichert:

$$\mathit{Cluster\text{-}Similarities} = (\, S_{1,max},\, S_{2,max},\, \ldots,\, S_{k,max} \,)$$

6. Entnehme aus *Cluster-Similarities* den höchsten Ähnlichkeitswert $S_{y,max}$ und speichere dessen Kennung y in die Variable `Cluster-Max-Similarity`.

7. Terminiere anhand der Fallunterscheidung:

$$\theta = \begin{cases} \mathcal{A}_{mask} = \mathcal{A}_j & \text{, falls } \texttt{Cluster-Mask} = \texttt{Cluster-Max-Similarity} \\ \mathcal{A}_{mask} \neq \mathcal{A}_j & \text{, falls } \texttt{Cluster-Mask} \neq \texttt{Cluster-Max-Similarity} \end{cases}$$

Bemerkungen: Das Verfahren hat einige Vor- und Nachteile die im Folgenden betrachtet werden:

- Ein Vorteil des Cluster-Based Maximum Similarity Verfahrens stellt der zweistufige Prozess dar. Während in der ersten Phase sämtliche Feature-Vektoren (anhand von $k-$Means) hinsichtlich allgemeiner Features gruppiert werden, wird in der zweiten Phase eine verfeinerte Suche nach Stilähnlichkeit durchgeführt, indem die häufigsten übereinstimmenden Textfragmenten von $\mathcal{D}_{\mathcal{A}_{mask}}$ und sämtlichen $\mathcal{D}_{i\mathcal{A}_j}$, in Betracht gezogen werden.

- Bei der Entscheidung θ im letzten Schritt, handelt es sich nicht um einen numerischen Schwellwert (wie dies meistens für Verifikations-Verfahren der Fall ist), sondern um eine Bedingung mit zwei Wahrheitswerten. Der Vorteil hierbei ist der, dass dadurch kein entsprechender Schwellwert erraten bzw. empirisch ermittelt werden muss, was sich in der Praxis des Öfteren als sehr schwierig erweist.

- In Schritt 3 terminiert der Algorithmus mit der Entscheidung $\mathcal{A}_{mask} \neq \mathcal{A}_j$. Der Grund hierfür ist, dass $\mathcal{F}_{\mathcal{A}_{mask}}$ einen einelementigen Cluster (und dadurch einen Ausreißer) darstellt, sodass keine stilistische Ähnlichkeit zwischen $\mathcal{F}_{\mathcal{A}_{mask}}$ und sämtlichen $\mathcal{F}_{i\mathcal{A}_j} \in \mathbb{F}_{train}$ bestimmt werden konnte. Dies stellt einen Idealfall dar, sofern $\mathcal{D}_{\mathcal{A}_{mask}}$ nicht von $\mathcal{A}_j$ verfasst wurde. Allerdings ergibt sich leider ein entscheidender Nachteil, falls $\mathcal{D}_{\mathcal{A}_{mask}}$ tatsächlich von $\mathcal{A}_j$ verfasst wurde, da in diesem Fall die Entscheidung $\mathcal{A}_{mask} \neq \mathcal{A}_j$ geliefert wird, die so nicht stimmt. Das Verfahren eignet sich somit nicht für die Verifikation eines Dokuments $\mathcal{D}_{\mathcal{A}_{mask}}$ (eines wahren Autors $\mathcal{A}_j$), dessen Stil sich stark von dem der Trainingsdokumente unterscheidet.

7.3 Ansätze für die intrinsische Exploration

In diesem Unterkapitel wird ein zusammengesetztes Verfahren für die intrinsische Exploration vorgestellt. Zum besseren Verständnis der einzelnen Schritte sei hier auf die Tabelle 6.4.1 auf der Seite 79 verwiesen. Nachfolgend werden zunächst einige grundlegende Komponenten erläutert, bevor im Anschluß das Verfahren präsentiert wird, welches anhand dieser Komponenten Stil-Inkonsistenzen in einem Dokument aufzuspüren versucht.

7.3.1 Grundlegende Komponenten

Das im weiteren Verlauf vorgestellte Verfahren setzt zwei grundlegende Komponenten voraus:

- Ein beliebiges[1] Dokument $\mathcal{D}$, welches auf eine Stil-Inkonsistenz untersucht werden soll.

- Eine Zerlegungsstrategie mit deren Hilfe $\mathcal{D}$ in k Slices $\Xi = \{ \xi_1, \xi_2, \ldots, \xi_k \}$ segmentiert wird. Ein Slice ξ_i repräsentiert dabei ein Textfragment und kann je nach gewählter Zerlegungsstrategie, beispielsweise aus einer Menge von abgeschlossenen Sätzen bestehen.

Die intrinsische Exploration stellt, wie in Kapitel 4.3.2 erwähnt, eine Ein-Klassen Klassifikation dar. Die einzige Zielklasse die hierbei vorliegt, ist $c_1 =$ `Stil-Konsistenz`. Repräsentiert wird c_1 innerhalb $\mathcal{D}$ durch einen Anteil von Slices $A \subseteq \Xi$, die einen konsistenten Schreibstil ausdrücken. Slices die hingegen stilistische Abweichungen zum Gesamtstil von $\mathcal{D}$ aufweisen, werden durch die Menge $B \subset \Xi$ dargestellt. Diese stellen das Gegenteil von c_1 dar, welches durch $\neg c_1 =$ `Stil-Inkonsistenz` angegeben wird. Formal betrachtet, lautet somit das Ziel der intrinsischen Exploration, mindestens ein $\xi_i \in B$ zu finden. Falls keine entsprechenden Slices gefunden werden, wird die Entscheidung getroffen, dass $\mathcal{D}$ einen durchgehenden (bzw. konsistenten) Stil aufweist.

Annahme: Um eine intrinsische Exploration durchführen zu können, wird die Bedingung $|A| > |B|$ gefordert. Informell: Das Dokument muss stets mehr stilistisch konsistente als inkonsistente Textfragmente enthalten. Andernfalls stellt der konsistente Text selbst einen stilistischen Ausreißer dar.

[1] Mit *beliebig* ist hier ein Dokument gemeint, dessen Autorschaft bekannt oder anonym sein kann.

7.3.2 Zerlegungsstrategien

Bei der intrinsischen Exploration besteht der erste Schritt darin, eine geeignete Zerlegungsstrategie zu wählen, mit der $\mathcal{D}$ in k Slices aufgeteilt werden soll. Im Folgenden werden drei solcher Zerlegungsstrategien vorgestellt, von denen jede in dem präsentierten Verfahren eingesetzt werden kann.

7.3.2.1 Near-Same-Length Strategie

Bei dieser Strategie muss $\mathcal{D}$ zuallererst in Sätze tokenisiert werden. Anschließend erhält jeder der Slices $\xi_1, \xi_2, \ldots, \xi_{k-1}$ genau $m = \left\lfloor \frac{\#(\sigma,\mathcal{D})}{k} \right\rfloor$ Sätze, wobei $\#(\sigma, \mathcal{D})$ die Anzahl aller Sätze in $\mathcal{D}$ darstellt. Der letzte Slice ξ_k erhält die restlichen Sätze.

Ein Vorteil dieser Strategie ist, dass die Satzstruktur innerhalb der Slices erhalten bleibt. Dadurch können beliebige Features verwendet werden, insbesondere solche die eine vollständige Satzstruktur voraussetzen (syntaktische Features). Ein Nachteil dagegen ist, dass falls k ungerade gewählt wurde, der letzte Slice ξ_k, bedingt durch $\#(\sigma, \mathcal{D})$, einen restlichen Text erhält, der nicht auf die Slices $\xi_1, \xi_2, \ldots, \xi_{k-1}$ aufgeteilt werden konnte. Dadurch kann ξ_k stilistisch ähnlicher zu $\mathcal{D}$ wirken, da hier mehr Features enthalten sein können. Um diesem Problem entgegenzuwirken, sollten die Features daher stets normalisiert werden (z.B. durch die Anzahl aller Tokens in einem Slice).

7.3.2.2 Exact-Same-Length Strategie

Bei dieser Strategie erhalten alle Slices exakt die gleiche Länge, sodass:

$$\forall i, j \in \{\, 1, 2, \ldots, k \,\} : \big(\mathit{length}(\xi_i) = \mathit{length}(\xi_j) \big)$$

gilt. Falls ein restlicher Text bedingt durch die Aufteilung übrigbleibt, so wird dieser verworfen und nicht wie bei der Near-Same-Length Strategie dem letzten Slice zugeordnet. Ein Vorteil ergibt sich hier für buchstabenbasierte Features, da diese aufgrund der identischen Längen der Slices, einheitlich (bzw. überhaput nicht) normalisiert werden müssen. Ein weiterer Vorteil hinsichtlich der gleichlangen Slices ist der, dass eine annähernde Gleichverteilung[1] der Features über die Slices hinweg erreicht werden kann. Allerdings ergibt sich für diese Strategie auch ein entscheidender Nachteil, der lexikalische, syntaktische als auch semantische Features betrifft. Einige dieser Features werden nämlich unvermeidbar verloren gehen, da die Wort- bzw. Satzstruktur von $\mathcal{D}$ bei der Zerlegung (nach je ζ Zeichen) zerstört werden kann. Dies wird anhand der nachfolgenden Beispiele verdeutlicht:

- **Beispiel 1.)** Sei $\mathcal{D}$ = „`Sein oder nicht sein. Das ist hier die Frage.`“ und $k = 3$, dann ergibt diese Strategie eine Zerlegung mit den folgenden drei Slices (je $\zeta = 15$ Zeichen lang):

ξ_1	ξ_2	ξ_3
`Sein␣oder␣nicht`	`␣sein.␣Das␣ist␣`	`hier␣die␣Frage.`

  Hieraus lässt sich erkennen, wie die Sätze von $\mathcal{D}$ auseinandergenommen werden, sodass dadurch syntaktische und semantische Features verloren gehen können, da diese meistens auf eine vollständige Satzstruktur angewiesen sind.

- **Beispiel 2.)** Sei $\mathcal{D}$ = „`Ein kurzer Text.`“ und $k = 4$, dann teilt sich $\mathcal{D}$ wiederum in folgende Slices auf:

ξ_1	ξ_2	ξ_3	ξ_4
`Ein␣`	`kurz`	`er␣T`	`ext.`

[1] Dies betrifft zumindest zeichenbasierte Features.

In diesem Fall wird deutlich, dass neben der Satzstruktur auch zusätzlich die Wortstruktur zerstört wird. Dies hat zur Folge, dass neben einigen syntaktischen und semantischen Features auch zusätzlich lexikalische Features verloren gehen können. Damit ist die Exact-Same-Length Strategie eher für zeichenbasierte Features geeignet, da diese von der Zerlegung zwar auch betroffen sind, jedoch im geringen Maße.

7.3.2.3 $n-$Gramm Overlay Strategie

Diese Strategie verfolgt eine ähnliche Idee wie die Near-Same-Length Strategie. Anstelle von Sätzen werden hier jedoch Satz $n-$Gramme auf die Slices aufgeteilt. Hierfür wird $\mathcal{D}$ zunächst in m Sätze $\sigma_1, \sigma_2, \ldots, \sigma_m$ segmentiert, die anschließend anhand einer $n-$Gramm Konstruktion zu k überlappenden Slices zusammengeführt werden. Der Parameter n (der vom Benutzer festgelegt werden muss) gibt an, aus wievielen Sätzen ein ξ_i bestehen soll. Die folgende Abbildung zeigt schematisch den Ablauf der Zerlegung:

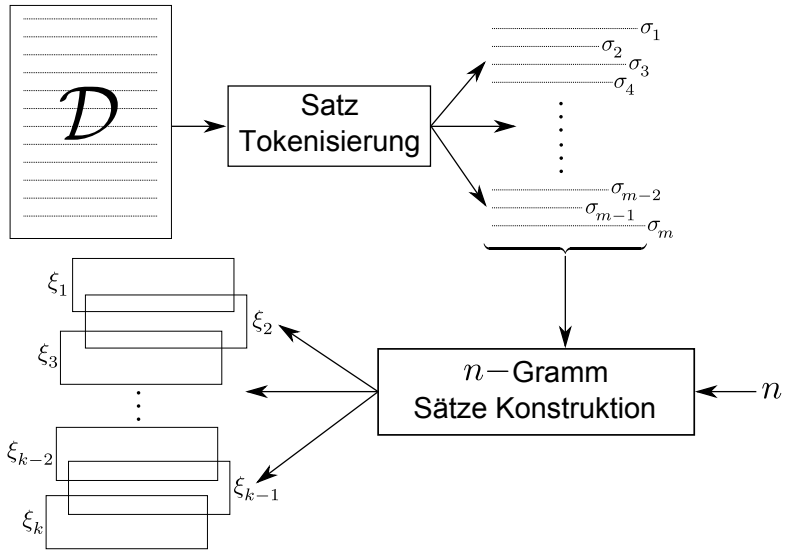


Abbildung 25: $n-$Gramm Overlay Methode: Dokumentzerlegung.

Die Vor- und Nachteile dieser Strategie sind die gleichen, wie bei der Near-Same-Length Strategie.

7.3.3 Cluster-Based Distinction Verfahren

In diesem Abschnitt wird ein Ansatz für die intrinsische Exploration vorgestellt. Hierbei handelt es sich um das sogenannte Cluster-Based Distinction Verfahren, welches eine Kombination des Fragmentwise Intersection- und des $k-$Means Verfahrens darstellt. Der Ansatz ist vierstufig aufgebaut, wobei die einzelnen Stufen wie folgt zusammengefasst werden:

- **Stufe 1:** Finden von repräsentativen Startpunkten für das $k-$Means Verfahren.
- **Stufe 2:** Durchführung von $k-$Means mit $k = 2$, sodass zwei Cluster resultieren.
- **Stufe 3:** Untersuchung der Cluster auf stilistische Inkonsistenzen.
- **Stufe 4:** Entscheidung ob eine Stil-Inkonsistenz in $\mathcal{D}$ vorliegt.

Die genaue Vorgehensweise des Verfahrens wird im Folgenden anhand dieser Stufen beschrieben:

Stufe 1: Ausgehend von $\mathcal{D}$ und einer gewählten Zerlegungsstrategie, werden in der ersten Stufe zunächst k Slices ξ_1, ξ_2, …, ξ_k generiert. Anschließend werden die $\ell-$häufigsten Textfragmente (mit Hilfe des Fragmentwise Intersection Verfahrens) aus jedem der Paare $(\xi_1, \mathcal{D})$, $(\xi_2, \mathcal{D})$, …, $(\xi_k, \mathcal{D})$ entnommen und in entsprechende Mengen M_{ξ_i} und $M_{\mathcal{D}}$ abgelegt. Auf jedes Mengenpaar wird anschließend

eine beliebige mengenbasierte Ähnlichkeitsfunktion angewendet, um so eine stilistische Ähnlichkeit $s_i = sim(M_{\xi_i}, M_{\mathcal{D}})$ bestimmen zu können. Jeder Ähnlichkeitwert s_i wird anschließend zusammen mit dem Slice ξ_i und dessen Kennung $\texttt{ID}_i$ in einer Folge gespeichert:

$$Similarities = \Big((s_1, \xi_1, \texttt{ID}_1),\, (s_2, \xi_2, \texttt{ID}_2),\, \ldots,\, (s_k, \xi_k, \texttt{ID}_k) \Big)$$

Im nächsten Schritt wird *Similarities* anhand der s_i absteigend sortiert, sodass sich diese wie folgt ändert:

$$Similarities' = \Big((s_{x_1}, \xi_{x_1}, \texttt{ID}_{x_1}),\, (s_{x_2}, \xi_{x_2}, \texttt{ID}_{x_2}),\, \ldots,\, (s_{x_k}, \xi_{x_k}, \texttt{ID}_{x_k}) \Big) \text{ , mit } x_i \in \{\, 1,\, 2,\, \ldots,\, k \,\}$$

Aus $Similarities'$ werden anschließend zwei Tupel entnommen, anhand derer in der nächsten Stufe, die Startpunkte für das $k-$Means Verfahren konstruiert werden. Hierbei wird zum einen $(s_{x_1}, \xi_{x_1}, \texttt{ID}_{x_1})$ und zum anderen $(s_{x_q}, \xi_{x_q}, \texttt{ID}_{x_q})$ als repräsentative Tupel ausgewählt. Beim zweiten Tupel stellt $q = \left\lceil \frac{k}{2} \right\rceil$ den Index dar. Die Semantik beider Tupel kann wie folgt verstanden werden:

- Das Tupel $(s_{x_1}, \xi_{x_1}, \texttt{ID}_{x_1})$ enthält den höchsten Ähnlichkeitswert s_{x_1} gegenüber allen anderen Tupeln, sodass der Slice ξ_{x_1} darin eine stilistische Konsistenz darstellt.
- Das Tupel $(s_{x_q}, \xi_{x_q}, \texttt{ID}_{x_q})$ enthält dagegen den mittleren Ähnlichkeitswert s_{x_q}, sodass der Slice ξ_{x_q} darin einen (leicht) verzerrten Stil repräsentiert. Die Annahme hierbei ist, dass mögliche stilistische Ausreißer eine höhere Ähnlichkeit zu diesem Slice aufweisen als zu ξ_{x_1}.

Diese beiden Tupel stellen das Resultat der ersten Stufe dar.

Stufe 2: In dieser Stufe wird das $k-$Means Verfahren initialisiert, bevor dieses dann auf sämtliche Slices angewendet wird. Da $k-$Means jedoch als Eingabe nur Feature-Vektoren akzeptiert, müssen diese zuvor aus den Slices generiert werden. Dazu werden (wieder) mit Hilfe des Fragmentwise Intersection Verfahrens, die $\ell-$häufigsten Textfragmente aus $\mathcal{D}$ entnommen, um damit den Feature-Vektor $\mathcal{F}_{\mathcal{D}}$ zu generieren. Dabei können die gleichen Textfragmente, wie aus der ersten Stufen verwendet, oder alternativ Andere gewählt werden. Für die k Slices $\xi_1, \xi_2, \ldots, \xi_k$ werden ebenfalls Feature-Vektoren generiert, deren Dimension die gleiche ist, wie die von $\mathcal{F}_{\mathcal{D}}$. Features, die in $\mathcal{F}_{\mathcal{D}}$ jedoch nicht in einen ξ_i vorkommen, werden mit 0 initialisiert. Das folgende Beispiel verdeutlicht die Konstruktion der Feature-Vektoren anhand eines kurzen Dokuments und dessen Slices:

Sei $\mathcal{D}$ = „`Ein kurzer Text.`“ und $k = 4$, dann liefert die Zerlegungsstrategie Exact-Same-Length die folgenden vier Slices:

ξ_1	ξ_2	ξ_3	ξ_4
`Ein␣`	`kurz`	`er␣T`	`ext.`

Als Textfragmente werden hierbei Symbole und Buchstaben (mit Beachtung der Groß- und Kleinschreibung) verwendet. Dadurch entstehen die folgenden $\ell = 13$ Features, die der Einfachheit halber nicht normalisiert werden:

Feature f_i	E	i	n	␣	k	u	r	z	e	T	x	t	.
Abs. Häufigkeit	1	1	1	2	1	1	2	1	2	1	1	1	1

Der Feature-Vektor $\mathcal{F}_{\mathcal{D}}$ kann direkt aus dieser Tabelle abgelesen werden::

$$\mathcal{F}_{\mathcal{D}} = (1, 1, 1, 2, 1, 1, 2, 1, 2, 1, 1, 1, 1)$$

Die Feature-Vektoren hinsichtlich der vier Slices sehen dagegen wie folgt aus:

$$\mathcal{F}_{\xi_1} = (1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0)$$
$$\mathcal{F}_{\xi_2} = (0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0)$$
$$\mathcal{F}_{\xi_3} = (0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0)$$
$$\mathcal{F}_{\xi_4} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1)$$

Die generierten Feature-Vektoren $\{ \mathcal{F}_{\xi_1}, \mathcal{F}_{\xi_1}, \ldots \mathcal{F}_{\xi_1} \}$ stellen die Trainingsmenge für $k-$Means dar. Der Feature-Vektor $\mathcal{F}_{\mathcal{D}}$ wird dagegen verworfen. Zusätzlich dazu benötigt $k-$Means Startpunkte, um die Cluster zu berechnen. Als Startpunkte werden zwei Medoide[1] verwendet, deren ID's aus der ersten Stufe bekannt sind. Der erste Medoid lautet somit $\mathcal{Z}_{\mathtt{ID}_{x_1}}$ und der zweite $\mathcal{Z}_{\mathtt{ID}_{x_q}}$. Zur Erinnerung: Medoide stellen Elemente der Trainingsmenge dar, sodass $\mathcal{Z}_{\mathtt{ID}_{x_1}} = \mathcal{F}_{\mathtt{ID}_{x_1}}$ und $\mathcal{Z}_{\mathtt{ID}_{x_q}} = \mathcal{F}_{\mathtt{ID}_{x_q}}$ gilt. Nachdem die Initialisierung abgeschlossen ist, wird $k-$Means auf die Trainingsmenge angewendet. Das Resultat ergibt eine Menge mit zwei Cluster $\mathbb{C} = \{ \mathcal{C}_{\mathtt{ID}_{x_1}}, \mathcal{C}_{\mathtt{ID}_{x_q}} \}$.

Stufe 3: In dieser Stufe gilt es eine Untersuchung beider Cluster auf stilistische Inkonsistenzen durchzuführen. Hierfür können unterschiedliche Strategien verwendet werden. Eine davon wäre beispielsweise die Berechnung der Durchmesser beider Cluster, anhand der Funktion $\mathit{Diameter} : \mathbb{C} \rightarrow \mathbb{R}$. Diese ist dabei wie folgt definiert:

$$\mathit{Diameter}(\mathcal{C}_i) = \underset{\mathcal{F}_j, \mathcal{F}_k \in \mathcal{C}_i}{\arg\max} \; \mathit{dist}(\mathcal{F}_j, \mathcal{F}_k) \;\; \text{mit} \;\; \mathcal{F}_j \neq \mathcal{F}_k$$

Der resultierende Wert dieser Funktion, kann dabei wie folgt interpretiert werden:

- Je kleiner $\mathit{Diameter}(\mathcal{C}_i)$ ist, desto ähnlicher verhalten sich die Slices (bzw. die Feature-Vektor Repräsentanten) darin zueinander.
- Umgekehrt gilt: Je größer $\mathit{Diameter}(\mathcal{C}_i)$ ist, desto größer ist auch die stilistische Varianz der Feature-Vektoren in $\mathcal{C}_i$.

Anstelle des Durchmessers kann alternativ auch die Kompaktheit der Cluster berechnet werden. Hierfür kann die folgende Formel benutzt werden:

$$\mathit{Compactness}(\mathcal{C}_i) = \frac{1}{|\mathcal{C}_i|} \sum_{\mathcal{F} \in \mathcal{C}_i} \mathit{dist}(\mathcal{F}, \mathcal{Z}_i)$$

Hierbei repräsentiert $\mathcal{Z}_i$ den Mittelpunkt (Medoid oder Zentroid) des Clusters $\mathcal{C}_i$ und $\mathit{dist}(\,\cdot, \cdot)$ wiederum eine beliebige Distanzfunktion, mit deren Hilfe die Abstände zwischen dem Mittelpunkt und allen Feature-Vektoren innerhalb $\mathcal{C}_i$ bestimmt werden. Die Intention dieses Maßes lautet:

- Je kleiner $\mathit{Compactness}(\mathcal{C}_i)$ ist, desto eher kann auf einen konsistenten Stil der Feature-Vektoren in $\mathcal{C}_i$ geschlossen werden.
- Umgekehrt gilt: Je größer $\mathit{Compactness}(\mathcal{C}_i)$ ist, desto höher ist auch die stilistische Varianz der Feature-Vektoren in $\mathcal{C}_i$.

Stufe 4: In dieser Stufe wird die Entscheidung gefällt, ob in $\mathcal{D}$ eine stilistische Inkonsistenz präsent ist. Ist dies der Fall, so liefert das Verfahren als Entscheidung: `Stil-Inkonsistenz`, andernfalls `Stil-Konsistenz`. Die Entscheidung ist dabei wie folgt definiert:

$$\theta_1 = \begin{cases} \texttt{Stil-Konsistenz} & \text{, falls } \; \mathit{Diameter}(\mathcal{C}_{\mathtt{ID}_{x_1}}) \leq \mathit{Diameter}(\mathcal{C}_{\mathtt{ID}_{x_q}}) \\ \texttt{Stil-Inkonsistenz} & \text{, sonst} \end{cases}$$

Alternativ dazu kann die Entscheidung hinsichtlich der Kompaktheit wie folgt ausgedrückt werden:

$$\theta_2 = \begin{cases} \texttt{Stil-Konsistenz} & \text{, falls } \; \mathit{Compactness}(\mathcal{C}_{\mathtt{ID}_{x_1}}) \leq \mathit{Compactness}(\mathcal{C}_{\mathtt{ID}_{x_q}}) \\ \texttt{Stil-Inkonsistenz} & \text{, sonst} \end{cases}$$

Damit ist die Aufgabe des Verfahrens abgeschlossen. Um $\mathcal{D}$ entsprechend aufzubereiten, sodass dadurch eine Attribution (bzw. Verifikation) ermöglicht werden kann, die vorher nicht möglich war, wird im nächsten Abschnitt das Verfahren um eine Postprocessing-Komponente erweitert. Hierfür werden

[1] Alternativ können auch Zentroide verwendet werden. Hierfür müsste jedoch die dritte Stufe des Verfahrens entsprechend angepasst werden.

beide Cluster $\mathcal{C}_{\mathtt{ID}_{x_1}}$ und $\mathcal{C}_{\mathtt{ID}_{x_q}}$ als Eingabe benötigt. Diese werden dahingehend modifiziert, dass die Feature-Vektoren darin entfernt, und stattdessen deren korrespondierende Tupel $(s_{x_i}, \xi_{x_i}, \mathtt{ID}_{x_i})$ in die entsprechenden Cluster eingefügt werden. Das folgende Beispiel verdeutlicht dies anhand von fünf Slices:

$$\begin{aligned} \mathcal{C}_{\mathtt{ID}_{x_1}} &= \{\, \mathcal{F}_{\xi_1}, \mathcal{F}_{\xi_2}, \mathcal{F}_{\xi_3} \,\} \quad \Rightarrow \quad \mathcal{C}_{\mathtt{ID}_{x_1}} = \{\, (s_1, \xi_1, \mathtt{ID}_1), (s_2, \xi_2, \mathtt{ID}_2), (s_3, \xi_3, \mathtt{ID}_3) \,\} \\ \mathcal{C}_{\mathtt{ID}_{x_q}} &= \{\, \mathcal{F}_{\xi_4}, \mathcal{F}_{\xi_5} \,\} \quad \Rightarrow \quad \mathcal{C}_{\mathtt{ID}_{x_q}} = \{\, (s_4, \xi_4, \mathtt{ID}_4), (s_5, \xi_5, \mathtt{ID}_5) \,\} \end{aligned}$$

7.3.4 Postprocessing-Komponente

In diesem Abschnitt wird das Cluster-Based Distinction Verfahren um eine Postprocessing-Komponente erweitert, mit derer Hilfe bestimmte Slices aus $\mathcal{D}$ wiederverwertet werden können. Als Eingabe für die Komponente dienen die beiden Cluster $\mathcal{C}_{\mathtt{ID}_{x_1}}$ und $\mathcal{C}_{\mathtt{ID}_{x_q}}$ aus dem vorherigen Abschnitt.

Die Postprocessing-Komponente erlaubt die folgenden zwei Strategien, um aus $\mathcal{D}$ Slices wiederzuverwerten:

1. **Style Inconsistency Cleaning:** Hierbei werden sämtlich Slices die Stil-Inkonsistenzen enthalten aus dem Dokument $\mathcal{D}$ entfernt, sodass dadurch ein modifiziertes Dokument $\mathcal{D}'$ entsteht, dessen Stil annähernd gleichverteilt ist. Realisieren lässt sich diese Strategie, indem sämtliche Slices aus dem Cluster $\mathcal{C}_{\mathtt{ID}_{x_1}}$ entnommen und zu $\mathcal{D}'$ konkateniert werden. Anschließend kann $\mathcal{D}'$ attribuiert (oder verifiziert werden).

2. **Style Consistency Cleaning:** Bei dieser Strategie wird genau anders vorgegangen. Diejenigen Slices, deren Stil annähernd gleich ist, werden verworfen, während Slices die stilistische Ausreißer enthalten, einzeln attribuiert werden. Dies hängt jedoch davon ab, ob die Textgröße der Slices ausreichend ist, um diese einzeln zu attribuieren. Ist dies nicht der Fall, so kann beispielsweise eine Bündelung ähnlicher Slices zu „Stil-Einheiten“ $\{\, E_1, E_2, \ldots \}$ vorgenommen werden, um dadurch jede Stil-Einheit E_i mit einem beliebigen Attributions-Verfahren gesondert attribuieren zu können. Die Bündelung der Slices zu Stil-Einheiten kann (wieder) mit dem $k-$Means Verfahren durchgeführt werden. Hier kommt jedoch das Problem auf, dass die Anzahl der Cluster (und damit die Anzahl der verschiedenen Stile) im Vorfeld nicht bekannt ist, anders als es beim Cluster-Based Distinction Verfahren der Fall war. Eine Lösung die sich hier anbietet ist es eine Clustering-Bewertung anhand des Silhouettenkoeffizienten[1] durchzuführen.

Das interessante an der zweiten Strategie ist, dass dadurch eine multiple Autorschafts-Attribution (aus einem Dokument heraus) ermöglicht werden kann. Allerdings müssen hier einige Annahmen getroffen werden:

- **Existenz verschiedener Stile:** Das Dokument $\mathcal{D}$ muss zwingend $\wp \geq 2$ Stile enthalten, die sich signifikant (in Abhängigkeit eines Schwellwerts θ) voneinander unterscheiden.
- **Unterschiedliche Autorenstile:** Die $\wp$ Stile in $\mathcal{D}$ müssen von unterschiedlichen Autoren stammen ($\mathcal{D}$ muss somit kollaborativ erstellt worden sein). Wenn diese Annahme nicht gilt (und $\mathcal{D}$ somit nur von einem Autor stammt), kann die Attribution fehlschlagen, da diese nur für stilkonforme Dokumente konzipiert ist.
- **Unterschiedliche Stil-Regionen:** Jeder Stil $\wp_i$ sollte möglichst von einem anderen Stil $\wp_j$ regional getrennt sein. Wenn die Stile vermischt sind (z.B. Paraphrasierungen von Autoren die sich gegeneinander korrigieren), kann keine korrekte Segmentierung des Textes (also kein eindeutiges Stil je Segment) erfolgen. Die regionale Trennung beginnt dabei stets ab der Satzebene.

Wenn diese Annahmen erfüllt sind, kann eine multiple Autorschafts-Attribution durchgeführt werden.

[1] Siehe die Beschreibung dazu in Kapitel 4.4.2.

7.4 Theoretische Ansätze

In diesem Abschnitt werden eigene theoretische Ansätze und Ideen für die Autorschaftsanalyse vorgeschlagen, welche sich mit dem heutigen Stand der Technik (zumindest teilweise) nicht praktisch durchführen lassen. Das primäre Ziel der vorgestellten Ansätze ist es Wissenschaftler aus dem Gebiet anzuregen, weiterführende Foschungsarbeit zu betreiben. Jeder Ansatz ist nach entsprechender Unterdisziplin (Attribution, Verifikation und intrinsischer Exploration) gekennzeichnet.

7.4.1 Autorschafts-Attribution: Pre-Profiling-Attribution

Dieser Ansatz kombiniert die Disziplin der Autorschafts-Attribution mit der des Sprachprofilings. Letztere kann als eine weitere Unterdisziplin der Autorschaftsanalyse verstanden werden. Die Idee des Ansatzes basiert im Wesentlichen darauf, eine gegebene Trainingsmenge $\mathbb{D}_{train}$ mit der Form:

$$\mathbb{D}_{train} = \{ \mathcal{D}_{1\mathcal{A}_1}, \mathcal{D}_{2\mathcal{A}_1}, \ldots, \mathcal{D}_{1\mathcal{A}_2}, \mathcal{D}_{2\mathcal{A}_2}, \ldots, \mathcal{D}_{r-1\mathcal{A}_m}, \mathcal{D}_{r\mathcal{A}_m} \}$$

zunächst nach bestimmten Parametern zu filtern, die innerhalb des anonymen Dokuments $\mathcal{D}_\varepsilon$ vorkommen. Bei den Parametern handelt es sich um sogenannte soziolinguistische Variablen (kurz SL-Variablen). Jede SL-Variable repräsentiert eine bestimmte Eigenschaft, wie etwa Alter, Geschlecht, Bildungsgrad, Dialekt, Sozialschicht oder auch Gruppenzugehörigkeit eines jeweiligen Autors. Auf der gefilterten Trainingsmenge wird anschließend ein beliebiges Attributions-Verfahren angewendet. Die Hoffnung dabei ist, dass der entsprechde Kandidat dadurch, zum einen schneller gefunden werden kann und zum anderen hinsichtlich der SL-Variablen zum unbekannten Autor „*kompatibel*“ ist. Die genauere Vorgehensweise des Ansatzes wird im Folgenden genauer erläutert.

Ausgehend von $\mathbb{D}_{train}$, $\mathcal{D}_\varepsilon$ und eine Menge von SL-Variablen die als $\mathcal{V} = \{ v_1, v_2, \ldots, v_n \}$ symbolisiert werden, gilt es zunächst $\mathbb{D}_{train}$ nach den darin befindlichen Autoren zu sortieren, sodass für jeden Autor eine Menge mit Beispiel-Dokumenten entsteht:

$$\begin{aligned}
\mathbb{D}_{\mathcal{A}_1} &= \{ \mathcal{D}_{1\mathcal{A}_1}, \mathcal{D}_{2\mathcal{A}_1}, \ldots, \} \\
\mathbb{D}_{\mathcal{A}_2} &= \{ \mathcal{D}_{1\mathcal{A}_2}, \mathcal{D}_{2\mathcal{A}_2}, \ldots, \} \\
&\vdots \\
\mathbb{D}_{\mathcal{A}_m} &= \{ \mathcal{D}_{1\mathcal{A}_m}, \mathcal{D}_{2\mathcal{A}_m}, \ldots, \}
\end{aligned}$$

Im nächsten Schritt gilt es für jedes der $\mathbb{D}_{\mathcal{A}_j}$ entsprechende Trainingsmengen $\mathcal{T}_{v_1}, \mathcal{T}_{v_2}, \ldots, \mathcal{T}_{v_n}$ zu generieren (für jede SL-Variable eine Trainingsmenge). Jede Menge $\mathcal{T}_{v_k}$ weist dabei die Form $\mathcal{T}_{v_k} = \{ \mathcal{F}_{1\mathcal{A}_j}, \mathcal{F}_{2\mathcal{A}_j}, \ldots \}$ auf. Ein Feature-Vektor $\mathcal{F}_{i\mathcal{A}_j} \in \mathcal{T}_{v_k}$ enthält Merkmale, die für v_k typisch sind und im korrespondierenden Dokument $\mathcal{D}_{i\mathcal{A}_j}$ vorkommen.

Im nächsten Schritt wird für jede Trainingsmenge $\mathcal{T}_{v_k}$ ein Ein-Klassen-Klassifikator classify$(\cdot)$ angewendet, um ein Modell zu erlernen, dass darüber entscheidet, welche $\mathcal{F}_{i\mathcal{A}_j} \in \mathcal{T}_{v_k}$ der Klasse v_k angehören (1) und welche nicht (0):

$$\text{classify}(\mathcal{F}_{i\mathcal{A}_j}) = b, \text{ mit } b \in \{ 0, 1 \}$$

Diese Prozedur wird für sämtliche Variablen $v_1, v_2, \ldots, v_n$ durchgeführt, sodass der folgende binärer Vektor entsteht:

$$\rho(\mathcal{A}_j) = (b_1, b_2, \ldots, b_n)$$

Der Vektor $\rho(\mathcal{A}_j)$ drückt aus, ob $\mathcal{A}_j$ über die entsprechenden SL-Variablen verfügt oder nicht. Für das unbekannte Dokument $\mathcal{D}_\varepsilon$ wird ebenfalls ein solches Profil erzeugt, welches durch $\rho(\varepsilon)$ symbolisiert wird. Nachdem für sämtliche $\mathcal{D}_{i\mathcal{A}_j} \in \mathbb{D}_{train}$ Profile generiert wurden, gilt es im nächsten Schritt aus $\mathbb{D}_{train}$ diejenigen Dokumente zu entfernen, deren Autoren keine übereinstimmenden SL-Variablen mit denen aus $\rho(\varepsilon)$ aufweisen. Das Resultat der Filterung ergibt somit eine Teilmenge $M \subseteq \mathbb{D}_{train}$

mit Autoren, die die gleichen SL-Variablen des unbekannten Autors ε aufweisen. Je nachdem, wieviele Variablen verwendet werden, kann dies den Umfang von $\mathbb{D}_{train}$ drastisch reduzieren, sodass $|M| \ll |\mathbb{D}_{train}|$ gilt. Im letzten Schritt wird schließlich auf M ein beliebiges Attributions-Verfahren angewendet, um so den zu ε stilistisch ähnlichsten Autor ausfindig zu machen.

Anmerkungen: Der Ansatz birgt (abgesehen von der Berechnungskomplexität) einige Herausforderungen und Fragestellungen, die vor allem die Generierung der Feature-Vektoren anbetreffen.

Aufgrund der Tatsache, dass SL-Variablen den Kern dieses Ansatzes darstellen, muss zunächst die Frage geklärt werden, wie und vor allem welche Features gefunden werden sollen, die SL-Variablen in den Dokumenten repräsentieren. In der Literatur existieren zu dieser Problematik Forschungsarbeiten, die zumindest für einige SL-Variablen erfolgsversprechende Resultate hervorgebracht haben. Dazu zählt unter anderem die Studie von Koppel [5] über die Klassifikation von Texten nach dem Geschlecht der Autoren. Koppel fand heraus, dass tatsächlich signifikante Unterschiede zwischen männlichen und weiblichen Autoren festgestellt werden können. So verwenden weibliche Autoren beispielsweise deutlich mehr Pronomen, während männliche Autoren dagegen mehr Substantive in ihren Texten verwenden, [5]. Neben solchen *Surface-Features*[1] wird vermutet, dass weitaus komplexere (hauptsächlich semantikbasierte) Features existieren, die jeweils für das eine oder andere Geschlecht spezifisch sind. Welche Features diese jedoch im Konkreten darstellen, kann aus aktueller Sicht (noch) nicht gesagt werden.

Was andere SL-Variablen, wie beispielsweise das Alter angeht, so existieren seit vielen Jahren sogenannte Lesbarkeitsformeln[2] (engl. *Readability Measures*), mit deren Hilfe das Alter (oder auch der Bildungsgrad) der Autoren geschätzt werden kann. Ein bekannter Vertreter dieser Formeln stellt z.B. der *Flesch Reading Ease Score* dar, welcher des Öfteren in Bildungsinstitutionen in der USA verwendet wird, um Texte dahingehend zu überprüfen, ob Kinder bzw. Schüler diese verstehen können. Lesbarkeitsformel stellen in der Praxis nur Annäherungen dar, sodass hierzu weitere Forschungsarbeit betrieben werden müsste, um verlässlichere Aussagen über das Alter der Autoren treffen zu können.

Fazit: Insgesamt kann nach aktuellem Stand, die automatische Klassifizierung von soziolinguistischen Variablen noch als Forschungsgegenstand betrachtet werden. Es existieren durchaus Ansätze (zumindest für einige SL-Variablen), die jedoch abhängig von der jeweiligen Ausgangssprache sind. Die meisten Ansätze und Methoden richten sich an der englischen Sprache aus, sodass für deutschsprachige Texte aktuell noch keine adäquaten Verfahren bekannt[3] sind. Sofern diese erforscht und prototypisch entwickelt werden, kann die Realisierung dieses Ansatzes angegangen werden.

7.4.2 Autorschafts-Attribution: Stylistic Devices Attribution

Beim Stylistic Devices Attribution handelt es sich um die Idee, eine Autorschafts-Attribution anhand von rhetorischen Stilmitteln zu ermöglichen. Diese werden zunächst in die Kategorien: *rhetorische Figuren* und *rhetorische Tropen* aufgeteilt und nach Kiefer, [54] wie folgt definiert:

Definition 7.1. ***Rhetorische Figuren:*** *„Rhetorische Figuren organisieren die Stellung und Beziehung von Wörtern zueinander. Sie dienen der rhetorischen Ausschmückung, außerdem der Verdeutlichung, Veranschaulichung und Verlebendigung einer sprachlichen Aussage durch syntaktische Besonderheiten. Dabei verändern sie den gemeinten, eigentlichen Wortlaut nicht“, [54].*

Definition 7.2. ***Rhetorische Tropen:*** *„Tropen sind sprachliche Ausdrucksmittel der uneigentlichen Rede, Wörter oder Wendungen, die im übertragenen oder bildlichen Sinne gebraucht werden (z.B.*

[1] Dies stellen in der Regel Features dar, die direkt aus dem Text ohne komplexe NLP-Werkzeuge (wie etwa Parser) entnommen werden können.

[2] Siehe hierzu einige Beispiele in Kapitel 5.6.4.

[3] Zumindest konnten hierfür im Rahmen einer Recherche keinerlei Anhaltspunkte (insbesondere auf automatisierten Verfahren) ermittelt werden.

»Blüte« für Jugend). Es besteht ein semantischer Unterschied zwischen dem Gesagten und dem Gemeinten, zwischen Zeichen und Bedeutung. Tropen dienten ursprünglich der Veranschaulichung von Sachverhalten", [54].

Zusammengefasst kann hier festgehalten werden, dass rhetorische Figuren die Syntax und rhetorische Tropen wiederum die Semantik eines Textes betreffen. Nachfolgend werden für beide einige Beispiele[1] aufgeführt:

- **Oxymoron** (griech.: scharfsinnige Dummheit): *wacher Schlaf*, *kalte Glut*, *beredetes Schweigen*, [54].
- **Pleonasmus** (griech.: Überfluss): *Weiser Schimmel*, *alter Greis*, [54].
- **Litotes** (griech.: Schlichtheit): *keine leichte Aufgabe* (statt: *sehr schwierige Aufgabe*), *nicht selten* (statt: *oft*), *nicht übel* (statt: *sehr gut*), [54].
- **Hendiadyoin** (griech.: eins durch zwei): *immer und ewig*, *voll und ganz*, *bitten und flehen*, [54].

Beispiele für rhetorische Tropen:

- **Metonymie** (griech.: Umbenennung): *Hüte deine Zunge!*, *ein Glas trinken*, [54].
- **Antonomasie** (griech.: Gegenbenennung): *der Korse* (statt *Napoleon*), *ein Judas* (statt *Verräter*), *ein Casanova* (statt *Frauenheld*), [54].
- **Synasthesie** (griech.: Zusammenempfinden): *Vom Licht berührt werden*, [54].
- **Hyperbel** (griech.: Übermaß): *Eine Ewigkeit warten*, *im Schneckentempo*, *blitzschnell*, [54].

Eine eigene Vermutung besagt, dass derartige Stilmittel in zahlreiche Texten (unabhängig vom Genre oder Register) vorkommen und dabei von den jeweiligen Autoren unterschiedlich eingesetzt werden, was wiederum zu einer stilistischen Differenzierung führen kann. Die Differenzierung betrifft die eingesetzten Stilmittel und deren Häufigkeit. Das besondere an rhetorischen Stilmitteln ist, dass diese (wie der Name es vermuten lässt) tatsächlich Stil wiederspiegeln, und nicht wie die meisten Features in der Literatur (und auch in dieser Arbeit), Stil anhand oberflächlicher Merkmale wie Wort-/Satzlängen, Buchstaben $n-$Gramme und Ähnliches zu approximieren versuchen.

Der Ansatz hat jedoch einige Herausforderungen, die dessen Realisierung betreffen. So müssen zunächst die jeweiligen Stilmittel innerhalb der Texte identifiziert werden, bevor sie als Features für die Attribution eingesetzt werden können. Doch genau hier liegt das Problem, da dies nur für einige rhetorische Figuren (die nur die reine Syntax betreffen) möglich ist. Die Identifizierung rhetorischer Tropen erfordert dagegen eine semantische Textanalyse, die unter anderem in der Lage sein müsste, Metaphern in Texten erkennen zu können. Es wird hierbei angenommen, dass eine derartige semantische Textanalyse in der aktuellen Forschung (wenn überhaupt) nur ansatzweise existiert.

Doch selbst unter der Annahme, dass rhetorische Stilmittel aus den Texten extrahiert werden können, stellt sich immernoch die Frage, ob deren Häufigkeiten tatsächlich ausreichen, um Autorenstile, vor allem in kürzeren Texten, verlässlich unterscheiden zu können. Diese Frage kann aus aktueller Sicht (noch) nicht beantwortet werden.

[1] Im Anhang (Seite 165) können weitere Beispiele betrachtet werden.

7.4.3 Autorschafts-Attribution/Verifikation: Lexical Chain Complexity

Das Lexical Chain Complexity Verfahren basiert auf eine eigene Vermutung, dass Autoren beim Verfassen eines Textes öfters ein (mehr oder weniger) einheitliches Muster hinsichtlich der Textstruktur verwenden. Dieses Muster zeichnet sich durch spezifische Einheiten (engl. *Entities*) im Text aus, die anhand von unterschiedlichen Beziehungen miteinander verbunden sind. Eine derartige Beziehung stellen sogenante Lexical Chains[1] dar. Nach Fankhauser et al. [27] kann unter diesem Begriff eine Folge von Wörtern verstanden werden, die semantisch verwandt sind und die Kohäsion eines Textes ausmachen. Koreferenzketten stellen eine Ausprägung von Lexical Chains dar. Der folgende Beispieltext aus [44] illustriert eine derartige Koreferenzkette anhand der hervorgehobenen Entities:

„Dieter Baumann hat sich in der Weltklasse zurückgemeldet: Im ersten Freiluftrennen nach Ablauf seiner zweijährigen Sperre feierte der 37-Jährige im italienischen Camaiore über 10000 Meter einen beeindruckenden Sieg. In der Weltjahresbestzeit von 27:38,51 Minuten schaffte der Tübinger zugleich die Qualifikation für die Leichtathletik-Europameisterschaften im August in München. [...] „Von den besten in Europa hat nur Ismail Sghir gefehlt. Deshalb ist es ein schönes Gefühl, hier zu gewinnen", strahlte der 37-Jährige trotzdem. Dennoch glänzte er 15 Tage vor seinem Marathon-Debut in Hamburg mit der drittbesten Zeit seiner Karriere und zeigte auch alte Qualitäten im Spurt. Nachdem er zwei Runden vor Schluss selbst die Führung übernommen hatte, ließ der Olympiasieger von 1992 dem Spanier Jose Rios (27:38,82) auf den letzten Metern keine Chance." [44]

Die Idee des Ansatzes ist es aus einem anonymen Dokument $\mathcal{D}_\varepsilon$ sowie allen $\mathcal{D}_{i\mathcal{A}_j}$ innerhalb einer Trainingsmenge, solche Lexical Chains ausfindig zu machen und diese nach Übereinstimmungen zu vergleichen. Der Vergleich kann unterschiedlich erfolgen, wobei im Folgenden eine Möglichkeit betrachtet wird.

Eine Lexical Chain kann stets als ein Graph $\mathcal{G}(\cdot)$ aufgefasst werden. So kann z.B. aus der oben aufgeführten Koreferenzkette (die wiederum eine Lexical Chain darstellt), der folgende Graph $\mathcal{G}(\mathcal{D})$ konstruiert werden:

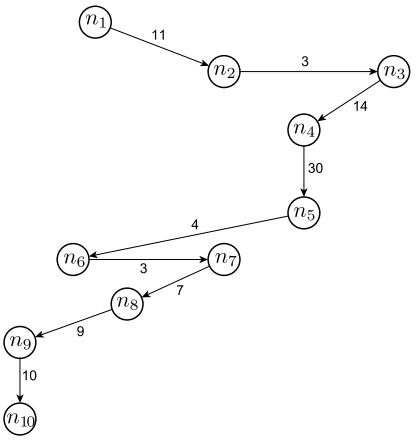


Abbildung 26: Graphdarstellung der Koreferenzkette von $\mathcal{D}$.

Die einzelnen Knoten $n_1, n_2, \ldots$ von $\mathcal{G}(\mathcal{D})$ stellen hierbei die Entities der Koreferenzkette dar, während die gewichteten Kanten in $\mathcal{G}(\mathcal{D})$, die Anzahl der Wörter repräsentieren, die zwischen n_i und n_{i+1} enthalten sind. Anhand eines solchen Graphen können unter anderem die folgenden stilistischen Messungen durchgeführt werden:

[1] Eine Recherche ergab für diesen Begriff leider keine deutsche Übersetzung.

- Messen wie variationsreich die einzelnen Entities n_1, n_2, ... vom Autor gewählt wurden. Eine eigene Vermutung ist hierbei, dass bei einem stilistisch erfahrenen Autor die meisten n_i voneinander verschieden sind.
- Messen wie groß die Distanzen zwischen zwei Entities n_i und n_{i+1} sind. Größere Distanzen deuten auf eine kohärente Form des Textes und möglicherweise auch auf eine komplexe Satzstruktur. Letztere ist öfters ein Indiz für einen erfahrenen Autor.
- Messen wieviele Entities eine Lexical Chain umfasst. Hierbei wird von einem stilistisch unerfahrenen Autor ausgegangen, wenn viele Entities (> 8) vorhanden sind, die sich nicht oder kaum voneinander unterscheiden. Wenige Entities (z.B. 4-6) die sich allesamt unterscheiden, deuten dagegen auf einen stilistisch erfahrenen Autor hin.

Eine Attribution ließe sich z.B. dadurch realisieren, dass zunächst für jeden Dokument innerhalb der Trainingsmenge mehrere „Trainings-Graphen“ $G_1(\mathcal{D}_{i_{\mathcal{A}_j}})$, $G_2(\mathcal{D}_{i_{\mathcal{A}_j}})$, ... generiert werden. Analog dazu werden für $\mathcal{D}_\varepsilon$ „anonyme Graphen“ $G_1(\mathcal{D}_\varepsilon)$, $G_2(\mathcal{D}_\varepsilon)$, ... erzeugt, um dadurch anhand der beschriebenen Messungen, Ähnlichkeiten zwischen den anonymen und den Trainings-Graphen zu überprüfen. Mit anderen Worten: Die Attribution basiert auf strukturellen Ähnlichkeiten zwischen diesen Graphen.

Wie bei den vorherigen Ansätzen existieren auch hier Herausforderungen, die die Realisierung erschweren. Die größte Herausforderung betrifft die Identifizierung von Lexical Chains. Zum einen müssen hier die jeweiligen Entities erkannt werden, wofür zwingend externes Wissen[1] benötigt wird, und zum anderen müssen die Zusammenhänge zwischen den Entities bestimmt werden. Während für englischsprachige Texte seit Jahrzehnten Ansätze existieren, sind für deutschsprachige Texte dagegen nur wenige Ansätze[2] bekannt. Eine weitere Herausforderung betrifft die Distanzen zwischen den Entities. Da diese in dem oben angegebenen Graphen nicht normalisiert sind, wird ein Ähnlichkeitsvergleich dadurch erschwert. Anstelle die Distanzen zwischen je zwei Knoten n_i und n_{i+1} auf eine Zahl (Anzahl der Tokens) abzubilden, kann stattdessen ein binärer Vektor erstellt werden, der die Wortarten zwischen n_i und n_{i+1} repräsentiert. Für eine gewählte Wortart (z.B. Artikel) wird so die 1 dargestellt, während andere alle anderen Wortarten durch Null ausgedrückt werden. Anhand eines solchen Vektors ließe sich möglicherweise ein besserer Ähnlichkeitsvergleich durchführen, wobei jedoch die Frage offen bleibt, wie groß dieser Vektor gewählt werden soll.

Neben solchen Herausforderungen enthält der Ansatz leider auch einige Nachteile. Einer davon ist z.B., dass sich dieser nicht für profilbasierte Verfahren eignet, da hier sämtliche Texte eines Autors zusammengeführt werden und dadurch die Information verloren geht, wo eine Lexical Chain anfängt und wo sie wiederum aufhört[3].

7.4.4 Autorschafts-Attribution/Verifikation: Ensemble Metric Scheme

Das Ensemble Metric Scheme beschreibt einen einfachen Ansatz, welcher aus dem Pairwise Similarity Verfahren hergeleitet werden kann. Die Attribution wird jedoch nicht anhand einer Metrik bestimmt, sondern anhand einer Mehrheitsentscheidung, an der k unterschiedliche Metriken beteiligt sind. Anhand der Metriken werden wahlweise Stilabweichungen, Stilähnlichkeiten oder eine Kombination aus beiden berechnet. Im Folgenden wird der Ansatz im Kontext der Autorschafts-Attribution genauer beschrieben. Hierbei werden der Einfachheit halber Distanzfunktionen als Metriken betrachtet.

Ausgehend von einem anonymen Dokument und einer Trainingsmenge die als Feature-Vektor Darstellungen $\mathcal{F}_\varepsilon$ und $\mathbb{F}'_{train}$ vorliegen, werden analog wie beim Pairwise-Similarity Verfahren Distanzen

[1] Hier wären beispielsweise Thesauri, Wörterbücher oder semantische Netze denkbar, um Entities (die in der Regel Eigennamen darstellen) identifizieren zu können.

[2] Siehe z.B. folgende Literaturempfehlung [18]

[3] Eine Lexical Chain zieht sich für gewöhnlich zwischen Anfang und Ende eines Textes.

zwischen $\mathcal{F}_\varepsilon$ und jedem $\mathcal{F}_{BIG\mathcal{A}_j} \in \mathbb{F}'_{train}$ berechnet. Für jede Distanzfunktion $dist_i(\cdot, \cdot)$ wird eine $\ell = |\mathbb{F}'_{train}|$ lange Folge erstellt, deren Elemente Distanzen (bzw. Stilabweichungen) und die assoziierten Autoren-Kennungen darstellen:

$$
\begin{aligned}
Distances_{\,1} &= \Big(\big(dist_1(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_1}), \mathcal{A}_1 \big),\ \big(dist_1(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_2}), \mathcal{A}_2 \big),\ \ldots,\ \big(dist_1(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_\ell}), \mathcal{A}_\ell \big) \big) \Big) \\
Distances_{\,2} &= \Big(\big(dist_2(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_1}), \mathcal{A}_1 \big),\ \big(dist_2(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_2}), \mathcal{A}_2 \big),\ \ldots,\ \big(dist_2(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_\ell}), \mathcal{A}_\ell \big) \big) \Big) \\
&\ \ \vdots \\
Distances_{\,k} &= \Big(\big(dist_k(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_1}), \mathcal{A}_1 \big),\ \big(dist_k(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_2}), \mathcal{A}_2 \big),\ \ldots,\ \big(dist_k(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_\ell}), \mathcal{A}_\ell \big) \big) \Big)
\end{aligned}
$$

Im nächsten Schritt werden alle Folgen aufsteigend sortiert, um anschließend aus jeder Folge $Distances_{\,i}$ den ersten Tupel $\big(dist_i(\mathcal{F}_\varepsilon, \mathcal{F}_{BIG\mathcal{A}_{x_1}}), \mathcal{A}_{x_1} \big)$ mit $x_1 \in \{\, 1,\, 2,\, \ldots,\, \ell \,\}$ zu entnehmen. Hierbei stellt $\mathcal{A}_{x_1}$ den zu ε stilistisch ähnlichsten Autor dar. Aus sämtlichen ersten Tupeln der k Folgen, werden anschließend die Autoren-Kennungen $\mathcal{A}_{x_i}$ entnommen, um diese in einer weiteren Folge abzuspeichern:

$$Candidates = \Big(\mathcal{A}_{x_1},\, \mathcal{A}_{x_2},\, \ldots,\, \mathcal{A}_{x_k} \Big)$$

Der letzte Schritt besteht darin, ausgehend von $Candidates$ eine Mehrheitsentscheidung hinsichtlich der $\mathcal{A}_{x_i}$ durchzuführen. Die Attribution erfolgt hierbei anhand dessjenigen Autors $\mathcal{A}_{freq}$, der innerhalb $Candidates$ am häufigsten vorkommt.

Aus technischer Sicht lässt sich dieses Verfahren ohne weiteres realisieren, jedoch nicht interpretieren. Dies betrifft hauptsächlich die Mehrheitsentscheidung, an der die unterschiedlichen k Metriken beteiligt sind. Das Problem dabei ist, dass zahlreiche Metriken nur schwer semantisch interpretiert werden können, was wiederum dazu führt, dass spezifische Vor- und Nachteile der Metriken nicht (oder nur teilweise) bekannt sind. Es müsste erforscht werden, wie genau sich bestimmte Metriken verhalten und welche von ihnen miteinander sinnvoll kombiniert werden können. Insbesondere sollte sichergestellt werden, dass die Kombination der Metriken robust ist, und keine signifikanten Anfälligkeiten[1] für bestimmte Domänen aufweisen.

Im Folgenden wird ein Beispiel gegeben, welche Distanzfunktionen keine sinnvolle Kombination darstellen. Dazu wird die parametrisierbare Minkowski-Distanzfunktion, anhand der beiden Vektoren $X, Y \in \mathbb{R}^n$ betrachtet:

$$dist_{Minkowski}(X, Y) = \sqrt[\lambda]{\sum_{i=1}^{n} \Big(|x_i - y_i|^\lambda \Big)^{\frac{1}{\lambda}}}$$

Je nachdem wie λ gewählt wird, ergeben sich unterschiedliche Formen des Einheitskreises[2], anhand dessen Distanzfunktionen in der Ebene geometrisch interpretiert werden können:

---

[1] In Kapitel 10.2.6 wird dazu ein Beispiel anhand einer Metrik gegeben.

[2] Ein Einheitskreis kann als ein einfacher Kreis verstanden werden, dessen Radius 1 beträgt und der Mittelpunkt den Koordinatenursprung darstellt.

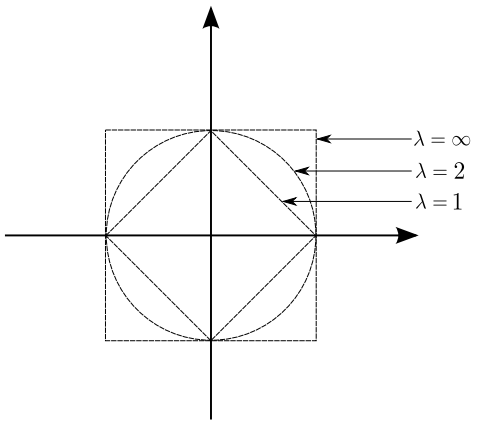


Abbildung 27: Geometrische Veranschaulichung von Minkowski-Distanzfunktionen.

Für $\lambda = \infty$ ergibt sich die Tschebyscheff-Distanz:

$$dist_{Tscheby}(X,Y) = \max\left(\,|x_1 - y_1|,\, |x_2 - y_2|,\, \ldots,\, |x_n - y_n|\,\right)$$

Wie aus der Abbildung 27 enonommen werden kann, schließt die Tschebyscheff-Distanz sämtliche Distanzfunktionen mit $\lambda < \infty$ ein, zu denen unter anderem die Euklidische Distanz $dist_{Euclid}(X,Y)$ für $\lambda = 2$ oder die Manhattan-Distanz $dist_{Manhattan}(X,Y)$ für $\lambda = 1$ dazugehören.

Eine Kombination von Minkowski-Distanzfunktionen ist somit nicht sinnvoll, da der Bildbereich stets von der Funktion dominiert wird, dessen λ größer ist. Werden dagegen Distanz- und Ähnlichkeitsfunktionen miteinander kombiniert, so scheint dies geeigneter zu sein, da hier auf unterschiedlichen Ebenen (bzw. Bildbereichen) Ähnlichkeiten gemessen werden. Eine von zahlreichen Kombination wäre z.B. die der Ähnlichkeitsfunktion $sim_{Cosine}(X,Y)$ mit der Distanzfunktion $dist_{Euclid}(X,Y)$. Erstere mißt dabei den Winkel zwischen X und Y, währernd die andere die Luftlinie zwischen beiden Vekoren berechnet (semantisch gesehen also zwei verschiedene Abstände).

Anmerkung: In der Disziplin der Textklassifikation wurden in der Vergangenheit bereits solche Ensemble-Ansätze vorgeschlagen[1], bei denen mehrere Distanzfunktionen für $k-$NN und andere Klassifikatoren kombiniert wurden. In der Disziplin der Autorschafts-Attribution (bzw. Verifikation) wurde ein solcher Ansatz, der insbesondere beliebige Metriken kombiniert (noch) nicht vorgefunden.

7.4.5 Intrinsische Exploration: Variant Spelling Complexity

Die Idee dieses Ansatzes ist es, Fehler innerhalb eines Dokuments ausfindig zu machen, die nur innerhalb eines bestimmten Abschnitts vorkommen, während die restlichen Textabschnitte frei von Fehlern sind. Dazu werden zunächst zwei Begriffe eingeführt:

- **Kompetenzfehler:** Dies sind Fehler, die durch mangelnde (Sprach-)Kenntnisse verursacht werden. In der Regel werden diese von Nicht-Muttersprachlern produziert.
- **Performanzfehler:** Dies sind Fehler, die hauptsächlich unbeabsichtigt verursacht werden (z.B. durch fehlende Konzentration). Diejenigen die solche Fehler produzieren, verfügen in der Regel über die nötigen Sprachkenntnisse.

Nach einer eigenen Vermutung wird hierbei angenommen, dass falls ein Text Performanzfehler enthält, diese nur an vereinzelten Stellen vorkommen werden. Enthält der Text dagegen Kompetenzfehler, so werden diese über den ganzen Text verteilt vorkommen. Wenn beim Zweiteren nur ein bestimmter Abschnitt Fehler und der restliche Text dagegen keine enthält, so liegt hier eine eindeutige Stil-Inkonsistenz vor. Diese ist z.B. dadurch bedingt, dass ein oder mehrere fremde Abschnitte im

[1] Siehe beispielsweise [114]

ursprünglichen Text eingebettet wurden oder, dass versucht wurde einen anderen Stil nachzuahmen, um so den eigenen Stil zu verbergen[1].

Das Ziel dieses Ansatzes ist es somit, Fehler im Text ausfindig zu machen und diese mit Hilfe eines Klassifikators, entsprechend beider Fehler-Typen zuzuordnen. Die Features hierfür können z.B. Wörter, Token $n-$Gramme oder Redewendungen (in Form von Konstituenten) darstellen. Nachdem die Fehler ermittelt und klassifiziert wurden, gilt es im nächsten Schritt deren Lokalisierung festzustellen. Hier können z.B. die Zerlegungsstrategien aus Kapitel 7.3.2 verwendet werden, um den Text in entsprechende Abschnitte zu segmentieren und dadurch beurteilen zu können, ob die Fehler global über dem ganzen Text verteilt sind, oder nur in einem Textsegment vorkommen.

Die größte Herausforderung dieses Ansatzes ist es herauszufinden, wie genau die jeweiligen Fehler im Text ausfindig gemacht werden können. Neben Effizienzüberlegungen hinsichtlich der Suche nach den Wörtern im Text, stellt sich die Frage, was mit Wörtern geschehen soll, die nicht in entsprechenden Wörterbüchern gefunden wurden, bzw. nicht klassifiziert werden konnten. Handelt es sich dabei um falsch geschriebene Wörter, oder ist das Wort einfach nur unbekannt[2]? Dass die Fehlersuche alles andere als einfach erscheint, kann anhand der meisten Rechtschreibprogramme beobachtet werden. Diese sind zwar durchaus in der Lage zahlreiche falschgeschriebene Wörter zu erkennen, aber auch hier gibt es klare Grenzen, insbesondere dann, wenn Texte Ausdrücke enthalten, die von der Alltagssprache abweichen. Neben der Erkennung von falschgeschriebenen Wörtern, stellt sich die nächste Frage, wie syntaktische und vor allem semantische Fehler ermittelt werden können, um dadurch Kompetenzfehler ausfindig zu machen. Zu den syntaktischen Fehlern zählen unter anderem falsche Wortstellungen wie z.B.: „*Annet nach Jamaica fliegt.*“ (*Subjekt-Objekt-Verb*). Nur wenige[3] Rechtschreibprogramme wären hier in der Lage solche Fehler zu erkennen. Deutlich aufwändiger gestalten sich semantische Fehler wie etwa: „*Das Haus geht zu Peter.*“ Hierbei wird angenommen, dass keine Rechtschreibprogramme existieren, die solche Fehler erkennen können. Der Grund ist, dass hier neben der reinen Semantik auch eine Logik involviert ist, die im Vorfeld erkannt werden müsste und dies zum einem sehr viel Aufwand und zum anderen den Einsatz von Weltwissen erfordern würde.

8 Vorstellung des Frameworks

In diesem Kapitel wird eine kurze Einführung in das Framework gegeben, welches im Rahmen dieser Arbeit implementiert wurde. Das Framework trägt die Bezeichnung Halvani AAF, wobei das Kürzel AAF für „Autorschaftsanalyse-Framework“ steht. Zunächst wird erleutert, welche Programmiersprache für die Implementierung gewählt wurde. Im Anschluß daran werden einige externe Werkzeuge und Komponenten genannt, die innerhalb von AAF zum Einsatz kamen. Zum Ende wird schließlich auf den internen Aufbau des Frameworks eingegangen. Hierbei werden die wichtigsten Module des AAF genannt und deren Aufgaben zusammengefasst.

8.1 Wahl der Programmiersprache

Als zugrundeliegende Programmiersprache für das Halvani AAF wurde Java ausgewählt. Der Grund dafür ist, dass für Java (gemessen an anderen Programmiersprachen) die meisten NLP-Bibliotheken existieren und diese für private bzw. wissenschaftliche Zwecke kostenlos[2] zur Verfügung stehen.

[1] Ein häufiges Beispiel hierfür stellen Erpresserschreiben dar, deren Autoren falsche Dialekte nachzuahmen versuchen.

[2] Anmerkung: *unbekannt* heißt nicht zwingend falsch.

[3] Streng genommen, kann an dieser Stelle der Begriff *wenige* nicht verwendet werden, da hier keinerlei Informationen vorliegen, ob überhaupt derartige Rechtschreibprogramme existieren.

[2] Bei den meisten Bibliotheken handelt es sich um Open-Source Projekte (GPL).

8.2 Externe Werkzeuge & Komponenten

Innerhalb des Halvani AAF wurden mehrere Werkzeuge und Komponenten aus unterschiedlichen Umfeldern eingesetzt. Im Folgenden werden diese hinsichtlich ihres Anwendungszwecks zusammengefasst.

8.2.1 Stanford NLP Toolkits

Die Stanford NLP Toolkits umfassen unterschiedliche NLP-Werkzeuge, die von der Forschungsgruppe „Stanford Natural Language Processing Group“ an der Universität Stanford entwickelt wurden. Zu diesen Werkzeugen zählen unter anderem POS-Tagger, Named Entity Recognizer, Parser, spezielle ML-Klassifikatoren oder auch maschinelle Übersetzungssysteme. Die Werkzeuge basieren auf State-of-The-Art Verfahren und erzielen dementsprechend sehr gute Ergebnisse in vielen Bereichen. Für das POS-Tagging erzielt z.B. der *Bidirectional MaxEnt Stanford Tagger* eine Erkennungsrate von 97.63 % für einige Domänen (z.B. Nachrichtentexte), [35].

Innerhalb des Halvani AAF wird der Stanford Parser, als auch der Standalone - POS Tagger *Bidirectional MaxEnt Stanford Tagger* verwendet. Für beide existieren deutsche Sprachmodelle (engl. *Model Files*), sodass dadurch eine Anwendung auf deutschsprachige Texte möglich ist.

8.2.2 OpenNLP

Analog zu den Stanford NLP Toolkits stellt das OpenNLP Framework ebenfalls verschiedene NLP-Komponenten zur Verfügung. Die meisten sind dabei eigenständige Komponenten (Standalone Werkzeuge), die in einer Pipeline-Architektur kombiniert werden können. Innerhalb des Halvani AAF wird OpenNLP hauptsächlich für die Tokenisierung von Wörtern, Sätzen sowie das Chunk-Parsing eingesetzt, um dadurch Dokumente hinsichtlich Wörter, Chunks (Phrasen) und Sätze zu annotieren. Die Chunks, die OpenNLP dabei generiert, sind leider nur auf Nominalphrasen beschränkt, da das zur Verfügung stehende Sprachmodell nur auf diese trainiert wurde.

8.2.3 JSoup - Java HTML Parser

Bei der JSoup Bibliothek handelt es sich um ein mächtiges Werkzeug, mit dessen Hilfe HTML Seiten, anhand des sogenannten „Document Object Model“ gezielt geparst werden können. Dies funktioniert sogar dann, wenn die Seiten nicht HTML-konform[1] erstellt wurden. Innerhalb des Halvani AAF wurde JSoup dazu verwendet, um aus Forenbeiträge (die in einer rohen HTML Struktur vorlagen) bestimmte Textabschnitte aus CSS bzw. HTML Container-Objekten extrahieren zu können. Die extrahierten Texte dienten wiederum der Erstellung von einigen Korpora.

8.2.4 NGramJ - Language Guessing Library

Die NGramJ Bibliothek stellt einen Language Guesser (siehe dazu Kapitel 2.6) dar und hat die Aufgabe für gegebene Textdokumente (oder Text-Fragmente) urteilen zu können, ob diese in der deutschen Sprache geschrieben sind oder nicht. NGramJ wird beim Halvani AAF in der Text-Normalisierungsstufe eingesetzt, um Forenbeiträge nach fremdsprachigen Inhalten zu filtern, sodass dadurch nur deutschprachige Beiträge in die Korpora aufgenommen werden.

8.2.5 Apache Lucene Core

Die Apache Lucene Bibliothek, welche ein Projekt der Apache Software Foundation darstellt, ist eine Ansammlung von Klassen, mit deren Hilfe Entwickler eigene Suchmaschinen realisieren können. Lucene ist vielfältig einsetzbar, unter anderem können damit Dokumente anhand von Gewichten/Rankings oder unscharfen Suchanfragen gefunden werden. Eine phonetische Suche ist dabei ebenfalls enthalten

[1] Beispielsweise nicht ordnungsgemäß geschlossene Tags, falsche Setzung von Anführungszeichen bei Tag-Attribute, etc.

(jedoch nur für einige Sprachen). Innerhalb des Halvani AAF wurde Lucene für „kleingewichtige“ Zwecke verwendet, wie z.B. Stemming von Wörtern oder die Suche nach Tokens anhand von bestimmten Positionen (z.B. Bestimmung des ersten Wortes in jedem Satz).

8.2.6 Weka & RapidMiner

Bei diesen beiden Frameworks handelt es sich um eine Ansammlung von zahlreichen Algorithmen aus dem Bereich des Maschinellen Lernens und des Data-Minings. Die Algorithmen (die bei RapidMiner als Operatoren bezeichnet werden) können dabei anhand einer intuitiven GUI instanziiert und miteinander kombiniert werden. Somit lassen sich z.B. ganze Prozessketten wie:

$$\text{Daten-Input} \rightsquigarrow \text{(Text-)Normalisierung} \rightsquigarrow \text{Feature-Auswahl} \rightsquigarrow \text{Klassifikation} \rightsquigarrow \text{Evaluierung}$$

definieren und automatisch durchführen. In der aktuellen Version von RapidMiner (5.2.008) sind sämtliche Weka Algorithmen enthalten, sodass nachfolgend nur von diesem Framework gesprochen wird. RapidMiner erlaubt es die unterschiedlichen Algorithmen auch außerhalb der GUI zu benutzen, um diese so in eigene Frameworks einzubinden, wie dies beim Halvani AAF z.B. der Fall ist. Hierbei wurde RapidMiner insbesondere für die Feature-Auswahl und die SVM Klassifikation verwendet. Außerdem wurden die selbst implementierten ML-Verfahren ($k-$NN, Naive Bayes und $k-$Means) mit denen aus RapidMiner verglichen, um dadurch Performanzunterschiede festzustellen und optimieren zu können.

8.3 Aufbau des Frameworks

Das Halvani AAF folgt einem modularen Design. Die folgende Abbildung zeigt die acht Kernmodule des Frameworks, die im Anschluß kurz zusammengefasst werden:

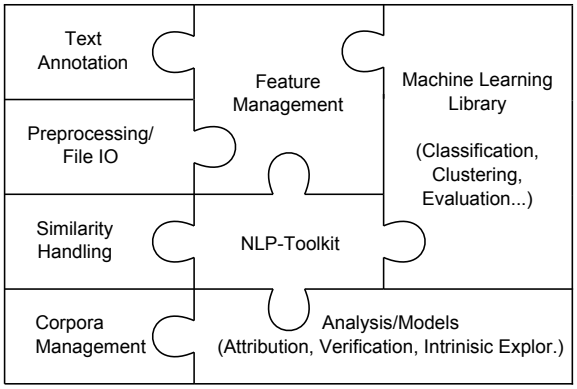


Abbildung 28: Die zentralen Module des Halvani AAF Frameworks.

1. **Text Annotation:** Dieses Modul realisiert die Annotation der Textdokumente hinsichtlich der sprachlichen Ebenen: Morphologie, Syntax und Semantik. Zu den Annotationen zählen unter anderem Tokens, Types, Stems, Anglizismen, Wortarten, Phrasen, Redewendungen, Nebensätze sowie vollständige Sätze. Die externen Werkzeuge, die hier eingesetzt werden, sind Stanford NLP Toolkits, OpenNLP sowie Apache Lucene.

2. **Corpora Management:** Dieses Modul realisiert die Kompilierung der annotierten Korpora, sowie deren Handhabung, die unter anderem die folgenden Aufgaben einschließt:

   - **Korpora Balancierung:** Vereinheitlichung der Korpora hinsichtlich Dokumentanzahl und Dokumentlängen (siehe dazu Kapitel 9.2).
   - **Profilerstellung:** Konkatenierung der Dokumente zu Autor-Profilen.
   - **Evaluierungsmechanismen/Korpora Aufteilung:** *Hold-Out*, *k−fold Cross-Validation* sowie *Leave-One-Out*.

3. **Machine Learning Library:** Dieses Modul beinhaltet Implementierungen der ML-Verfahren ($k-$NN, Naive Bayes und $k-$Means). Des Weiteren sind hier Schnittstellen zum RapidMiner Framework definiert, die es erlauben aus Halvani AAF heraus, einige der RapidMiner-Operatoren anzusteuern. Die wichtigste Schnittstelle ist die des SVM Klassifikators. Hierbei wurden im Vorfeld unterschiedliche SVM-Implementierungen ausprobiert, wobei für die Experimente in Kapitel 10 die Implementierungen LibSVM und mySVM ausgewählt wurden.

4. **Similarity Handling:** Dieses Modul umfasst Implementierungen von insgesamt 48 Metriken[1], für Mengen und binäre bzw. reele Vektoren. Die Metriken werden hauptsächlich von den Modulen *Machine Learning Library* und *Analysis* verwendet.

5. **Feature Management:** Die Kernfunktionalität dieses Moduls stellt die Feature-Vektor Generierung der Dokumente dar. Diese umfasst dabei die folgenden Komponenten:

   - **Feature-Entnahme:** Die Aufgabe dieses Moduls ist das Finden und Erstellen von Features, um diese später für die Feature-Vektor Generierung verwenden zu können. Hierbei werden unter anderem Wortlisten eingesetzt, für die eine effiziente Hashing-basierte Indizierung implementiert wurde.
   - **Feature-Normalisierung:** In diesem Modul werden sämtliche Features anhand der Strategie[2] „individuelle relative Häufigkeiten“ normalisiert. Durch die Normalisierung wird jedes Feature einheitlich auf das Intervall $[0\,;1]$ skaliert.
   - **Feature-Auswahl:** Dieses Modul enthält insgesamt 12 Feature-Selektionsalgorithmen, die optional von den Verfahren verwendet werden können. Die meisten Selektionsalgorithmen stellen hierbei RapidMiner-Operatoren dar, die aus Halvani AAF heraus instanziiert werden können.

   Eine wichtige Eigenschaft dieses Moduls ist, dass bestehende Feature-Kategorien sehr leicht erweitert werden können. Hierfür müssen lediglich Features in Form von Wortlisten ins Halvani AAF eingebunden werden. Dadurch können neue Features integriert werden, ohne dabei den Quellcode zu verändern.

6. **Preprocessing/File IO:** Dieses Modul hat viele unterschiedliche Aufgaben, die mit Dateien bzw. Datei-Inhalte zusammenhängen. Neben den Aufgaben wie Text-Normalisierung (inklusive HTML-Parsing) oder Häufigkeitsermittlung von Textfragmenten ($n-$Gramme, Tokens, Wortarten, etc.), werden in diesem Modul auch Feature-Matrizen generiert, die die Eingabe für die implementierten ML-Verfahren darstellen. Darüber hinaus werden hier unterschiedliche Objekte serialisiert, um dadurch die Anwendung von laufzeitintensiven NLP-Komponenten auf ein Minimum zu reduzieren. Des Weiteren sind hier mehrere Konvertierungswerkzeuge enthalten, die einen Austausch von Daten (bzw. Datenstrukturen) aus unterschiedlichen Quellen ermöglichen.

7. **NLP-Toolkit:** Dieses Modul umfasst sämtliche NLP-Komponenten wie etwa Tokenizer, POS-Tagger oder Chunker, mit deren Hilfe die Annotationen der Dokumente ermöglicht wurde.

8. **Analysis:** Bei diesem Modul handelt es sich um die wichtigste Komponente des Halvani AAF. Diese enthält die Implementierungen für sämtliche Autorschaftsanalyse-Verfahren, die in Kapitel 7 vorgestellt wurden.

[1] Auf Seite 36 wurde nur ein Teil der verwendeten Metriken vorgestellt.

[2] Siehe dazu Kapitel 5.7.1

9 Korpora

In diesem Kapitel werden diejenigen Korpora vorgestellt, welche die Grundlage für die Experimente in Kapitel 10 darstellen. Dazu wird zunächst die Notwendigkeit beschrieben, warum eigene Korpora erstellt werden mussten, anstatt bestehende Korpora zu beziehen. Anschließend werden die Unterschiede der einzelnen Korpora erläutert, wobei hier vor allem die sogenannte *Class Imbalance* Eigenschaft hervorgehoben wird. Danach wird der Prozess beschrieben, wie die Korpora erstellt wurden, bevor diese schließlich in Form einer Kurzbeschreibung sowie einer Statistik präsentiert werden.

9.1 Notwendigkeit eigener Korpora

Die Suche nach verwertbaren Korpora im Internet erweist sich als äußerst schwierig. Unter „verwertbare Korpora“ werden hierbei solche verstanden, die viele Texte (sortiert nach Autoren) in ausreichender Textlänge enthalten. Die Faktoren, die die Suche nach brauchbaren Korpora für die Autorschaftsanalyse erschweren, sind unter anderem urheberrechtlich bedingt. So müssen, zumindest nach deutschem Recht, Autoren ihre Zustimmung geben, bevor ihre Texte öffentlich zugänglich gemacht werden, was wiederum mit einem erheblichen Aufwand verbunden ist. Es existieren im Internet zwar einige Textarchive wie z.B. das „Projekt Guttenberg“ mit Werken, deren Urheberrechte abgelaufen[1] und somit frei zugänglich sind, jedoch sind diese sehr veraltet (teilweise aus dem 18. Jahrhundert und früher), sodass sich damit keine praxistauglichen Analyse-Szenarien[2] realisieren lassen.

Des Weiteren existieren im Internet einige Korpora von (anonymen Autoren), die z.B. aus Foren oder Blogs erschlossen wurden und für wissenschaftliche Zwecke frei zugänglich sind. Leider sind diese jedoch zum einen nicht deutschsprachig und zum anderen nur auf sehr spezifische Domänen fixiert (z.B. Filme oder biblische Literatur), sodass sich solche Korpora für diese Arbeit ebenfalls nicht eignen, da hier der Fokus auf deutschsprachige sowie domänenunabhängige Texte gelegt wird. Aus diesen Gründen wurde entschieden für die Experimente innerhalb dieser Arbeit eigene Korpora zu kompilieren. Hierfür wurden Texte aus unterschiedlichen Quellen erschlossen, die zum einen öffentlich verfügbar waren und zum anderen aus privaten Datenbeständen bezogen wurden. Alle Korpora enthalten dabei deutschsprachige Texte und weisen (zumindest teilweise) eine garantiert eindeutige Autorschaft aus. Für andere dagegen, kann die eindeutige Autorschaft mit einer hohen Wahrscheinlichkeit angenommen werden (mehr dazu später).

9.2 Class Imbalance

Die in dieser Arbeit verwendeten Korpora unterscheiden sich voneinander hinsichtlich mehrerer Parameter. Zu diesen zählen unter anderem Textsorte, Genre, Register, Jargon, Rauschen oder auch *Class Imbalance* (deut. *Klassen-Ungleichgewicht*). Der letzte Parameter ist dabei vom besonderem Interesse. Class Imbalance bescheibt im Rahmen dieser Arbeit zwei verschiedene problematische Eigenschaften:

1. Uneinheitliche Anzahl von Dokumenten (je Autor).
2. Unterschiedliche Länge von Dokumenten.

Bei der Erstellung von Korpora aus verschiedenen Quellen, wie beispielsweise E-Mails oder Forenbeiträge, treten diese Eigenschaften zwangsweise in Kombination auf. Dies lässt sich damit erklären, dass es Autoren gibt, die kurze (dafür aber viele) Dokumente verfassen und umgekehrt. Trifft die erste Eigenschaft auf einen Korpus zu, so wird dieser als *Instanz-unbalanciert* bezeichnet, trifft die zweite zu, so ist die Rede von *Längen-unbalanciert*. Falls beide Eigenschaften gleichzeitig zutreffen, so wird ein Korpus als *unbalnciert* bezeichnet, anderfalls *balanciert*.

[1] Laut § 64 UrhG „Das Urheberrecht erlischt siebzig Jahre nach dem Tode des Urhebers.“

[2] Hierzu zählen Szenarien mit Texten die alltäglich anzutreffen sind wie etwa E-Mails, Nachrichten in sozialen Netzwerken oder Forenbeiträge.

Die meisten Korpora, die in diesem Kapitel vorgestellt werden, liegen in einer unbalancierten Form vor. Die Balancierung lässt sich jedoch innerhalb des vorgestellten Frameworks regulieren, sodass dadurch bestimmte Klassifikatoren (z.B. die SVM) angewendet werden können, deren Schwächen unbalancierte Korpora darstellen. Auf der Seite 164 im Anhang kann eine Visualisierung der Korpora, hinsichtlich der Dokumentverteilungen (für jeden Autor) betrachtet werden.

9.3 Erstellung der Korpora

Die Erstellung der Korpora umfasste mehrere Schritte, die je nach Korpus durchlaufen werden mussten. Die einzelnen Schritte sind folgende:

- **Datenakquisition:** Einige Textdokumente, die ein Bestandteil der Korpora darstellen, stammten aus privaten Datenbeständen (z.B. E-Mails oder kostenpflichtige elektronische Magazine). Andere Dokumente mussten dagegen erst aus dem Internet heruntergeladen werden, bevor diese in ihrer Gesamtheit als Korpora kompiliert werden konnten. Um an diese Dokumente (bzw. Webseiten) zu gelangen, wurde ein Offline-Browser[1] eingesetzt.

- **Preprocessing:** Alle Dokumente, die nun lokal vorlagen, wurden mit Hilfe von eigenen Preprocessing Methoden automatisch normalisiert und mittels Postprocessing manuell nachbereinigt. Letzteres beinhaltete unter anderem das Filtern von Zitaten oder unleserlichen Inhalten (z.B. Formeln), die nicht durch das automatische Preprocessing entfernt werden konnten.

- **Zuordnung:**[2] Alle Dokumente mussten zu deren korrespondierenden Autoren zugeordnet werden, da ansonsten keine Evaluierung hinsichtlich der Autorschaftsanalyse-Verfahren stattfinden konnte. Dieser Schritt wurde semi-automatisch durchgeführt.

- **Annotation:** Um Features aus den sprachlichen Ebenen der Dokumente gewinnen zu können, mussten die Texte zuvor mit Hilfe unterschiedlicher Werkzeuge aus Kapitel 8 annotiert werden. Die Annotation umfasste unter anderem Tokens, Types, POS-Tags, Konstituenten und tokenisierte Sätze.

- **Anonymisierung:** Aus datenschutzrechtlichen Gründen mussten sämtliche Autoren (für jeden Korpus) nach einem gleichem Schema ($\mathcal{A}_1$, $\mathcal{A}_2$, ...) anonymisiert werden.

- **Aufteilung:** Um die Autorschaftsanalyse-Verfahren evaluieren zu können, ist eine Aufteilung der Korpora in Trainings- und Testmengen erforderlich. Dieser Schritt kann dabei wahlweise manuell (durch den Benutzer) oder automatisch (seitens des Frameworks) erfolgen.

Von besonderer Bedeutung ist der vierte Schritt (Annotation), bei dem linguistische Metadaten zu sämtlichen Dokumenten eines Korpus hinzugefügt wurden, um dadurch die sprachlichen Ebenen abzubilden. Hierfür wurden die Korpora inklusive dieser Metadaten serialisiert[3], um dadurch einen schnellen Zugriff auf die sprachlichen Ebenen zu erhalten und so nicht auf die laufzeitintensive Anwendung der NLP-Komponenten angewiesen zu sein.

---

[1] Hierbei handelte es sich um das Programm *HTTrack Website Copier*, [17].

[2] Dieser Vorgang wird im ML-Jargon als *Labeling* bezeichnet.

[3] Objekte sind in Programmiersprachen nur zur Laufzeit verfügbar, sodass nach der Terminierung eines Programms diese nicht mehr zur Verfügung stehen. Hierfür bietet sich die Serialisierung an, mit deren Hilfe Objekte dauerhaft in Form von Dateien gespeichert werden können. Diese können dann bei Bedarf zu einem späteren Zeitpunkt deserialisiert werden und sind damit wieder zur Laufzeit verfügbar.

9.4 Verwendete Korpora

In diesem Abschnitt werden sämtliche Korpora präsentiert, die zum Zwecke der Evaluierung der implementierten Verfahren verwendet wurden. Zu jedem Korpus wird dabei eine Kurzbeschreibung sowie eine Statistik in tabellarischer Form angegeben, welche versucht die interne Struktur der Korpora widerzuspiegeln. Die Spaltenbezeichnungen der Tabellen enthalten neben den bereits eingeführten Begriffen (Sätze, Phrasen, Types, etc.) zwei weitere Begriffe, die in den Grundlagenkapiteln nicht abgedeckt wurden: Anglizismen und Füllwörter. Unter dem ersten Begriff werden englische Wörter verstanden, die aus dem Englischen in eine andere Sprache (z.B. Deutsch) eingebracht werden wie etwa *Chat*, *E-Book*, *Scanner*, etc. Unter dem Begriff „Füllwörter“ werden dagegen Wörter ohne spezifische Bedeutung verstanden. Zu diesen zählen unter anderem überflüssige Adverbien, Konjuktionen, Dopplungen oder Relativierungen wie etwa *quasi*, *sozusagen*, *ziemlich*, etc. Um die Anglizismen und Füllwörter erkennen zu können, wurden diej Wortlisten aus Kapitel 5.6.5 verwendet.

Beobachtung: Ein häufiges Einsetzen von Anglizismen und Füllwörtern deutet meistens auf einen „schlechten Schreibstil“ hin und kann damit als Unterscheidungsmerkmal von Autoren dienen. Die Experimente in Kapitel 10 verdeutlichen diese These.

9.4.1 KOM Korpus

Beim KOM Korpus $\mathcal{K}_{KOM}$ handelt es sich um eine Studienarbeit mit dem Titel: „Storytelling und Digital Educational Games in der Hochschullehre“, welche von fünf Autoren am Fachgebiet KOM der TU Darmstadt angefertigt wurde. Da die Arbeit aus einem großen Dokument bestand, wurde diese in insgesamt 58 Abschnitte segmentiert, wobei jeder Abschnitt einem Autor eindeutig[1] zugeordnet ist. Die Abschnitte entsprechen somit den Dokumenten $\mathcal{D}_{i\mathcal{A}_j}$ (das i-te Dokument des j-ten Autors) von $\mathcal{K}_{KOM}$. Die Verteilung hinsichtlich der $\mathcal{A}_j$ und die Anzahl ihrer Dokumente $|\mathbb{D}_{\mathcal{A}_j}|$ ist dabei nicht einheitlich. Das gleiche gilt auch für die Längen der einzelnen $\mathcal{D}_{i\mathcal{A}_j}$, sodass damit $\mathcal{K}_{KOM}$ einen unbalancierten Korpus darstellt.

$\mathcal{K}_{KOM}$ hat eine interessante Eigenschaft, die ihn von den anderen Korpora unterscheidet. Sämtliche $\mathcal{D}_{i\mathcal{A}_j}$ (die von unterschiedlichen Autoren verfasst wurden) behandeln allesamt das gleiche Thema und weisen das selbe Vokabular (bestehend aus Wörter und Phrasen) auf. Diese Eigenschaft stellt eine Herausforderung dar, da genau dieses Vokabular die einzelnen Autorenstile nicht unterscheidet, zumindest nicht signifikant. Eine andere Eigenschaft von $\mathcal{K}_{KOM}$ betrifft dessen Umfang. Da dieser gegenüber allen anderen Korpora am kleinsten ist, bot es sich hier an, die implementierten Verfahren zunächst nur auf $\mathcal{K}_{KOM}$ zu testen und bei einem entsprechenden Erfolg auf andere Korpora anzuwenden. Es zeigte sich dabei, dass diejenigen Verfahren die brauchbare Ergebnisse für $\mathcal{K}_{KOM}$ geliefert haben, auch für andere Korpora gute und teilweise sogar bessere Ergebnisse vorweisen konnten. Damit kann $\mathcal{K}_{KOM}$ als ein Referenzkorpus für alle Verfahren in dieser Arbeit angesehen werden.

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	9	194	244	2662	1493	46 (1.73 %)	1543 (57.96 %)	15447	13.72	0.34
$\mathcal{A}_2$	12	256	413	4992	2608	76 (1.52 %)	3027 (60.64 %)	28345	19.50	0.72
$\mathcal{A}_3$	11	157	238	2663	1572	55 (2.07 %)	1510 (56.70 %)	15830	16.96	0.69
$\mathcal{A}_4$	17	189	439	4335	2526	46 (1.06 %)	2658 (61.31 %)	24623	22.94	1.21
$\mathcal{A}_5$	9	340	398	4656	2310	58 (1.25 %)	2810 (60.35 %)	26312	13.69	0.66

Tabelle 18: Statistik des $\mathcal{K}_{KOM}$

[1] Die eindeutige Autorschaft wurde von den einzelnen Autoren persönlich zugesichert.

9.4.2 FAZ Korpus

Beim FAZ Korpus $\mathcal{K}_{FAZ}$ handelt es sich um eine Kollektion von Artikel, die von Redakteuren der „Frankfurter Allgemeine Zeitung„ (Online-Version) verfasst worden sind. Die Artikel wurden dabei abgeschnitten, sodass daraus Dokumente mit annähernd gleichen Textlängen entstanden sind. Zusätzlich dazu wurde Wert darauf gelegt, dass jeder Autor die gleiche Anzahl an Dokumenten besitzt, sodass $\mathcal{K}_{FAZ}$ anhand beider Eigenschaften als balanciert betrachtet werden kann.

$\mathcal{K}_{FAZ}$ zeichnet sich in erster Linie durch eine journalistische Fachsprache aus. Die einzelnen Artikel umfassen dabei Themengebiete wie Politik, Wirtschaft, Finanzen, Gesellschaft, Technik und Wissen, sodass $\mathcal{K}_{FAZ}$ damit keinen einheitlichen Genre zugeordnet werden kann. Eine weitere Besonderheit von $\mathcal{K}_{FAZ}$ ist, dass dieser (bis auf Zitate) nahezu kein Rauschen enthält. Dies schließt vor allem auch Rechtschreibfehler mit ein, sodass fehlerbasierte Features in diesem Korpus kaum präsent sind.

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	8	204	347	4105	2464	50 (1.22 %)	2060 (50.18 %)	26920	20.12	1.30
$\mathcal{A}_2$	8	226	446	4830	2862	62 (1.28 %)	2808 (58.14 %)	29832	21.37	1.63
$\mathcal{A}_3$	8	245	464	4725	2821	53 (1.12 %)	2580 (54.60 %)	28220	19.29	1.28
$\mathcal{A}_4$	8	265	425	4371	2707	46 (1.05 %)	2491 (56.99 %)	26232	16.49	0.91
$\mathcal{A}_5$	8	230	454	4591	2817	53 (1.15 %)	2566 (55.89 %)	29578	19.96	1.29
$\mathcal{A}_6$	8	322	523	5578	3076	45 (0.81 %)	3238 (58.05 %)	32524	17.32	1.20
$\mathcal{A}_7$	8	267	470	5473	3000	47 (0.86 %)	2850 (52.07 %)	32670	20.50	1.01
$\mathcal{A}_8$	8	327	450	4850	2918	117 (2.41 %)	2692 (55.51 %)	28431	14.83	1.06
$\mathcal{A}_9$	8	306	469	4622	2700	27 (0.58 %)	2691 (58.22 %)	28272	15.10	0.83
$\mathcal{A}_{10}$	8	232	396	4497	2689	37 (0.82 %)	2345 (52.15 %)	28876	19.38	1.21

Tabelle 19: Statistik des $\mathcal{K}_{FAZ}$

9.4.3 Thesen Korpus

Beim Thesen Korpus $\mathcal{K}_{Thesen}$ handelt es sich um eine Kollektion aus Diplom-, Bachelor- und Master Arbeiten, die innerhalb des Fraunhofer-Instituts für Sichere Informationstechnologie von Studenten angefertigt wurden. Um aus den umfangreichen Arbeiten kleinere Dokumente erhalten zu können, wurden diese jeweils in 10 Abschnitte mit annähernd gleicher Länge segmentiert, sodass damit $\mathcal{K}_{Thesen}$ als ein balancierter Korpus angesehen werden kann. Der Korpus selbst, zeichnet sich dabei durch eine technische Fachsprache und spezielles Textrauschen auf. Dazu zählen z.B. zahlreiche Matheformeln, Quellcode-Abschnitte, symbolische Ausdrücke, Zitierungen und weiteres Textrauschen, das selbst mit manueller Nachbearbeitung nicht gänzlich entfernt werden konnte. Eine weitere Herausforderung von $\mathcal{K}_{Thesen}$ sind ähnliche Formulierungen bzw. Redewendungen, die vor allem in den Grundlagenabschnitte der Arbeiten enthalten sind und die Unterscheidung der Autorenstile erschweren.

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	10	470	509	7039	2872	91 (1.29 %)	3966 (56.34 %)	44758	14.98	0.74
$\mathcal{A}_2$	10	556	544	6949	3137	129 (1.86 %)	4010 (57.71 %)	43641	12.50	0.73
$\mathcal{A}_3$	10	378	380	5080	2512	103 (2.03 %)	2683 (52.81 %)	32380	13.44	0.71
$\mathcal{A}_4$	10	279	419	4685	2610	114 (2.43 %)	2780 (59.34 %)	27730	16.79	1.01
$\mathcal{A}_5$	10	458	506	6925	3595	196 (2.83 %)	3711 (53.59 %)	47502	15.12	0.64
$\mathcal{A}_6$	10	428	442	6072	2892	70 (1.15 %)	3449 (56.80 %)	38626	14.19	0.65
$\mathcal{A}_7$	10	457	538	7349	3749	324 (4.41 %)	3861 (52.54 %)	49428	16.08	0.93
$\mathcal{A}_8$	10	364	612	6722	3147	77 (1.15 %)	3916 (58.26 %)	41513	18.47	1.37
$\mathcal{A}_9$	10	531	454	7369	3633	203 (2.75 %)	4000 (54.28 %)	50634	13.88	0.52
$\mathcal{A}_{10}$	10	481	473	5906	2766	130 (2.20 %)	3377 (57.18 %)	38633	12.28	0.57
$\mathcal{A}_{11}$	10	379	385	5167	2849	160 (3.10 %)	2794 (54.07 %)	33730	13.63	0.71
$\mathcal{A}_{12}$	10	390	442	6966	3660	141 (2.02 %)	3776 (54.21 %)	46265	17.86	1.20
$\mathcal{A}_{13}$	10	472	582	7247	3480	117 (1.61 %)	3985 (54.99 %)	47911	15.35	0.75
$\mathcal{A}_{14}$	10	341	412	5963	2816	63 (1.06 %)	3366 (56.45 %)	39602	17.49	1.14
$\mathcal{A}_{15}$	10	333	481	6308	3131	97 (1.54 %)	3405 (53.98 %)	42747	18.94	0.87

Tabelle 20: Statistik des $\mathcal{K}_{Thesen}$

9.4.4 Mails Korpus

Beim Mails Korpus $\mathcal{K}_{Mails}$ handelt es sich um eine Kollektion von privaten und geschäftlichen E-Mails, die aus unterschiedlichen Kommunikationsquellen entstanden sind. Zu diesen Quellen zählen neben klassischer E-Mail-Korrespondenz auch moderne Kommunikationsformen wie XING, LinkedIn oder Facebook. Dementsprechend variieren die Themen der einzelnen Dokumente sehr stark voneinander, sodass hier kaum ein gemeinsamer Schnitt am gleichem Vokabular besteht. Neben der reinen Themenvielfalt enthalten die einzelnen Dokumente von $\mathcal{K}_{Mails}$ typische E-Mail-Merkmale, die eine besondere Herausforderung für die Autorschaftsanalyse darstellen. Zu diesen zählen z.B.:

- Rechtschreib-/Grammatikfehler (insbesondere falscher Satzbau, falsche Kommasetzung, etc.)
- Kurze bzw. teilweise abgehackte Sätze
- Vermischung von Slang, Jargon und Register
- Viele untypische Symbole/Ausdrücke (Smilies, E-Mail-Kürzel, Leetspeak Abkürzungen, etc.)
- Zahlreiche Out-of-Vocabulary Wörter (Wörter die von der Standardsprache abweichen)
- Inkonsistente Groß- und Kleinschreibung

Eine weitere Herausforderung des $\mathcal{K}_{Mails}$ betrifft dessen Aufbau. Da E-Mails üblicherweise sehr kurz gehalten sind, war es schwierig ausreichend Textumfang für die Trainingsmenge aufzubringen. Aus diesem Grund wurden mehrere kleinere E-Mails zu größeren Dokumenten zusammengeführt, um damit einen geeigneteren Rahmen für die Autorschaftsanalyse schaffen zu können. Durch die Zusammenführung kleinerer E-Mails entstand somit eine Vermischung unterschiedlicher Stile für einige Autoren[1].

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	6	135	206	2098	1352	35 (1.67 %)	1183 (56.39 %)	13288	15.54	0.86
$\mathcal{A}_2$	6	281	313	3066	1716	37 (1.21 %)	1915 (62.46 %)	16569	10.91	0.61
$\mathcal{A}_3$	5	116	166	1601	1054	24 (1.50 %)	895 (55.90 %)	9077	13.80	1.13
$\mathcal{A}_4$	6	248	424	3486	1700	46 (1.32 %)	2273 (65.20 %)	16460	14.06	0.62
$\mathcal{A}_5$	7	265	361	3671	2208	71 (1.93 %)	2108 (57.42 %)	21272	13.85	0.82
$\mathcal{A}_6$	6	154	324	2458	1562	54 (2.20 %)	1447 (58.87 %)	15362	15.96	0.73
$\mathcal{A}_7$	6	103	353	2877	1574	34 (1.18 %)	1986 (69.03 %)	14645	27.93	1.16
$\mathcal{A}_8$	6	269	363	2760	1573	59 (2.14 %)	1696 (61.45 %)	15349	10.26	0.37
$\mathcal{A}_9$	11	489	988	7661	4252	173 (2.26 %)	4753 (62.04 %)	42495	15.67	0.98
$\mathcal{A}_{10}$	5	187	310	2456	1446	21 (0.86 %)	1609 (65.51 %)	13260	13.13	0.95
$\mathcal{A}_{11}$	7	299	491	4372	2443	50 (1.14 %)	2573 (58.85 %)	25510	14.62	1.05
$\mathcal{A}_{12}$	5	95	158	1293	838	12 (0.93 %)	860 (66.51 %)	6508	13.61	1.11
$\mathcal{A}_{13}$	5	143	193	1772	1116	19 (1.07 %)	1128 (63.66 %)	9321	12.39	1.06
$\mathcal{A}_{14}$	8	401	777	6413	3550	99 (1.54 %)	3764 (58.69 %)	36098	15.99	1.16
$\mathcal{A}_{15}$	5	102	155	1540	1022	29 (1.88 %)	826 (53.64 %)	9260	15.10	0.96
$\mathcal{A}_{16}$	9	203	547	4746	2845	65 (1.37 %)	2887 (60.83 %)	26112	23.38	1.44
$\mathcal{A}_{17}$	5	121	196	1795	1052	26 (1.45 %)	1130 (62.95 %)	9212	14.83	0.93
$\mathcal{A}_{18}$	5	211	340	2623	1445	50 (1.91 %)	1748 (66.64 %)	13519	12.43	0.76
$\mathcal{A}_{19}$	5	235	315	3014	1540	91 (3.02 %)	1889 (62.67 %)	16116	12.83	0.34
$\mathcal{A}_{20}$	5	256	327	2443	1220	16 (0.65 %)	1739 (71.18 %)	11824	9.54	0.69
$\mathcal{A}_{21}$	5	256	385	3267	1689	30 (0.92 %)	1954 (59.81 %)	16314	12.76	1.02
$\mathcal{A}_{22}$	7	372	539	4960	2667	83 (1.67 %)	3104 (62.58 %)	27165	13.33	0.92
$\mathcal{A}_{23}$	6	195	383	3088	1748	62 (2.01 %)	1813 (58.71 %)	17462	15.84	1.51
$\mathcal{A}_{24}$	5	121	133	1685	1062	57 (3.38 %)	933 (55.37 %)	10523	13.93	0.64
$\mathcal{A}_{25}$	7	343	385	3782	2107	68 (1.80 %)	2151 (56.87 %)	21975	11.03	0.49
$\mathcal{A}_{26}$	5	153	258	2312	1459	23 (0.99 %)	1208 (52.25 %)	11686	15.11	0.19

Tabelle 21: Statistik des $\mathcal{K}_{Mails}$

[1] Betroffen sind hierbei die Autoren $\mathcal{A}_j$ mit $j \in \{3, 12, 13, 15, 17\}$

9.4.5 Recht Korpus

Beim Recht Korpus $\mathcal{K}_{Recht}$ handelt es sich um eine Kollektion von Forenbeiträge eines bekannten juristischen Recht-Portals[1]. Auch hier ist das Class Imbalance Problem vertreten, welches sich insbesondere bei den unterschiedlichen Dokumentlängen bemerkbar macht. $\mathcal{K}_{Recht}$ zeichnet sich durch ein formelles Register und eine juristische Fachsprache aus. Dementsprechendend finden sich in den Dokumenten Merkmale, die in keinem anderen Korpus vertreten sind. Dazu zählen z.B. Erläuterungen von Paragraphen, Fallbeispiele aus abgeschlossenen Gerichtsprozessen oder auch Eklärungen zu Vertragsklauseln, AGB's, etc. Interessant ist dabei, dass bei einer genaueren Betrachtung der Texte nur sehr wenige Rechtschreibfehler vorgefunden wurden (was für Forenbeiträge eher untypisch ist). Dies könnte eventuell dadurch begründet werden, dass die Autoren in $\mathcal{K}_{Recht}$ ausgebildete Juristen sind und bedingt durch den häufigen Schriftverkehr in ihrem Beruf weniger Rechtschreibfehler produzieren.

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	5	76	261	2164	1141	13 (0.60 %)	1532 (70.79 %)	10768	28.47	2.63
$\mathcal{A}_2$	6	123	224	2147	1372	21 (0.98 %)	1217 (56.68 %)	14004	17.46	1.04
$\mathcal{A}_3$	8	336	756	7558	3943	68 (0.90 %)	4355 (57.62 %)	44926	22.49	1.72
$\mathcal{A}_4$	6	142	170	2114	1397	16 (0.76 %)	1187 (56.15 %)	13500	14.89	0.46
$\mathcal{A}_5$	6	384	423	4068	2206	19 (0.47 %)	2560 (62.93 %)	22554	10.59	0.90
$\mathcal{A}_6$	6	121	255	2080	1313	31 (1.49 %)	1312 (63.08 %)	12321	17.19	1.25
$\mathcal{A}_7$	6	224	388	3704	1881	39 (1.05 %)	2328 (62.85 %)	21012	16.54	1.34
$\mathcal{A}_8$	7	355	643	6389	3204	57 (0.89 %)	3903 (61.09 %)	35648	18.00	0.83
$\mathcal{A}_9$	6	412	451	4459	2340	54 (1.21 %)	2757 (61.83 %)	26379	10.82	0.54
$\mathcal{A}_{10}$	6	150	231	2038	1176	16 (0.79 %)	1318 (64.67 %)	11795	13.59	0.79
$\mathcal{A}_{11}$	7	193	257	2329	1428	23 (0.99 %)	1454 (62.43 %)	13563	12.07	0.52
$\mathcal{A}_{12}$	6	172	294	2635	1576	15 (0.57 %)	1655 (62.81 %)	15415	15.32	0.86
$\mathcal{A}_{13}$	6	139	159	2576	1342	11 (0.43 %)	1484 (57.61 %)	17039	18.53	1.27
$\mathcal{A}_{14}$	8	475	723	6726	3411	44 (0.65 %)	4327 (64.33 %)	38299	14.16	1.04
$\mathcal{A}_{15}$	5	155	232	2005	1104	13 (0.65 %)	1254 (62.54 %)	11340	12.94	0.76
$\mathcal{A}_{16}$	6	134	249	2119	1231	15 (0.71 %)	1337 (63.10 %)	13016	15.81	1.16
$\mathcal{A}_{17}$	7	481	633	6044	3138	47 (0.78 %)	3672 (60.75 %)	34276	12.57	0.67
$\mathcal{A}_{18}$	6	143	340	2566	1480	36 (1.40 %)	1646 (64.15 %)	13798	17.94	1.19
$\mathcal{A}_{19}$	9	540	696	6920	3507	60 (0.87 %)	4045 (58.45 %)	41429	12.81	0.77
$\mathcal{A}_{20}$	5	104	227	1955	1102	15 (0.77 %)	1158 (59.23 %)	11785	18.80	0.76
$\mathcal{A}_{21}$	7	397	626	5314	2518	39 (0.73 %)	3472 (65.34 %)	28837	13.39	0.59
$\mathcal{A}_{22}$	5	176	246	1998	1104	24 (1.20 %)	1242 (62.16 %)	10952	11.35	0.65
$\mathcal{A}_{23}$	6	187	366	3026	1752	18 (0.59 %)	1887 (62.36 %)	16865	16.18	1.22
$\mathcal{A}_{24}$	8	330	429	4128	2321	39 (0.94 %)	2517 (60.97 %)	24923	12.51	0.68
$\mathcal{A}_{25}$	7	221	409	3865	2193	34 (0.88 %)	2355 (60.93 %)	22632	17.49	1.05
$\mathcal{A}_{26}$	6	156	247	2579	1420	17 (0.66 %)	1706 (66.15 %)	13165	16.53	0.77
$\mathcal{A}_{27}$	8	313	472	4401	2688	44 (1.00 %)	2566 (58.30 %)	26792	14.06	0.72
$\mathcal{A}_{28}$	6	106	258	2401	1415	13 (0.54 %)	1507 (62.77 %)	13971	22.65	1.70
$\mathcal{A}_{29}$	8	284	475	4190	2279	34 (0.81 %)	2626 (62.67 %)	24047	14.75	0.64
$\mathcal{A}_{30}$	6	180	274	2378	1396	22 (0.93 %)	1534 (64.51 %)	13109	13.21	0.69
$\mathcal{A}_{31}$	5	139	174	1850	1213	16 (0.86 %)	1082 (58.49 %)	11536	13.31	0.39
$\mathcal{A}_{32}$	6	245	291	3198	1925	13 (0.41 %)	1865 (58.32 %)	19676	13.05	0.65

Tabelle 22: Statistik des $\mathcal{K}_{Recht}$

9.4.6 CT Korpus

Beim CT Korpus $\mathcal{K}_{CT}$ handelt es sich um eine Kollektion elektronischer Ausgaben von c't-Magazinen (Jahrgänge 2008 - 2011), die von den Betreuern dieser Arbeit freundlicherweise zur Verfügung gestellt wurden. Die Magazine wurden hierbei nach ihren Autoren segmentiert[2]. Dadurch entstand eine Varianz hinsichtlich der Dokumentenanzahl je Autor als auch der Dokumentlängen, sodass $\mathcal{K}_{CT}$ als unbalanciert betrachtet werden kann.

[1] Die Webseite des Portals lautet: `http://www.Recht.de`

[2] Anders als in $\mathcal{K}_{Mails}$ wurde hier (aufgrund des ausreichenden Textumfangs) auf eine Zusammenführung der Texte verzichtet.

$\mathcal{K}_{CT}$ zeichnet sich durch ein technisches Jargon sowie ein beschränktes Genrespektrum aus, welches hauptsächlich praxisbezogene Computerthemen (z.B. Hardware/Software Berichte, Tests oder IT News) fokussiert. Analog zu $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Recht}$ enthält $\mathcal{K}_{CT}$ sehr wenig Textrauschen, das vermutlich durch die professionelle Schreibweise der Autoren bedingt ist. Dies hat zur Folge, dass hier vor allem Features, wie beispielsweise grammatikalische Fehler, nicht bzw. kaum vorhanden sind.

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	9	139	233	2625	1731	163 (6.21 %)	1393 (53.07 %)	16102	18.88	0.90
$\mathcal{A}_2$	13	394	468	5377	3244	145 (2.70 %)	2858 (53.15 %)	32708	13.65	0.66
$\mathcal{A}_3$	9	262	365	4660	2677	152 (3.26 %)	2559 (54.91 %)	28861	17.79	0.96
$\mathcal{A}_4$	9	205	374	4126	2659	149 (3.61 %)	2280 (55.26 %)	25321	20.13	1.10
$\mathcal{A}_5$	10	177	283	3394	2410	97 (2.86 %)	1681 (49.53 %)	23279	19.18	1.14
$\mathcal{A}_6$	8	114	217	2176	1389	171 (7.86 %)	1248 (57.35 %)	13488	19.09	1.30
$\mathcal{A}_7$	11	225	304	3664	2393	143 (3.90 %)	1964 (53.60 %)	22672	16.28	0.76
$\mathcal{A}_8$	9	191	300	3587	2265	158 (4.40 %)	1957 (54.56 %)	22834	18.78	0.93
$\mathcal{A}_9$	7	190	216	3044	1883	123 (4.04 %)	1578 (51.84 %)	18875	16.02	0.54
$\mathcal{A}_{10}$	14	195	315	4160	2780	178 (4.28 %)	2116 (50.87 %)	26256	21.33	0.99
$\mathcal{A}_{11}$	11	349	490	5766	3590	270 (4.68 %)	3044 (52.79 %)	37042	16.52	0.74
$\mathcal{A}_{12}$	12	247	365	4265	2728	143 (3.35 %)	2367 (55.50 %)	25101	17.27	0.99
$\mathcal{A}_{13}$	9	129	189	2209	1491	126 (5.70 %)	1157 (52.38 %)	13885	17.12	0.78
$\mathcal{A}_{14}$	10	224	333	4025	2564	158 (3.93 %)	2190 (54.41 %)	25359	17.97	0.81
$\mathcal{A}_{15}$	24	580	992	10998	6711	290 (2.64 %)	5960 (54.19 %)	69574	18.96	1.08
$\mathcal{A}_{16}$	13	305	464	5729	3357	277 (4.84 %)	3177 (55.45 %)	35693	18.78	1.08
$\mathcal{A}_{17}$	9	206	296	3687	2392	209 (5.67 %)	1891 (51.29 %)	22858	17.90	1.07
$\mathcal{A}_{18}$	8	152	176	2356	1662	40 (1.70 %)	1240 (52.63 %)	16736	15.50	0.86
$\mathcal{A}_{19}$	10	226	289	3266	2314	147 (4.50 %)	1649 (50.49 %)	20009	14.45	0.68
$\mathcal{A}_{20}$	9	285	362	4448	2687	171 (3.84 %)	2493 (56.05 %)	26389	15.61	0.85
$\mathcal{A}_{21}$	15	278	484	5290	3495	210 (3.97 %)	2889 (54.61 %)	33554	19.03	1.05
$\mathcal{A}_{22}$	13	190	256	3062	2138	89 (2.91 %)	1685 (55.03 %)	19202	16.12	0.98
$\mathcal{A}_{23}$	9	209	408	4188	2541	171 (4.08 %)	2396 (57.21 %)	25781	20.04	1.18
$\mathcal{A}_{24}$	8	191	404	3886	2453	117 (3.01 %)	2155 (55.46 %)	24880	20.35	1.21
$\mathcal{A}_{25}$	8	248	338	3998	2323	205 (5.13 %)	2267 (56.70 %)	23177	16.12	0.80
$\mathcal{A}_{26}$	9	201	264	3337	2140	128 (3.84 %)	1806 (54.12 %)	20501	16.60	0.92
$\mathcal{A}_{27}$	9	114	248	2780	1814	97 (3.49 %)	1567 (56.37 %)	17972	24.39	1.61
$\mathcal{A}_{28}$	12	166	293	2882	2042	94 (3.26 %)	1522 (52.81 %)	18126	17.36	0.87
$\mathcal{A}_{29}$	8	177	210	2712	1727	103 (3.80 %)	1459 (53.80 %)	16416	15.32	0.93
$\mathcal{A}_{30}$	10	278	463	5492	3334	166 (3.02 %)	2964 (53.97 %)	33082	19.76	1.23
$\mathcal{A}_{31}$	9	242	381	4145	2610	156 (3.76 %)	2379 (57.39 %)	25024	17.13	0.94
$\mathcal{A}_{32}$	9	270	432	4717	2760	118 (2.50 %)	2451 (51.96 %)	29641	17.47	0.67
$\mathcal{A}_{33}$	9	163	304	3415	2242	84 (2.46 %)	1771 (51.86 %)	21052	20.95	1.18
$\mathcal{A}_{34}$	14	271	475	5485	3475	241 (4.39 %)	3017 (55.00 %)	34759	20.24	0.93
$\mathcal{A}_{35}$	8	192	391	4363	2665	154 (3.53 %)	2333 (53.47 %)	28301	22.72	1.15
$\mathcal{A}_{36}$	12	231	374	4329	2801	236 (5.45 %)	2328 (53.78 %)	27110	18.74	1.22
$\mathcal{A}_{37}$	18	270	453	5303	3715	104 (1.96 %)	2776 (52.35 %)	34418	19.64	1.01
$\mathcal{A}_{38}$	10	292	425	5016	3077	144 (2.87 %)	2710 (54.03 %)	31510	17.18	0.79
$\mathcal{A}_{39}$	12	318	601	6445	3861	117 (1.82 %)	3565 (55.31 %)	39731	20.27	1.12
$\mathcal{A}_{40}$	12	378	564	6249	3591	301 (4.82 %)	3430 (54.89 %)	37796	16.53	0.89
$\mathcal{A}_{41}$	8	222	350	4290	2616	90 (2.10 %)	2376 (55.38 %)	27658	19.32	1.14
$\mathcal{A}_{42}$	8	111	144	2006	1327	145 (7.23 %)	1002 (49.95 %)	12741	18.07	0.87
$\mathcal{A}_{43}$	12	293	430	5265	3333	193 (3.67 %)	2773 (52.67 %)	31927	17.97	0.95
$\mathcal{A}_{44}$	15	266	364	4586	2960	242 (5.28 %)	2392 (52.16 %)	28834	17.24	0.66
$\mathcal{A}_{45}$	10	160	214	2629	1793	80 (3.04 %)	1395 (53.06 %)	16929	16.43	0.68
$\mathcal{A}_{46}$	11	257	388	5650	3275	242 (4.28 %)	3047 (53.93 %)	35429	21.98	0.87
$\mathcal{A}_{47}$	7	122	205	2031	1390	93 (4.58 %)	1088 (53.57 %)	12624	16.65	0.72
$\mathcal{A}_{48}$	10	178	313	3331	2136	72 (2.16 %)	1824 (54.76 %)	21488	18.71	0.75
$\mathcal{A}_{49}$	11	291	448	4690	2829	121 (2.58 %)	2726 (58.12 %)	28493	16.12	0.89
$\mathcal{A}_{50}$	9	177	260	3220	2125	158 (4.91 %)	1626 (50.50 %)	20529	18.19	0.95

Tabelle 23: Statistik des $\mathcal{K}_{CT}$

9.4.7 D120 Korpus

Das D120 Korpus $\mathcal{K}_{D120}$ basiert auf der gleichnamigen Webseite (`www.D120.de/forum`) der Informatik Fachschaft an der TU-Darmstadt. Es handelt sich hierbei um den größten Korpus innerhalb dieser Arbeit, sodass darauf etwas genauer eingegangen wird. Das D120 Forum repräsentiert ein großes Forum, welches in zahlreiche Unterforen gegliedert ist. Einige der Unterforen sind dabei für unregistrierte Benutzer offen zugänglich, sodass diese nicht nur Beiträge lesen sondern auch selbst verfassen können. Zu solchen „Pseudo-Benutzern“ zählen unter anderem Gäste sowie Job-Verteiler, die das Forum für Stellenausschreibungen nutzen. Hierbei kommt das Problem auf, dass die eindeutige Autorschaft aufgrund der fehlenden Registrierung nicht garantiert werden kann. Daher wurden diese im Vorfeld manuell ausgefiltert, sodass $\mathcal{K}_{D120}$ ausschließlich nur registrierte Benutzer umfasst, die erst nach einem entsprechenden Login Beiträge verfassen dürfen. Dadurch kann die eindeutige Autorschaft zumindest mit einer hohen Wahrscheinlichkeit angenommen werden. Die Forenbeiträge, die die Dokumente von $\mathcal{K}_{D120}$ darstellen, stammen von den 100 aktivsten Mitglieder des Forums. Da zahlreiche Beiträge dabei sehr kurz bemessen waren (z.B. zwei bis drei Sätze), wurden diese zu größeren Dokumenten zusammengefasst.

$\mathcal{K}_{D120}$ hat mehrere Eigenschaften, die sich von den anderen Korpora unterscheiden. Zunächst ist dieser ein stark unbalancierter Korpus und enthält, im Gegensatz zu allen anderen Korpora, das meiste Textrauschen. Neben Rechtschreibfehlern, mathematischen Formeln, Zitierungen, Code-Abschnitten, Forensignaturen und chattypischen Abkürzungen finden sich in $\mathcal{K}_{D120}$ auch viele fremdsprachige Abschnitte, die jedoch anhand einer Text-Normalisierung großteils entfernt werden konnten.

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_1$	6	151	268	2081	1355	57 (2.74 %)	1367 (65.69 %)	11284	13.78	1.17
$\mathcal{A}_2$	8	291	439	4104	2505	73 (1.78 %)	2541 (61.92 %)	23072	14.10	0.80
$\mathcal{A}_3$	18	612	1021	9367	5397	166 (1.77 %)	5829 (62.23 %)	53078	15.31	0.90
$\mathcal{A}_4$	19	991	1572	12992	6879	220 (1.69 %)	8434 (64.92 %)	73448	13.11	0.75
$\mathcal{A}_5$	6	178	326	3065	1718	62 (2.02 %)	1895 (61.83 %)	16758	17.22	1.22
$\mathcal{A}_6$	5	140	289	2488	1464	42 (1.69 %)	1554 (62.46 %)	14096	17.77	1.21
$\mathcal{A}_7$	7	322	615	4872	2318	36 (0.74 %)	3345 (68.66 %)	25201	15.13	0.72
$\mathcal{A}_8$	19	598	1250	10099	5777	107 (1.06 %)	6709 (66.43 %)	54822	16.89	1.10
$\mathcal{A}_9$	5	172	254	2317	1351	25 (1.08 %)	1532 (66.12 %)	13233	13.47	0.66
$\mathcal{A}_{10}$	5	191	284	2500	1418	40 (1.60 %)	1679 (67.16 %)	13399	13.09	0.79
$\mathcal{A}_{11}$	7	211	418	3496	2098	32 (0.92 %)	2214 (63.33 %)	20687	16.57	1.14
$\mathcal{A}_{12}$	5	192	354	3031	1629	109 (3.60 %)	1945 (64.17 %)	17152	15.79	1.06
$\mathcal{A}_{13}$	6	250	438	3669	2037	40 (1.09 %)	2407 (65.60 %)	20656	14.68	1.11
$\mathcal{A}_{14}$	5	132	413	2800	1396	17 (0.61 %)	1914 (68.36 %)	15070	21.21	1.70
$\mathcal{A}_{15}$	5	240	360	2953	1470	27 (0.91 %)	1967 (66.61 %)	15559	12.30	0.50
$\mathcal{A}_{16}$	5	157	330	2466	1504	37 (1.50 %)	1548 (62.77 %)	13308	15.71	0.97
$\mathcal{A}_{17}$	7	203	408	3532	2054	99 (2.80 %)	2153 (60.96 %)	18474	17.40	0.80
$\mathcal{A}_{18}$	19	658	1236	9975	5834	180 (1.80 %)	6381 (63.97 %)	53211	15.16	0.79
$\mathcal{A}_{19}$	10	285	781	5506	2793	64 (1.16 %)	3748 (68.07 %)	29311	19.32	1.54
$\mathcal{A}_{20}$	7	196	414	3572	2160	86 (2.41 %)	2254 (63.10 %)	20210	18.22	1.08
$\mathcal{A}_{21}$	7	242	363	3496	1891	50 (1.43 %)	2218 (63.44 %)	18315	14.45	0.78
$\mathcal{A}_{22}$	5	137	345	2456	1449	43 (1.75 %)	1595 (64.94 %)	13436	17.93	1.17
$\mathcal{A}_{23}$	43	1420	2654	22871	13418	332 (1.45 %)	14679 (64.18 %)	128667	16.11	1.05
$\mathcal{A}_{24}$	5	160	326	2512	1560	49 (1.95 %)	1574 (62.66 %)	14383	15.70	0.99
$\mathcal{A}_{25}$	5	187	373	3068	1474	28 (0.91 %)	2000 (65.19 %)	16177	16.41	1.12
$\mathcal{A}_{26}$	15	502	986	7580	4510	139 (1.83 %)	4881 (64.39 %)	42027	15.10	0.89
$\mathcal{A}_{27}$	7	227	465	3920	2144	70 (1.79 %)	2627 (67.02 %)	19621	17.27	0.91
$\mathcal{A}_{28}$	16	404	865	7914	4800	93 (1.18 %)	5143 (64.99 %)	44137	19.59	1.23
$\mathcal{A}_{29}$	5	135	380	2672	1410	36 (1.35 %)	1850 (69.24 %)	13948	19.79	0.96
$\mathcal{A}_{30}$	8	167	494	4271	2407	88 (2.06 %)	2845 (66.61 %)	22364	25.57	1.95
$\mathcal{A}_{31}$	6	218	396	3372	1833	54 (1.60 %)	2236 (66.31 %)	18107	15.47	0.80
$\mathcal{A}_{32}$	10	340	891	6463	2978	113 (1.75 %)	4450 (68.85 %)	33418	19.01	1.46
$\mathcal{A}_{33}$	12	441	659	5758	3428	62 (1.08 %)	3648 (63.36 %)	32882	13.06	0.87
$\mathcal{A}_{34}$	7	212	488	3757	1901	45 (1.20 %)	2597 (69.12 %)	19420	17.72	0.87
$\mathcal{A}_{35}$	5	170	273	2455	1336	28 (1.14 %)	1558 (63.46 %)	13566	14.44	1.20
$\mathcal{A}_{36}$	25	743	1359	12760	7730	368 (2.88 %)	7921 (62.08 %)	70603	17.17	0.77

$\mathcal{A}_j$	$\lvert\mathbb{D}_{\mathcal{A}_j}\rvert$	Sätze	Phrasen	Wörter	Types	Anglizismen	Füllwörter	Zeichen	Wörter/Satz	Kommas/Satz
$\mathcal{A}_{37}$	5	213	339	2587	1467	57 (2.20 %)	1739 (67.22 %)	13546	12.15	0.53
$\mathcal{A}_{38}$	16	475	969	8243	4818	127 (1.54 %)	5459 (66.23 %)	45071	17.35	0.96
$\mathcal{A}_{39}$	9	436	518	4716	2829	68 (1.44 %)	2889 (61.26 %)	26195	10.82	0.46
$\mathcal{A}_{40}$	10	461	800	6267	3083	97 (1.55 %)	4366 (69.67 %)	31138	13.59	0.43
$\mathcal{A}_{41}$	19	601	1018	8954	4942	151 (1.69 %)	5736 (64.06 %)	48208	14.90	0.88
$\mathcal{A}_{42}$	6	160	372	3128	1642	45 (1.44 %)	2024 (64.71 %)	16576	19.55	1.47
$\mathcal{A}_{43}$	5	147	356	2486	1395	32 (1.29 %)	1685 (67.78 %)	13252	16.91	1.18
$\mathcal{A}_{44}$	13	340	776	6638	3972	175 (2.64 %)	4289 (64.61 %)	36611	19.52	1.42
$\mathcal{A}_{45}$	6	181	323	3097	1821	54 (1.74 %)	1967 (63.51 %)	17535	17.11	1.14
$\mathcal{A}_{46}$	9	256	521	4496	2779	70 (1.56 %)	2836 (63.08 %)	24313	17.56	1.16
$\mathcal{A}_{47}$	5	150	242	2348	1556	69 (2.94 %)	1330 (56.64 %)	14512	15.65	0.79
$\mathcal{A}_{48}$	11	350	830	6466	3573	72 (1.11 %)	4372 (67.62 %)	35797	18.47	1.15
$\mathcal{A}_{49}$	10	314	565	5058	2992	80 (1.58 %)	3276 (64.77 %)	28371	16.11	1.17
$\mathcal{A}_{50}$	10	354	431	4811	2999	131 (2.72 %)	2698 (56.08 %)	26896	13.59	0.67
$\mathcal{A}_{51}$	5	227	250	2530	1456	37 (1.46 %)	1495 (59.09 %)	14321	11.15	0.62
$\mathcal{A}_{52}$	13	328	822	6862	4093	158 (2.30 %)	4342 (63.28 %)	36132	20.92	0.94
$\mathcal{A}_{53}$	5	127	261	2454	1418	33 (1.34 %)	1510 (61.53 %)	13455	19.32	0.90
$\mathcal{A}_{54}$	8	161	459	4400	2445	54 (1.23 %)	2846 (64.68 %)	23430	27.33	1.25
$\mathcal{A}_{55}$	20	480	1201	10542	6144	186 (1.76 %)	6527 (61.91 %)	58501	21.96	1.19
$\mathcal{A}_{56}$	9	348	599	4607	2547	97 (2.11 %)	3097 (67.22 %)	24162	13.24	0.91
$\mathcal{A}_{57}$	5	150	347	2502	1246	56 (2.24 %)	1564 (62.51 %)	13663	16.68	1.20
$\mathcal{A}_{58}$	11	398	712	5892	3247	128 (2.17 %)	3623 (61.49 %)	32416	14.80	0.85
$\mathcal{A}_{59}$	6	200	329	3029	1663	60 (1.98 %)	1874 (61.87 %)	15915	15.15	0.84
$\mathcal{A}_{60}$	8	284	426	4063	2373	39 (0.96 %)	2509 (61.75 %)	23224	14.31	0.87
$\mathcal{A}_{61}$	15	396	839	7604	4504	177 (2.33 %)	4757 (62.56 %)	42329	19.20	1.15
$\mathcal{A}_{62}$	33	773	2021	16882	10330	223 (1.32 %)	10627 (62.95 %)	92597	21.84	1.46
$\mathcal{A}_{63}$	24	626	1372	12093	7324	232 (1.92 %)	7823 (64.69 %)	65484	19.32	1.31
$\mathcal{A}_{64}$	15	533	811	6956	3945	125 (1.80 %)	4112 (59.11 %)	40300	13.05	1.13
$\mathcal{A}_{65}$	5	168	254	2614	1417	62 (2.37 %)	1658 (63.43 %)	14082	15.56	1.09
$\mathcal{A}_{66}$	6	248	216	2636	1482	36 (1.37 %)	1479 (56.11 %)	15720	10.63	0.80
$\mathcal{A}_{67}$	5	78	332	2819	1570	34 (1.21 %)	1857 (65.87 %)	14961	36.14	1.82
$\mathcal{A}_{68}$	5	185	333	2740	1562	44 (1.61 %)	1809 (66.02 %)	13994	14.81	0.95
$\mathcal{A}_{69}$	7	228	457	3749	2126	81 (2.16 %)	2456 (65.51 %)	19462	16.44	1.29
$\mathcal{A}_{70}$	7	241	425	3636	2099	137 (3.77 %)	2294 (63.09 %)	19823	15.09	1.00
$\mathcal{A}_{71}$	21	914	1315	11809	6645	237 (2.01 %)	7617 (64.50 %)	60158	12.92	0.21
$\mathcal{A}_{72}$	28	1172	1967	15331	8568	212 (1.38 %)	10048 (65.54 %)	84670	13.08	0.67
$\mathcal{A}_{73}$	10	329	554	4863	2600	77 (1.58 %)	3129 (64.34 %)	25060	14.78	0.49
$\mathcal{A}_{74}$	13	381	687	6194	3646	106 (1.71 %)	3769 (60.85 %)	36598	16.26	0.86
$\mathcal{A}_{75}$	9	299	596	4765	2611	66 (1.39 %)	3029 (63.57 %)	26374	15.94	0.74
$\mathcal{A}_{76}$	16	560	869	8723	5039	168 (1.93 %)	5222 (59.86 %)	51440	15.58	0.83
$\mathcal{A}_{77}$	6	130	402	3325	1927	62 (1.86 %)	2142 (64.42 %)	19159	25.58	2.12
$\mathcal{A}_{78}$	6	177	364	2804	1675	55 (1.96 %)	1691 (60.31 %)	16248	15.84	0.94
$\mathcal{A}_{79}$	9	295	504	4730	2741	79 (1.67 %)	3083 (65.18 %)	24510	16.03	0.78
$\mathcal{A}_{80}$	15	489	859	8130	4747	139 (1.71 %)	5113 (62.89 %)	46156	16.63	0.99
$\mathcal{A}_{81}$	7	296	456	3907	2312	51 (1.31 %)	2324 (59.48 %)	22500	13.20	0.91
$\mathcal{A}_{82}$	19	630	1185	9520	5362	161 (1.69 %)	6033 (63.37 %)	51724	15.11	0.75
$\mathcal{A}_{83}$	6	167	368	2602	1478	39 (1.50 %)	1778 (68.33 %)	13769	15.58	0.79
$\mathcal{A}_{84}$	47	1409	2378	24018	13882	311 (1.29 %)	15170 (63.16 %)	139521	17.05	1.04
$\mathcal{A}_{85}$	10	396	646	5348	2947	76 (1.42 %)	3340 (62.45 %)	28732	13.51	0.91
$\mathcal{A}_{86}$	7	282	448	3734	2150	77 (2.06 %)	2267 (60.71 %)	20233	13.24	0.62
$\mathcal{A}_{87}$	5	176	289	2888	1647	62 (2.15 %)	1742 (60.32 %)	17234	16.41	1.29
$\mathcal{A}_{88}$	14	392	711	6850	4218	138 (2.01 %)	4320 (63.07 %)	38844	17.47	0.97
$\mathcal{A}_{89}$	6	195	376	2659	1410	53 (1.99 %)	1808 (68.00 %)	13907	13.64	0.82
$\mathcal{A}_{90}$	8	298	522	4264	2443	88 (2.06 %)	2680 (62.85 %)	25259	14.31	0.83
$\mathcal{A}_{91}$	10	279	501	5010	2819	95 (1.90 %)	3112 (62.12 %)	27090	17.96	1.28
$\mathcal{A}_{92}$	5	178	355	2614	1485	33 (1.26 %)	1755 (67.14 %)	13500	14.69	0.70
$\mathcal{A}_{93}$	6	250	458	4082	2176	68 (1.67 %)	2684 (65.75 %)	22998	16.33	1.32
$\mathcal{A}_{94}$	13	430	730	6871	3957	86 (1.25 %)	4518 (65.75 %)	38497	15.98	0.95
$\mathcal{A}_{95}$	10	312	664	5153	2924	72 (1.40 %)	3399 (65.96 %)	27750	16.52	0.95
$\mathcal{A}_{96}$	8	237	427	3970	2382	94 (2.37 %)	2479 (62.44 %)	21761	16.75	0.58
$\mathcal{A}_{97}$	5	191	320	2684	1533	38 (1.42 %)	1713 (63.82 %)	14413	14.05	1.04
$\mathcal{A}_{98}$	10	321	665	5435	3154	72 (1.32 %)	3612 (66.46 %)	29785	16.93	1.23
$\mathcal{A}_{99}$	5	107	311	2600	1489	35 (1.35 %)	1689 (64.96 %)	13806	24.30	1.22
$\mathcal{A}_{100}$	14	484	988	8199	4516	104 (1.27 %)	5305 (64.70 %)	46164	16.94	1.20

Tabelle 24: Statistik des $\mathcal{K}_{D120}$

10 Experimente

In diesem Kapitel werden Experimente hinsichtlich sämtlicher Verfahren, die in dieser Arbeit vorgestellt wurden, beschrieben und durchgeführt. Die Basis für die Experimente bilden die dreizehn Feature-Kategorien F_1, F_2, ..., F_{13} aus Kapitel 5.5 sowie die sieben Korpora $\mathcal{K}_{KOM}$, $\mathcal{K}_{FAZ}$, ..., $\mathcal{K}_{D120}$ aus Kapitel 9.4. Um die Experimente thematisch abzugrenzen, werden diese wie folgt aufgeteilt: (1) *Autorschafts-Attribution*, (2) *Autorschafts-Verifikation*, (3) *Intrinsische Exploration* und schließlich (4) *Sonstige Experimente*.

Bevor die Experimentierreihe beginnt, wird zunächst eine kurze Einführung gegeben, die erläutert, welche Evaluierungsmaße sowie Evaluierungsstrategien für sämtliche Verfahren verwendet werden.

10.1 Eingesetzte Evaluierungsmaße & Evaluierungsstrategien

In diesem Abschnitt werden die für die Experimente verwendeten Evaluierungsmaße und Evaluierungsstrategien vorgestellt, welche in Kapitel 4.5.1 eingeführt wurden.

10.1.1 Evaluierungsmaße

- **Autorschafts-Attribution:** Für die Autorschafts-Attributions-Experimente wird die folgende *Accuracy*-Variante als Evaluierungsmaß gewählt:

$$Accuracy = \frac{\text{Anzahl aller tatsächlichen Autoren, die als solche korrekt vorhergesagt wurden}}{\text{Anzahl aller Autoren}}$$

  In einem binären Klassifikations-Szenario würde dies der Formel $\frac{TP}{TP+TN+FP+FN}$ entsprechen. Die Maßzahl TN (Anzahl der tatsächlich falschen Autoren, die auch als falsch klassifiziert wurden) beeinflußt nicht das Ergebniss, im Gegensatz zu der ursprünglichen Definition: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$.

- **Autorschafts-Verifikation:** Für diese Disziplin wird die ursprüngliche *Accuracy*-Variante mit $\frac{TP+TN}{TP+TN+FP+FN}$ als Evaluierungsmaß gewählt, da hier die TN benötigt werden. Im Kontext der Ein-Klassen-Klassifikation mit $\mathcal{A}_j$ als einzige Zielklasse, kann die Maßzahl TN wie folgt interpretiert werden: Anzahl der tatsächlichen $\neg\mathcal{A}_j$, die als solche und nicht fälschlicherweise als $\mathcal{A}_j$ erkannt wurden.

- **Intrinsische Exploration:** Als Evaluierungsmaß wird hier $F_1 - Meassure = \frac{2PR}{P+R}$ gewählt, da die intrinsische Exploration als eine typische *Information Retrieval* Aufgabe betrachtet werden kann und dieses Maß dafür einen De-facto-Standard darstellt. Die Semantik der Terme P und R innerhalb der Formel lautet:

  $P =$ Wie viele der gefundenen Dokumente sind relevant?

  $R =$ Wie viele der relevanten Dokumente wurden gefunden?

  Als relevant werden dabei solche Dokumente bezeichnet, die eine Stil-Inkonsistenz aufweisen. Dokumente deren Stil wiederum konsistent ist, sind dagegen irrelevant, da sie kein Postprocessing benötigen und direkt attribuiert oder verifiziert werden können. Für die ermittelten Dokumenten, die Stil-Inkonsistenzen enthalten, wird zudem das $Accuracy = \frac{TP}{TP+TN+FP+FN}$ Maß verwendet. Dieser beschreibt den Anteil der tatsächlichen Stil-Inkonsistenz, die auch als solche ermittelt werden konnte (TP), im Verhältnis zum Gesamtdokument.

Anmerkung: Alle Evaluierungsmaße werden in Prozentangaben ausgedrückt, um dadurch ein einheitliches Bild zu gewährleisten.

10.1.2 Evaluierungsstrategien

- *Leave-One-Out:* Diese Strategie wird für die Experimente der Attributions-Verfahren eingesetzt. Der Grund hierfür ist, dass dadurch kaum verzerrte Ergebnisse entstehen, da mit allen verfügbaren Dokumenten getestet wird (anders als bei der $k-$fold Cross-Validation Strategie).
- *$k-$fold Cross-Validation:* Diese Strategie wird für die Experimente der Verifikations-Verfahren eingesetzt. Hierbei wird $k = 3$ gewählt, da größere k den zeitlichen Rahmen[1] der Tests gesprengt hätten.

Anmerkung: Für die intrinsische Exploration wird keine Evaluierungsstrategie benötigt, da hier keine Trainingsmenge existiert. Die Textdokumente selbst (bzw. die Slices darin) können stattdessen als Trainingsmengen aufgefasst werden).

10.2 Autorschafts-Attribution

In diesem Abschnitt werden sämtliche Attributions-Verfahren evaluiert, die innerhalb dieser Arbeit vorgestellt und im Rahmen des Frameworks implementiert wurden. Zunächst werden Experimente anhand der drei Klassifikatoren aus Kapitel 4.3 durchgeführt, bevor im Anschluß die eigenen Verfahren evaluiert werden. Sowohl für die Klassifikatoren, als auch für die eigenen Verfahren werden alle vorgestellten Korpora und Feature-Kategorien eingesetzt. Die Ergebnisse werden dabei zum einem in tabellarischer und zum anderen in visueller Form angegeben. Aufgrund der hohen Dimensionen (13 Feature-Kategorien $\times$ 7 Korpora), werden jedoch nicht alle Ergebnisse hinsichtlich der Feature-Kategorien visualisiert, sondern nur die besten. Am Ende dieses Unterkapitels wird zudem ein Vergleich zwischen den zwei besten Klassifikatoren und dem besten eigenen Attributions-Verfahren gezogen.

10.2.1 ML-Verfahren: $k-$NN Klassifikator

In diesem Experiment wird der $k-$NN Klassifikator anhand von drei Metriken initialisiert und auf sämtliche Korpora angewendet. Bei den ersten zwei Metriken handelt es sich um die Distanzfunktionen $dist_{Euclid}(\cdot,\cdot)$ und $dist_{Cosine}(\cdot,\cdot)$, bei der dritten Metrik handelt es sich dagegen um die Ähnlichkeitsfunktion $sim_{Overlap}(\cdot,\cdot)$. Hierbei wurden jeweils alle Feature-Kategorien betrachtet und als Anzahl der nächsten Nachbarn $k = 5$ gewählt. Eine Ausnahme bildete hier $\mathcal{K}_{KOM}$, aufgrund der kleinen Autoranzahl, sodass für diesen Korpus stattdessen $k = 3$ gewählt wurde. Die folgende Tabelle zeigt zunächst die Ergebnisse hinsichtlich der drei Metriken:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	**50.00**	**48.75**	27.33	38.71	21.95	11.53	22.02	31.47
F_2	43.10	**53.75**	48.00	25.16	39.02	26.09	16.11	35.89
F_3	46.55	**77.50**	**72.00**	43.87	**49.76**	41.49	32.89	**52.01**
F_4	32.76	**50.00**	48.00	37.42	36.59	18.15	16.78	34.24
F_5	32.76	**48.75**	32.67	17.42	17.07	6.99	10.77	23.78
F_6	17.24	41.25	20.00	18.06	9.27	8.88	4.77	17.07
F_7	37.93	42.50	34.67	41.29	35.61	15.12	22.97	32.87
F_8	44.83	27.50	25.33	24.52	20.98	7.37	12.87	23.34
F_9	**48.28**	35.00	27.33	19.35	20.00	3.97	8.67	23.23
F_{10}	46.55	40.00	42.67	20.00	22.44	6.24	12.68	27.23
F_{11}	**55.17**	36.25	43.33	21.29	20.00	23.63	8.67	29.76
F_{12}	12.07	11.25	6.67	9.03	2.93	3.97	3.05	7.00
F_{13}	**48.28**	47.50	40.00	38.61	36.10	16.82	21.26	35.51

Tabelle 25: $k-$NN Klassifikation anhand der Euklid-Distanzfunktion.

[1] Für $k = 10$ dauerte ein Test beim $\mathcal{K}_{D120}$ beispielsweise 107.31 Stunden.

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	50.00	35.00	20.67	31.01	20.49	11.15	18.21	26.65
F_2	**51.72**	**53.75**	44.67	36.71	40.49	26.09	16.78	38.60
F_3	**51.72**	**78.75**	**68.67**	**51.90**	50.24	43.10	30.89	**53.61**
F_4	43.10	45.00	**54.67**	40.51	39.02	18.34	18.02	36.95
F_5	29.31	41.25	27.33	18.90	10.73	6.62	10.87	20.72
F_6	43.10	31.25	30.00	25.32	21.95	8.32	10.58	24.36
F_7	37.93	46.25	38.67	43.67	36.59	13.61	22.21	34.13
F_8	37.93	26.25	32.67	22.15	21.46	6.62	8.87	22.28
F_9	**55.17**	32.50	30.00	19.62	20.98	3.40	9.82	24.50
F_{10}	48.28	35.00	45.33	23.42	24.88	6.99	13.35	28.18
F_{11}	**55.17**	46.25	46.00	27.85	16.10	31.00	9.63	33.14
F_{12}	10.34	3.75	7.33	10.13	2.44	3.21	1.81	5.57
F_{13}	44.83	51.25	40.67	39.87	34.15	18.53	23.16	36.07

Tabelle 26: $k-$NN Klassifikation anhand der Manhattan-Distanzfunktion.

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	24.14	8.75	8.00	1.90	6.83	4.16	2.10	7.98
F_2	31.03	12.50	8.67	3.80	2.93	4.16	4.96	9.72
F_3	**65.52**	**82.50**	**69.33**	**53.80**	**49.76**	44.61	35.27	**57.26**
F_4	**53.45**	43.75	32.67	42.41	35.61	6.05	12.49	32.35
F_5	25.86	7.50	17.33	14.56	8.29	5.10	5.82	12.07
F_6	32.76	31.25	30.67	17.09	13.17	6.62	4.96	19.50
F_7	24.14	45.00	40.67	39.24	25.85	11.91	15.16	28.85
F_8	**46.55**	43.75	37.33	36.08	26.34	11.53	8.87	30.06
F_9	**46.55**	36.25	30.00	20.89	20.49	6.81	8.96	24.28
F_{10}	24.14	22.50	18.00	6.96	2.44	3.21	2.48	11.39
F_{11}	24.14	30.00	28.00	6.96	1.95	18.53	2.86	16.06
F_{12}	10.34	5.00	4.00	7.59	2.44	2.08	1.72	4.74
F_{13}	37.93	23.75	24.00	25.95	13.17	8.70	4.58	19.73

Tabelle 27: $k-$NN Klassifikation anhand der Overlap-Ähnlichkeitsfunktion.

Wie aus den Tabellen entnommen werden kann, liegen beim $k-$NN die Erkennungsgenauigkeiten generell im niedrigen Bereich. Der Grund hierfür sind Ausreißer innerhalb der Feature-Matrix, die eine Fehlklassifikation begünstigen (später mehr dazu).

Das beste Ergebniss innerhalb jeder Tabelle liefert offensichtlich die Feature-Kategorie F_3. Daher wird im Folgenden nur diese betrachtet, um weitere Erkenntnisse bzgl. der $k-$NN Klassifikation bestimmen zu können. Die folgende Abbildung verdeutlicht zunächst, welche der drei Metriken anhand von F_3 die höchsten Genauigkeiten für alle Korpora liefert:

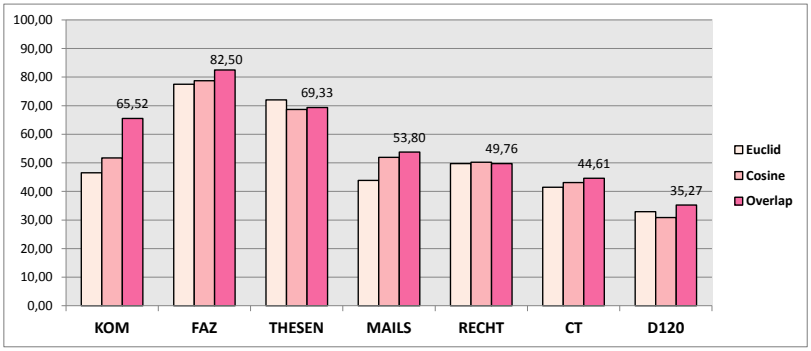


Abbildung 29: $k-$NN Klassifikationsergebnisse für die Feature-Kategorie F_3.

Es kann festgestellt werden, dass die Ähnlichkeitsfunktion $sim_{Overlap}(\cdot,\cdot)$ das höchste Ergebniss liefert (fünf aus sieben Korpora). Im folgenden Teilexperiment werden anhand dieser Metrik und die Feature-Kategorie F_3, Klassifikationsergebnisse für die $k-$Größen $\{5, 7, 9, 11, 13\}$ ermittelt. Das Ziel des

Teilexperiments ist es festzustellen, ob dadurch Verbesserungen erkennbar sind. Hierbei werden nur die Korpora $\mathcal{K}_{Thesen}$, $\mathcal{K}_{Mails}$, $\mathcal{K}_{Recht}$, $\mathcal{K}_{CT}$ und $\mathcal{K}_{D120}$ betrachtet, da hier für k ein größerer Spielraum vorhanden ist als bei $\mathcal{K}_{KOM}$ oder $\mathcal{K}_{FAZ}$. Die Ergebnisse für die unterschiedlichen $k-$Größen lauten wie folgt:

	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
$k = 5$	**69.33**	**53.80**	**49.76**	44.61	35.27	**50.55**
$k = 7$	**71.33**	**52.53**	**49.27**	43.48	36.13	**50.55**
$k = 9$	**71.33**	**51.27**	46.34	40.83	35.37	49.03
$k = 11$	**68.00**	48.73	44.88	38.19	37.18	47.40
$k = 13$	**69.33**	44.94	40.98	37.62	36.70	45.91

Tabelle 28: $k-$NN Klassifikationsergebnisse für $sim_{Overlap}(\cdot,\cdot)$, F_3 und $k \in \{5, 7, 9, 11, 13\}$.

Es kann hier beobachtet werden, dass größere k nicht zur Verbesserung, sondern eher zur Verschlechterung der Ergebnisse beitragen (zumindest bei drei aus fünf Korpora). Die folgende Abbildung verdeutlicht diese Feststellung:

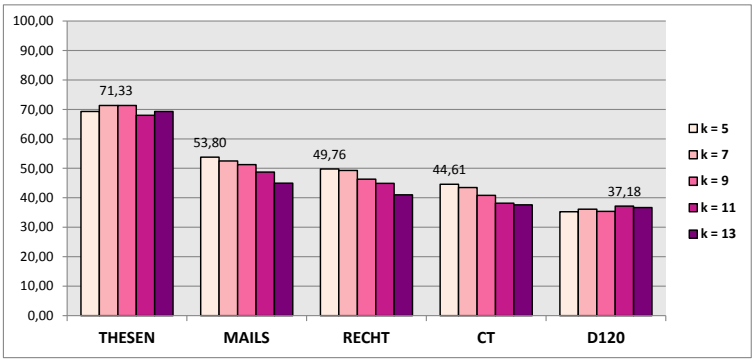


Abbildung 30: $k-$NN Klassifikationsergebnisse für $sim_{Overlap}(\cdot,\cdot)$, F_3 und $k \in \{5, 7, 9, 11, 13\}$.

In einem weiteren Teilexperiment wir gezeigt, wie die Klassifikationsergebnisse von $k-$NN durch eine Feature-Auswahl verbessert werden können. Dazu wird eine Feature-Auswahl basierend auf einer schrittweisen Rückwärtsselektion verwendet (siehe dazu Kapitel 5.8.1). Aufgrund der extrem langen Laufzeit die sich dadurch ergab, wurden für diesen Teilexperiment nur drei Korpora betrachtet. Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	∅
F_1	53.45	38.75	28.67	40.29
F_2	**62.07**	67.50	55.33	61.63
F_3	55.17	**88.75**	**77.33**	**73.75**
F_4	50.00	53.75	58.00	53.92
F_5	44.83	53.75	40.00	46.19
F_6	29.31	47.50	28.67	35.16
F_7	53.45	**65.00**	54.00	57.48
F_8	**60.34**	58.75	44.00	54.36
F_9	**63.79**	41.25	36.67	47.24
F_{10}	51.72	46.25	41.74	46.57
F_{11}	**60.34**	**61.25**	52.00	57.86
F_{12}	18.97	16.25	14.00	16.41
F_{13}	58.62	**62.50**	42.67	54.60

Tabelle 29: $k-$NN: Feature-Auswahl (schrittweise Rückwärtsselektion).

Das beste Ergebnis liefert auch hier wieder die Feature-Kategorie F_3. Allerdings ist dieses im Schnitt höher, als das des vorherigen Experiments anhand von $sim_{Overlap}(\cdot,\cdot)$ und $k = 5$.

Die folgende Tabelle zeigt den Vergleich des besten Ergebnis aus der Tabelle 27 und dem besten Ergebnis aus der Tabelle 30:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	∅
$\boldsymbol{k-NN}$ (ohne FSS)	65.52	**82.50**	69.33	72.45
$\boldsymbol{k-NN}$ (mit FSS)	55.17	**88.75**	77.33	**73.75**

Tabelle 30: $k-$NN: Feature-Auswahl (schrittweise Rückwärtsselektion).

Es kann hierbei beobachtet werden, dass für zwei aus drei Korpora die Ergebnisse für $k-$NN mit einer Feature-Selektion (engl. *Feature Subset Selection*, kurz FSS) höher ausfallen, gegenüber $k-$NN ohne FSS.

Anmerkung: FSS ist generell eine gute Idee, jedoch muss hier ein Trade-Off zwischen einer höheren Laufzeit und besseren Ergenissen eingegangen werden.

10.2.2 ML-Verfahren: NB Klassifikator

In diesem Experiment wird der Naive Bayes (NB) Klassifikator anhand aller Feature-Kategorien auf sämtliche Korpora angewendet. Die einzige Einstellung, die hier vorgenommen werden kann ist die Anwendung der Laplace-Glättung[1] hinsichtlich der bedingen Wahrscheinlichkeiten. Die folgende Tabelle zeigt zunächst die Ergebnisse:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
$\boldsymbol{F_1}$	53.45	50.00	36.00	40.65	28.78	16.82	30.98	36.67
$\boldsymbol{F_2}$	63.79	**65.00**	62.00	49.68	47.32	37.81	36.61	51.74
$\boldsymbol{F_3}$	**68.97**	**91.25**	**86.67**	**72.90**	**83.90**	**66.16**	55.67	**75.07**
$\boldsymbol{F_4}$	53.45	**75.00**	**80.67**	46.45	61.95	31.76	43.37	56.09
$\boldsymbol{F_5}$	32.76	46.25	41.33	19.35	18.05	10.21	11.25	25.60
$\boldsymbol{F_6}$	41.38	43.75	28.67	9.68	5.85	6.62	7.63	20.51
$\boldsymbol{F_7}$	58.62	55.00	59.33	43.87	44.39	19.85	34.03	45.01
$\boldsymbol{F_8}$	37.93	55.00	**64.00**	24.52	32.20	12.85	24.50	35.86
$\boldsymbol{F_9}$	34.48	41.25	37.33	10.32	12.68	6.81	6.39	21.32
$\boldsymbol{F_{10}}$	39.66	46.25	42.00	25.81	32.20	11.15	18.30	30.77
$\boldsymbol{F_{11}}$	51.72	62.50	61.33	40.65	31.71	35.92	18.88	43.24
$\boldsymbol{F_{12}}$	31.03	23.75	14.67	7.10	6.83	3.02	1.81	12.60
$\boldsymbol{F_{13}}$	51.72	48.75	59.33	39.24	37.07	19.09	31.17	40.91

Tabelle 31: NB Klassifikationsergebnisse.

Wie aus aus der Spalte mit den Durchschnittswerten ersichtlich ist, liefert NB deutlich höhere Ergebnisse als $k-$NN. Der Grund hierfür ist vermutlich, dass NB weniger anfällig gegenüber Ausreißern ist, sodass diese hier nicht ins Gewicht fallen.

Das beste Ergebniss hinsichtlich aller Feature-Kategorien stellt auch hier wieder F_3 dar. Interessant ist dabei die Tatsache, dass diese Kategorie für jeden Korpus das höchste Ergebniss darstellt, mit anderen Worten: Innerhalb eines jeden Korpus liefert F_3 die höchste Erkennungsgenauigkeit. Weitere Feature-Kategorien, die zumindest teilweise brauchbare Ergebnisse liefern, sind F_2 und F_4. In der folgenden Abbildung werden diese gegeneinander verglichen:

[1] Die Vorgehensweise dazu wurde in Kapitel 4.3.4 erläutert.

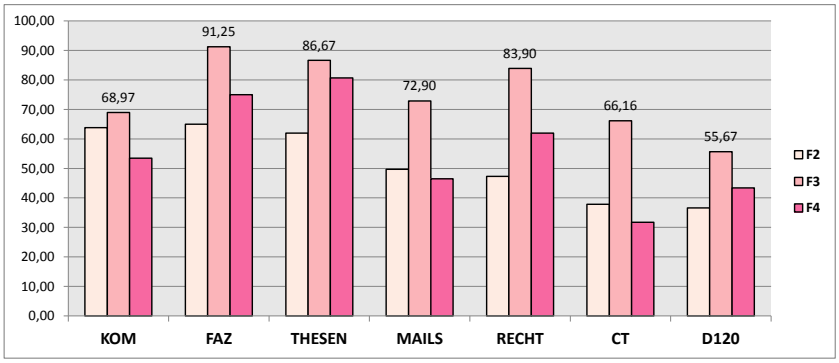


Abbildung 31: NB Klassifikationsergebnisse für die Feature-Kategorie F_2, F_3 und F_4.

Hieraus lassen sich die folgenden Erkenntnisse herleiten:

- NB liefert für die Korpora $\mathcal{K}_{KOM}$ und $\mathcal{K}_{FAZ}$ die höchsten Genauigkeiten und zwar für jede der drei Feature-Kategorien. Die Vermutung liegt nah, dass dies auf die Balanciertheit der beiden Korpora zurückzuführen ist.
- NB liefert für F_3 selbst für größere Korpora ($\geq$ 50 Autoren) stabile Ergebnisse, die oberhalb der 50-Prozent-Marke liegen. Alle anderen Feature-Kategorien liefern dagegen Ergebnisse, die teilweise weit unter der 50-Prozent-Marke liegen.

Ein andere interessante Erkenntniss, die weder aus der Tabelle noch aus dem Diagramm hervorgeht, betrifft die schnelle Laufzeit des NB-Klassifikators. Während der Evaluierungsphase konnte beobachtet werden, dass NB im Vergleich zu allen anderen Attributions-Verfahren in dieser Hinsicht unschlagbar ist. Eine weitere positive Eigenschaft von NB ist, dass dieses Verfahren parameterlos ist (mit Ausnahme der Laplace-Glättung). Dies hat den entscheidenden Vorteil, dass eine Suche nach einer optimalen Parametrisierung entfällt.

10.2.3 ML-Verfahren: SVM Klassifikator

In diesem Experiment werden zwei unterschiedliche Implementierungen des SVM Klassifikators anhand verschiedener Kernel auf sämtliche Korpora angewendet. Bei den Implementierungen handelt es sich zum einem um *LibSVM* und zum anderen um *mySVM*, die in RapidMiner enthalten sind. Um eine Multiklassen-Klassifikation ermöglichen zu können, wurde hier die `One-Against-All` Strategie eingesetzt. In den folgenden Tabellen werden die Resultate hinsichtlich dieser SVM-Klassifikatoren aufgelistet:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	**34.48**	0.00	0.00	13.92	3.90	6.99	12.77	10.29
F_2	**29.31**	0.00	0.00	6.96	3.41	4.54	4.48	6.96
F_3	**29.31**	0.00	0.00	6.96	4.39	4.73	4.48	7.12
F_4	27.59	0.00	0.00	6.96	3.90	4.54	4.48	6.78
F_5	**29.31**	0.00	0.00	6.96	2.44	4.54	7.34	7.23
F_6	27.59	22.50	**30.00**	13.92	8.29	11.15	15.92	18.48
F_7	**29.31**	0.00	0.00	6.96	3.90	4.54	10.68	7.91
F_8	**31.03**	0.00	6.00	10.76	6.83	5.86	10.68	10.17
F_9	**50.00**	0.00	10.00	12.66	5.37	5.29	11.53	13.55
F_{10}	**41.38**	0.00	2.67	7.59	7.80	4.91	12.39	10.96
F_{11}	**50.00**	18.75	**33.33**	9.49	5.85	14.56	10.20	**20.31**
F_{12}	**32.76**	15.00	3.33	7.59	3.90	5.29	5.91	10.54
F_{13}	**29.31**	0.00	0.00	9.49	4.39	5.10	12.49	8.68

Tabelle 32: SVM: *LibSVM* (Linearer Kernel).

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	**51.72**	15.00	10.67	12.90	11.25	6.81	16.11	**17.78**
F_2	**29.31**	0.00	0.00	6.45	4.88	4.91	4.86	7.20
F_3	**34.48**	0.00	14.00	7.10	6.83	12.85	14.59	12.84
F_4	**31.03**	6.25	2.00	5.81	3.90	4.91	9.06	8.99
F_5	**32.76**	0.00	8.00	9.03	3.90	4.91	11.06	9.95
F_6	**29.31**	7.50	4.00	5.81	0.49	7.37	6.67	8.74
F_7	**34.48**	0.00	0.00	5.16	5.85	7.55	13.63	9.52
F_8	24.14	5.00	14.67	11.61	6.83	4.91	6.96	10.59
F_9	**36.21**	10.00	28.00	10.32	4.88	4.91	12.39	15.24
F_{10}	**39.66**	7.50	23.33	5.70	5.85	6.05	14.87	14.71
F_{11}	**29.31**	13.75	14.67	12.03	6.34	7.18	7.91	13.03
F_{12}	**31.03**	1.25	0.67	7.59	1.95	5.11	4.67	7.47
F_{13}	25.86	7.50	6.67	7.59	6.83	8.13	14.68	11.04

Tabelle 33: SVM: *LibSVM* (Polynomieller Kernel).

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	**51.72**	15.00	10.67	12.90	11.25	6.81	16.11	17.78
F_1	41.38	25.00	55.00	6.96	2.93	2.27	1.43	19.28
F_2	51.72	55.00	19.33	8.23	7.80	3.21	2.19	21.07
F_3	82.76	**95.00**	**88.00**	**79.75**	**76.59**	54.25	41.36	**73.96**
F_4	**63.79**	**77.50**	**78.00**	56.33	54.63	18.53	6.17	50.71
F_5	39.66	23.75	15.33	10.76	5.37	3.78	1.43	14.30
F_6	36.21	57.50	44.67	18.35	20.49	5.67	0.92	30.48
F_7	51.72	32.50	10.00	8.23	3.90	2.65	1.43	15.78
F_8	44.83	63.75	58.67	43.67	37.07	13.23	3.42	37.81
F_9	44.83	21.25	24.67	9.49	4.39	1.89	2.19	15.53
F_{10}	46.55	42.25	36.67	24.42	24.39	5.48	2.19	25.99
F_{11}	51.72	63.75	56.00	23.42	14.63	11.09	0.43	31.58
F_{12}	32.76	22.50	12.67	5.70	5.37	3.59	3.21	12.26
F_{13}	56.90	37.50	18.67	4.43	4.43	2.27	0.86	20.70

Tabelle 34: SVM: *mySVM* (Linearer Kernel).

Die folgende Tabelle listet die besten drei Ergebnisse aus den oberen Tabellen auf:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
LibSVM (Poly.Kernel 3) F_1	**51.72**	15.00	10.67	12.90	11.25	6.81	16.11	17.78
LibSVM (Linearer Kernel) F_{11}	**50.00**	18.75	33.33	9.49	5.85	14.56	10.20	20.31
mySVM (Linearer Kernel) F_3	**82.76**	**95.00**	**88.00**	**79.75**	**76.59**	**54.25**	**41.36**	**73.96**

Tabelle 35: SVM: Klassifikationsergebnisse für die besten Feature-Kategorien (F_1, F_{11} und F_3).

Die besten drei Ergebnisse aus den Tabellen werden wie folgt visualisiert:

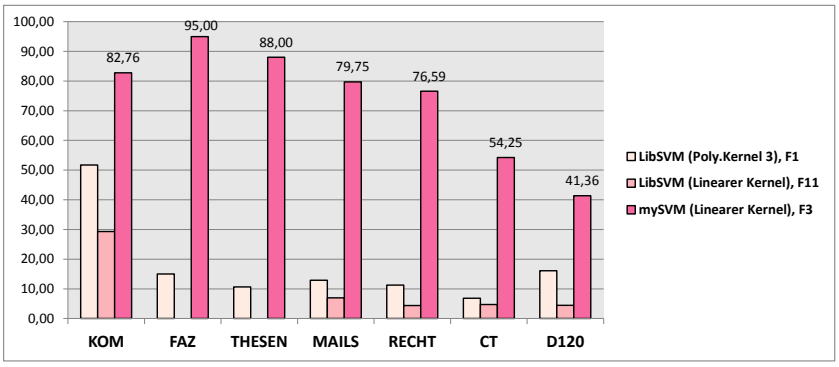


Abbildung 32: SVM: Klassifikationsergebnisse für die besten Feature-Kategorien (F_1, F_{11} und F_3).

Hieraus lassen sich die folgenden Erkenntnisse herleiten:

- Je nach eingesetzten Kernel kann der SVM Klassifikator gute Ergebnisse erzielen. Allerdings lassen sich die einzustellenden Parameter nicht bzw. nur schwer interpretieren. So war es z.B. nicht klar, wie der Parameter für die *Slack-Variables* gewählt werden sollte. Hierfür wurde die Voreinstellung mit 0 verwendet.
- Die besten Ergebnisse werden seitens der *mySVM* Implementierung (mit linearem Kernel) für die Korpora $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Thesen}$ erzielt. Als möglicher Grund wird hierbei die Balanciertheit dieser Korpora vermutet. Beim *mySVM* fällt auf, dass dieser nur für kleinere Korpora brauchbare Ergebnisse liefert. Für größere Korpora ($\geq$ 50 Autoren) klingen dagegen die Ergebnisse signifikant ab.
- Das schlechteste Ergebnis (welches dabei auch alle anderen Attributions-Verfahren einschließt) lieferte *LibSVM* (mit linearem Kernel). Teilweise konnten hierbei überhaupt keine Autoren richtig klassifiziert werden (z.B. bei $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Thesen}$). Dies ist dahingehend überraschend, da diese Korpora von den meisten Attributions-Verfahren am besten erkannt werden. Die Ursache für diese Ergebnisse können hierbei nicht begründet werden.

10.2.4 Pairwise Similarity Verfahren: Feature-Kategorien im Vergleich

In diesem Experiment wird das Pairwise Similarity Verfahren anhand jeder einzelnen Feature-Kategorie und der Distanzfunktion $dist_{Euclid}(\cdot,\cdot)$, auf sämtliche Korpora angewendet. Das primäre Ziel dieses Experiments ist die Feststellung, welche Feature-Kategorie die beste Erkennungsgenauigkeit für dieses Verfahren liefern kann. Die folgende Tabelle zeigt dazu die Ergebnisse:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
F_1	54.99	51.25	37.33	44.62	32.04	15.02	27.95	37.60
F_2	60.42	**66.25**	53.33	39.00	46.74	29.60	21.55	45.27
F_3	**77.50**	**85.00**	**79.33**	**69.98**	**77.42**	58.69	46.01	**70.56**
F_4	**64.40**	**62.50**	51.33	40.98	13.12	28.93	25.65	40.99
F_5	42.21	58.75	36.66	25.18	29.18	12.30	13.70	31.14
F_6	43.87	50.00	48.00	26.88	27.98	18.59	13.44	32.68
F_7	58.40	57.50	53.33	53.76	13.64	24.63	26.44	41.10
F_8	53.16	36.25	40.00	31.65	46.77	12.15	21.23	34.46
F_9	**61.81**	35.00	48.66	29.05	30.33	8.71	14.11	32.52
F_{10}	51.23	47.50	60.00	34.86	32.34	11.97	16.34	36.32
F_{11}	38.95	**65.00**	7.33	9.23	14.06	4.90	10.27	21.39
F_{12}	12.97	23.75	10.00	9.34	4.31	4.91	1.05	9.48
F_{13}	58.65	61.25	41.33	42.60	42.70	23.82	24.75	42.16

Tabelle 36: Pairwise Similarity Verfahren: Feature-Kategorien im Vergleich.

Aus der Tabelle kann entnommen werden, dass die Feature-Kategorie F_3 im Schnitt die einzige ist, die ein Attributions-Ergebniss jenseits der 50-Prozent-Marke erreicht. Die Kategorien F_2 und F_{13} liefern im Schnitt die nächsthöchsten Ergebnisse und werden in der folgenden Abbildung mitbetrachtet:

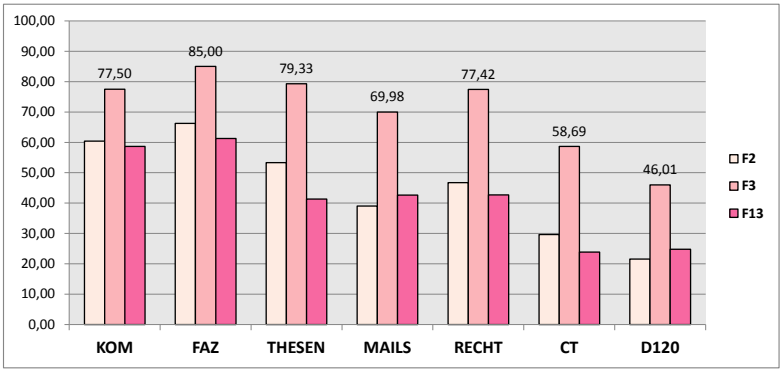


Abbildung 33: Pairwise Similarity Verfahren: Feature-Kategorien F_2, F_3 und F_{13} im Vergleich.

Hieraus wird deutlich, dass die Kategorien F_2 und F_{13} sich nur für kleinere Korpora eignen, um Ergebnisse oberhalb der 50-Prozent-Marke zu erreichen, während F_3 in der Lage ist für die ersten sechs Korpora Ergebnisse oberhalb dieser Marke zu liefern. Aus der Abbildung 10.2.4 kann weiterhin beobachtet werden, dass F_3 sich für die ersten Korpora stabil verhält, sodass hier keine größere Sprünge zwischen den Ergebnissen zu verzeichnen sind. Des Weiteren ist erkennbar, dass F_3 nur für die kleineren Korpora $\mathcal{K}_{KOM}$, $\mathcal{K}_{FAZ}$, $\mathcal{K}_{Thesen}$ und $\mathcal{K}_{Recht}$ brauchbare Ergebnisse ($> 75\,\%$) erzielt, während für größere Korpora (≥ 50 Autoren) die Erkennungsgenauigkeiten teils signifikant sinken.

10.2.5 Pairwise Similarity Verfahren: Kombinationen von Feature-Kategorien

In diesem Experiment wird das Pairwise Similarity Verfahren anhand verschiedener Kombinationen von Feature-Kategorien auf vier Korpora angewendet. Als Distanzfunktion wird auch hier die Metrik $dist_{Euclid}(\cdot,\cdot)$ verwendet. Das primäre Ziel dieses Experiments ist die Fragestellung, ob eine Kombination von Feature-Kategorien, gegenüber einzelnen Feature-Kategorien zu einem besseren Attributionsergebniss führen. Die verwendeten Kombinationen von Feature-Kategorien lauten:

$$\begin{aligned}
\textbf{\textit{Param}}\,\mathbf{1}&: \quad F_1 \cup F_2\\
\textbf{\textit{Param}}\,\mathbf{2}&: \quad F_5 \cup F_7 \cup F_8\\
\textbf{\textit{Param}}\,\mathbf{3}&: \quad F_2 \cup F_3 \cup F_7\\
\textbf{\textit{Param}}\,\mathbf{4}&: \quad F_3 \cup F_4 \cup F_7\\
\textbf{\textit{Param}}\,\mathbf{5}&: \quad F_1 \cup F_2 \cup F_3 \cup F_9
\end{aligned}$$

Die folgende Tabelle zeigt das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\varnothing$
$\textbf{\textit{Param}}_1$	**60.42**	**68.75**	53.33	44.05	56.64
$\textbf{\textit{Param}}_2$	**58.78**	**61.25**	58.00	37.05	53.77
$\textbf{\textit{Param}}_3$	**64.44**	**72.50**	58.67	49.11	**61.18**
$\textbf{\textit{Param}}_4$	**66.80**	**66.25**	55.33	52.57	60.24
$\textbf{\textit{Param}}_5$	**67.73**	56.25	**63.33**	43.97	57.82

Tabelle 37: Pairwise Similarity Verfahren: Kombinationen von Feature-Kategorien.

Aus der Tabelle kann entnommen werden, dass die Kombination von Feature-Kategorien nur beim $Param_1$ und $Param_2$ im Schnitt (gegenüber einzelnen) Feature-Kategorien zu einer Verbesserung der Ergebnisse führt. Die Kombinationen $Param\,3$, $Param\,4$ und $Param\,5$, die die Feature-Kategorie F_3 enthalten, führen allesamt zu einer Verschlechterung der Ergebnisse. Die folgende Abbildung verdeutlicht dies:

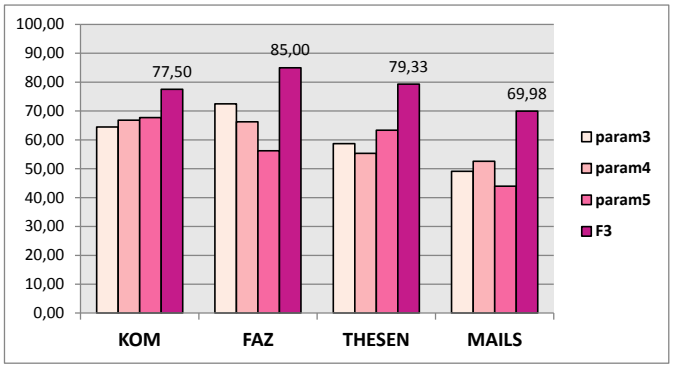


Abbildung 34: Pairwise Similarity Verfahren: Vergleich zwischen $Param\,3-5$ und F_3.

Als Grund für die Verschlechterung wird zusätzliches Rauschen vermutet, welches durch die Feature-Kategorien F_2 und F_7 verursacht wird.

10.2.6 Pairwise Similarity Verfahren: Metriken im Vergleich

In diesem Experiment wird das Pairwise Similarity Verfahren anhand von fünf verschiedenen Metriken, auf kleinere Korpora ($\leq$ 32 Autoren) angewendet. Für alle Metriken werden dabei die gleichen Feature-Kategorien (F_1, F_2, F_7 und F_{13}) verwendet. Ziel des Experiments ist die Feststellung, wie sich unterschiedliche Metriken zueinander verhalten und ob diese robuste[1] Ergebnisse über Korpora hinweg liefern. Die folgende Tabelle zeigt das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\varnothing$
$dist_{Euclid}(\cdot,\cdot)$	**58.65**	**63.75**	46.67	42.08	43.67	50.96
$dist_{Canberra}(\cdot,\cdot)$	41.54	**55.00**	**80.00**	**60.97**	41.12	**55.73**
$sim_{Overlap}(\cdot,\cdot)$	44.23	**51.25**	41.33	35.44	26.95	39.84
$sim_{Jaccard}(\cdot,\cdot)$	**62.29**	**60.00**	48.67	42.61	44.79	51.67
$sim_{Cosine}(\cdot,\cdot)$	**63.47**	**62.50**	50.67	42.73	46.50	53.17

Tabelle 38: Pairwise Similarity Verfahren: Metriken im Vergleich.

Aus der Spalte mit den Durchschnittswerten kann entnommen werden, dass der Großteil der Metriken ähnliche Ergebnisse liefert, die zudem oberhalb der 50-Prozent-Marke liegen. Einzig die Ähnlichkeitsfunktion $sim_{Overlap}(\cdot,\cdot)$ liefert im Vergleich das schlechteste Ergebniss, hinsichtlich der Erkennungsgenauigkeit. Aus der folgenden Abbildung lassen sich zudem weitere Erkenntnisse gewinnen:

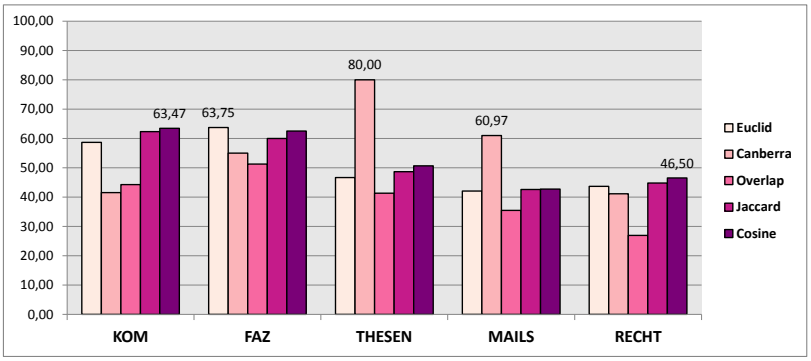


Abbildung 35: Pairwise Similarity Verfahren: Metriken im Vergleich.

Hieraus lassen sich die folgenden Erkenntnisse ableiten:

- $\mathcal{K}_{FAZ}$ ist der einzige Korpus bei dem alle Metriken Ergebnisse $\geq$ 50 % erzielen konnten.
- Die Euclid-Distanzfunktion liefert nahezu für jeden Korpus annähernd gleiche Ergebnisse, wie die beiden Ähnlichkeitsfunktionen Jaccard und Cosine. Diese drei Metriken können als robust betrachtet werden, da von Korpus zu Korpus keine größeren Sprünge festzustellen sind.
- Jaccard und Cosine verhalten sich sehr ähnlich zueinander, allerdings liefert Cosine stets (minimal) bessere Werte gegenüber Jaccard.
- Die Canberra-Distanzfunktion liefert für zwei Korpora signifikant hohe Ergebnisse (im Verhältnis zu allen anderen Metriken). Eine genauere Begründung hierfür konnte jedoch nicht ermittelt werden.

10.2.7 Pairwise Similarity Verfahren: Minkowski-Distanzfunktionen im Vergleich

In diesem Experiment wird das Pairwise Similarity Verfahren anhand der Feature-Kategorie F_{13} und der Distanzfunktion $dist_{Minkowski}(\cdot,\cdot)$ mit unterschiedlichen $\lambda-$Größen, auf insgesamt fünf Korpora angewendet. Das Ziel dieses Experiments ist die Feststellung, welches λ das bestmögliche Ergebnis liefert. Die folgende Tabelle zeigt dazu die Ergebnisse:

[1] Eine Metrik wird als *robust* bezeichnet, falls ihre Anwendung auf mehrere Korpora zu keinen (größeren) Sprüngen führt.

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	∅
$\lambda = 2$	**58.65**	**61.25**	41.33	42.60	42.70	49.31
$\lambda = 1.5$	**60.47**	**63.75**	46.00	44.26	44.64	51.82
$\lambda = 1$	**64.43**	**65.00**	48.67	51.71	50.52	56.07
$\lambda = 0.5$	**70.24**	**66.25**	57.33	55.66	56.73	**61.24**
$\lambda = 0.25$	**71.69**	**60.00**	54.67	54.56	57.04	59.59
$\lambda = 0.125$	53.92	47.50	51.33	43.57	42.04	47.67

Tabelle 39: Pairwise Similarity Verfahren: Minkowski-Distanzfunktionen im Vergleich.

Aus der Tabelle wird zunächst ersichtlich, dass die höchsten Ergebnisse für die Korpora $\mathcal{K}_{KOM}$ und $\mathcal{K}_{FAZ}$ erzielt werden. Des Weiteren kann hier beobachtet werden, dass $\lambda = 0.5$ im Schnitt das beste Ergebniss für alle Korpora liefert. In der Spalte mit den Durchschnittswerten kann zudem ein linearer Anstieg, hinsichtlich der ersten vier λ beobachtet werden. Die folgende Abbildung verdeutlicht dies genauer:

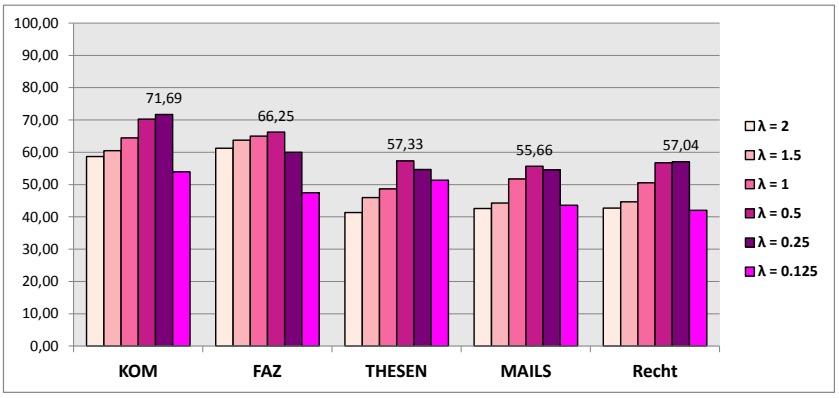


Abbildung 36: Pairwise Similarity Verfahren: Minkowski-Distanzfunktionen im Vergleich.

Hieraus wird deutlich, dass mit abfallenden $\lambda-$Größen die Attributions-Genauigkeiten ansteigen, allerdings nur bis $\lambda = 0.5$. Für $\lambda < 0.5$ sinken wiederum die Genauigkeiten bei drei aus fünf Korpora (teilweise sogar signifikant für $\lambda = 0.125$). Zusammenfassend lässt sich hieraus folgern, dass die beste Wahl auf den Parameter $\lambda = 0.5$ fällt.

10.2.8 Fragmentwise Intersection Verfahren: Variable $n-$Gramm Größen

In diesem Experiment wird das Fragmentwise Intersection Verfahren mit unterschiedlichen $n-$Gramm Größen initialisiert und auf sämtliche Korpora angewendet. Hierbei werden jeweils die 100 häufigsten $n-$Gramme aus der reinen Textform[1] der Dokumente entnommem. Als mengenbasierte Distanzfunktion wird $dist_{Dice}(\cdot,\cdot)$ eingesetzt. Ziel des Experiments ist die Erkennung, wie sich zunehmende $n-$Gramm Größen auf die Attributions-Genauigkeiten auswirken. Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
$n = 2$	41.52	43.75	66.00	32.65	39.52	30.61	14.82	38.41
$n = 3$	64.70	**77.50**	**66.66**	52.23	60.02	44.55	22.19	55.41
$n = 4$	60.45	**77.50**	**77.33**	58.06	64.23	49.56	30.30	59.63
$n = 5$	56.25	**77.50**	**71.33**	60.54	**67.28**	54.09	31.29	59.75
$n = 6$	59.87	**86.25**	**75.33**	65.07	**69.86**	55.73	33.74	**63.69**

Tabelle 40: Fragmentwise Intersection Verfahren: Variable $n-$Gramm Größen.

Zunächst geht aus der Tabelle hervor, dass $\mathcal{K}_{KOM}$ und $\mathcal{K}_{FAZ}$ im Vergleich zu den anderen Korpora, am besten abschneiden. Des Weiteren kann in der Spalte mit den Durchschnittswerten eindeutig

[1] Eine Text-Normalisierung wurde dabei nicht durchgeführt.

festgestellt werden, dass zunehmende $n-$Gramm Größen die Attributions-Genauigkeiten verbessern. Das folgende Diagramm verdeutlicht dies genauer:

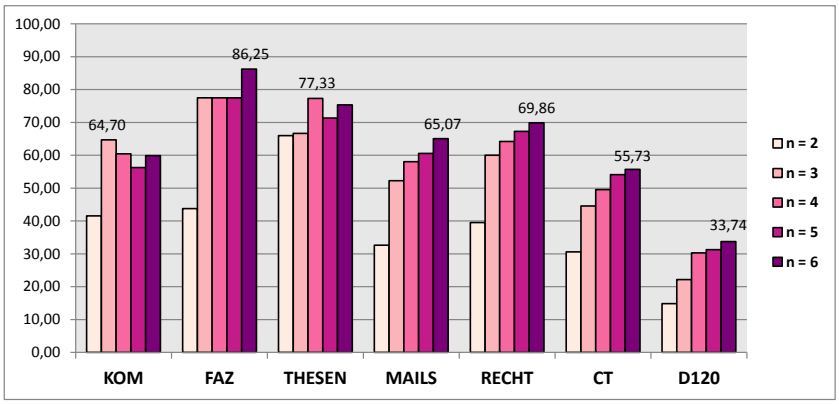


Abbildung 37: Fragmentwise Intersection Verfahren: Variable $n-$Gramm Größen.

Aus dieser Abbildung lassen sich die folgenden Erkenntnisse ableiten:

- Für $n = 6$ können bei fünf aus sieben Korpora, die besten Ergebnisse festgestellt werden.
- Für $n = 2$ werden wiederum die schlechtesten Ergebnisse (bei allen Korpora) erzielt.
- Bei Korpora mit einer Größenordnung ab 26 Autoren, kann eine lineare Steigerung der Genauigkeiten hinsichtlich der $n-$Gramm Größen festgestellt werden. Eine genauere Erklärung für dieses Phänomen kann jedoch nicht gegeben werden.

10.2.9 Fragmentwise Intersection Verfahren: Ähnlichkeitsfunktionen im Vergleich

In diesem Experiment wird das Fragmentwise Intersection Verfahren anhand von unterschiedlichen Ähnlichkeitsfunktionen auf sämtlichen Korpora angewendet. Hierbei werden jeweils die 100 häufigsten 7−Gramme aus der reinen Textform der Dokumente entnommem. Als mengenbasierte Ähnlichkeitsfunktionen dienen $sim_{Overlap}(\cdot,\cdot)$, $sim_{Cosine}(\cdot,\cdot)$, $sim_{Jaccard}(\cdot,\cdot)$ sowie $sim_{Dice}(\cdot,\cdot)$. Das Ziel dieses Experiments ist es festzustellen, in wie weit unterschiedliche Ähnlichkeitsfunktionen das Ergebnis beeinflussen. Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
$sim_{Overlap}(\cdot,\cdot)$	60.25	**81.25**	**70.67**	59.26	**70.95**	55.20	34.26	**61.69**
$sim_{Cosine}(\cdot,\cdot)$	60.25	**81.25**	**70.67**	59.26	**70.95**	55.20	34.26	**61.69**
$sim_{Jaccard}(\cdot,\cdot)$	60.25	**81.25**	**70.67**	59.26	**70.95**	55.20	34.26	**61.69**
$sim_{Dice}(\cdot,\cdot)$	60.25	**81.25**	**70.67**	59.26	**70.95**	55.20	34.26	**61.69**

Tabelle 41: Fragmentwise Intersection Verfahren: Ähnlichkeitsfunktionen im Vergleich.

Aus der Tabelle geht zunächst eindeutig hervor, dass es offensichtlich irrelevant ist, welche mengenbasierte Ähnlichkeitsfunktion verwendet wird, da für jede Funktion stets die gleichen Ergebnisse geliefert werden. Von allen Korpora liefert dieses Verfahren das beste Ergebnis für $\mathcal{K}_{FAZ}$. Eine weitere Erkenntnis, die aus dieser Tabelle sowie der Tabelle 10.3.2 entnommen werden kann ist, dass 7−Gramme im Schnitt schlechter abschneiden als 6−Gramme. Die folgende Abbildung zeigt dies anhand eines direkten Vergleichs:

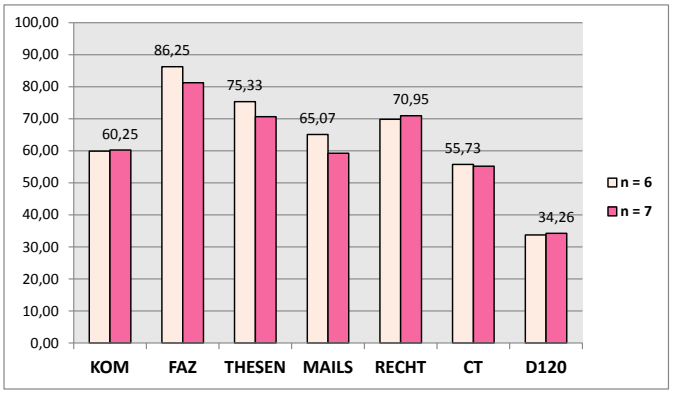


Abbildung 38: Fragmentwise Intersection Verfahren: Variable $n-$Gramm Größen.

Für vier aus sieben Korpora liefern 6−Gramme im Schnitt 2 % höhere Attributions-Genauigkeiten gegenüber 7−Gramme.

10.2.10 Fragmentwise Intersection Verfahren: Vereinigung variabler Textfragmente

In diesem Experiment wird das Fragmentwise Intersection Verfahren mit unterschiedlichen Textfragmenten initialisiert und auf sämtliche Korpora angewendet. Als Textfragmente werden die Folgenden ausgewählt:

***Param* 1**: Top(30, `POS-Tag 1-Gramme`) ∪ Top(40, `Buchstaben 2-Gramme`) ∪ Top(100, `Buchstaben 6-Gramme`)

***Param* 2**: Top(30, `POS-Tag 2-Gramme`) ∪ Top(50, `Buchstaben 4-Gramme`) ∪ Top(20, `Token 2-Gramme`)

***Param* 3**: Top(35, `POS-Tag 3-Gramme`) ∪ Top(70, `Buchstaben 3-Gramme`) ∪ Top(80, `Buchstaben 7-Gramme`) ∪ Top(40, `Token 2-Gramme`)

Hierbei repräsentiert die Funktion `Top(k,Textfragment)` die $k-$häufigsten Textfragmente. Als mengenbasierte Distanzfunktion kam $dist_{Dice}(\cdot,\cdot)$ zum Einsatz. Ziel dieses Experiments ist die Feststellung, welche Auswirkungen unterschiedliche Textfragmente auf die Attributions-Genauigkeiten des Verfahrens haben. Die folgende Tabelle zeigt zunächst das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
***Param* 1**:	**65.71**	**88.75**	**76.66**	**63.79**	**71.11**	57.03	37.00	65.72
***Param* 2**:	**69.26**	**88.75**	**80.00**	**67.50**	**77.29**	59.31	38.73	**68,69**
***Param* 3**:	60.53	56.25	60.00	48.42	58.67	41.10	21.44	49.49

Tabelle 42: Fragmentwise Intersection Verfahren: Vereinigung variabler Textfragmente.

Aus der Tabelle kann zunächst entnommen werden, dass $Param1$ und $Param2$ die höchsten Ergebnisse liefern, wobei $Param2$ gegenüber $Param1$ im Schnitt überlegen ist. Das folgende Diagramm zeigt den Vergleich anhand der drei unterschiedlichen Vereinigungen:

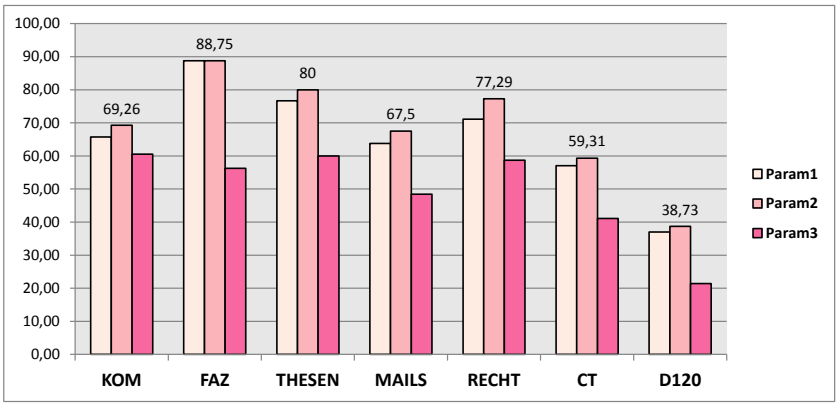


Abbildung 39: Fragmentwise Intersection Verfahren: Vereinigung unterschiedlicher Textfragmente.

Daraus lassen sich mehrere Erkenntnisse herleiten:

- *Param*2 liefert für sechs aus sieben Korpora höhere Ergebnisse als die anderen beiden. Interessant ist dabei die Tatsache, dass hier (in der Summe gesehen) nur 100 Features vorliegen, während bei *Param*1 170 und bei *Param*3 225 Features verwendet werden. Es kann daraus gefolgert werden, dass mehr Features zu niedrigen Erkennungsgenauigkeiten führen.
- Da *Param*2 das beste Ergebniss wiederspiegelt, kann angenommen werden, dass eine Attribution anhand von Kollokationen (häufige Token Bigramme) nicht nur funktioniert, sondern dazu noch Token Unigramme übertrifft. Diese Feststellung steht im Wiederspruch zu manchen Ergebnissen in der Literarur, beispielsweise in [94]. Eine weitere interessante Beobachtung ist, dass sowohl 2−Gramme als auch 4−Gramme bei vorherigen Experimenten (siehe Tabelle 40 sowie Tabelle 41) schlechter abgeschnitten haben, als 6−Gramme und 7−Gramme, während beim *Param*2 genau das Gegenteil der Fall ist.
- *Param*2 liefert gegenüber den vorherigen Experimenten hinsichtlich des Fragmentwise Intersection Verfahrens, die besten (korpusübergreifenden) Ergebnissen. Daraus kann gefolgert werden, dass eine Vereingung von Textfragmenten (insbesondere solche, die unterschiedliche n−Gramm Größen enthalten) sich besser eignet, als n−Gramme für ein fest gewähltes n.
- Die beiden Korpora $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Thesen}$ werden (wie schon bei einigen vorherigen Experimenten) am besten erkannt. Auch hier wird dies auf die Balanciertheit beider Korpora zurückgeführt.

10.2.11 Two-Stage Pairwise Similarity Verfahren: Feature-Kategorien im Vergleich

In diesem Experiment wird das Two-Stage Pairwise Similarity Verfahren anhand jeder einzelnen Feature-Kategorie und der folgenden Parametrisierung auf sämtliche Korpora angewendet:

	***Param*1**
Stufe: 1, n−Gramm Größe:	7
Stufe: 1, k−häufigsten n−Gramme:	120
Stufe: 1, Ähnlichkeitsfunktion:	$sim_{Jaccard}(\cdot,\cdot)$
Stufe: 1, `SPLIT`:	20 % (bei $\mathcal{K}_{KOM}$: 60 %)
Stufe: 2, Feature-Kategorien:	Alle (siehe unten)
Stufe: 2, Distanzfunktion:	$dist_{Euclid}(\cdot,\cdot)$

Tabelle 43: Two-Stage Pairwise Similarity Verfahren: Parametrisierung.

Das Ziel dieses Experiments ist es zunächst eine Übersicht zu geben, wie die einzelnen Feature-Kategorien abschneiden. Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
F_1	70.58	**88.75**	69.33	73.84	70.87	37.95	47.79	65.59
F_2	75.44	**87.50**	75.33	69.86	78.77	57.75	47.92	70.37
F_3	**88.50**	**91.25**	**89.33**	82.54	**89.22**	78.67	70.21	**84.25**
F_4	76.12	**90.00**	81.33	75.47	84.73	60.91	54.29	74.69
F_5	66.20	**87.50**	73.33	68.66	84.73	45.25	39.07	66.39
F_6	**91.94**	**91.25**	82.67	76.72	84.97	65.99	66.04	79.94
F_7	71.93	81.25	78.67	74.28	78.40	54.00	51.34	69.98
F_8	74.09	86.25	78.00	66.70	70.00	49.58	41.15	66.54
F_9	82.55	**87.50**	82.67	73.71	78.40	30.09	52.48	69.63
F_{10}	72.67	86.25	85.33	74.45	75.14	54.89	47.12	70.84
F_{11}	38.72	43.75	34.67	21.06	27.08	9.43	5.70	25.77
F_{12}	40.61	72.50	53.33	39.92	46.63	24.22	18.58	42.26
F_{13}	73.38	**88.75**	72.67	65.05	71.57	51.38	43.56	66.62

Tabelle 44: Two-Stage Pairwise Similarity Verfahren: Feature-Kategorien im Vergleich.

Die Ergebnisse in der Tabelle zählen zu den höchsten, von allen bisher getesteten Attributions-Verfahren. Wie in den meisten vorherigen Experimenten stellt auch hier wieder die Feature-Kategorie

F_3 im Schnitt die beste Autorschafts-Genauigkeit dar. Im folgenden Teilexperiment wird versucht, dieses Ergebnis zu übertreffen. Hierfür wird die Minkowski-Distanzfunktion mit $\lambda = 0.5$ verwendet, da in einem vorherigen Experiment (siehe dazu Tabelle 39) gezeigt wurde, dass diese die euklidische Distanzfunktion (bzw. die Minkowski Funktion mit $\lambda = 2$) übertrifft. Alle anderen Einstellungen werden dagegen beibehalten. Das Resultat lautet wie folgt:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
***Minkowski*(0.5)**	**93.76**	**91.25**	88.67	84.51	89.55	74.72	60.26	83.25
***Minkowski*(2)**	88.50	**91.25**	89.33	82.54	89.22	78.67	70.21	**84.25**

Tabelle 45: Two-Stage Pairwise Similarity Verfahren: Vergleich von Minkowski-Distanzfunktionen.

Die Erkenntniss, die hieraus gewonnen werden kann ist, dass die Minkowski-Distanzfunktion anhand von $\lambda = 2$ gegenüber $\lambda = 0.5$ ein minimal besseres Ergebniss liefern kann. Anhand der folgenden Abbildung wird der Unterschied hinsichtlich des λ Parameters besser verdeutlicht:

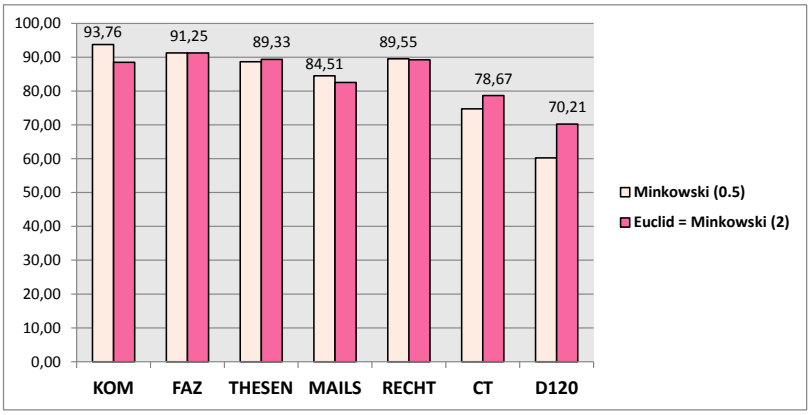


Abbildung 40: Two-Stage Pairwise Similarity Verfahren: Vergleich von Minkowski-Distanzfunktionen.

Für drei aus sieben Korpora liefert die Minkowski-Distanzfunktion mit $\lambda = 0.5$ das bessere Ergebniss. Allerdings sind die Unterschiede zum einen minimal und zum anderen betreffen diese nur kleinere Korpora (≤ 32 Autoren). Die Minkowski-Distanzfunktion mit $\lambda = 2$ weist dagegen höhere Unterschiede hinsichtlich der Ergebnisse auf und betrifft dabei die zwei größten Korpora (≥ 50 Autoren). Daher kann gefolgert werden, dass diese die bessere Wahl darstellt.

10.2.12 Two-Stage Pairwise Similarity Verfahren: Bi-/Trigramme im Vergleich

In diesem Experiment wird das Two-Stage Pairwise Similarity Verfahren mit der selben Parametrisierung aus dem vorherigen Experiment (siehe Tabelle 43) auf sämtliche Korpora angewendet. Der einzige Unterschied hierbei ist, dass die Feature-Kategorie F_3 um Trigramme erweitert wird und somit nicht nur aus Bigrammen besteht. Das Ziel dieses Experiments ist es festzustellen, ob die zusätzlichen Trigrammen zu einer Verbesserung der Ergebnisse beitragen. Die folgende Tabelle zeigt dazu die Anwendung des Verfahrens anhand von F_3 zunächst nur mit Bigrammen und anschließend mit Bi- und Trigrammen:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
$\boldsymbol{F_3}$ (mit Bigramme)	88.50	**91.25**	89.33	82.54	89.22	78.67	70.21	84.25
$\boldsymbol{F_3}$ (mit Bi-/Trigramme)	**90.84**	**91.25**	89.33	85.49	**90.70**	80.98	73.62	**86.03**

Tabelle 46: Two-Stage Pairwise Similarity Verfahren: Bi-/Trigramme im Vergleich.

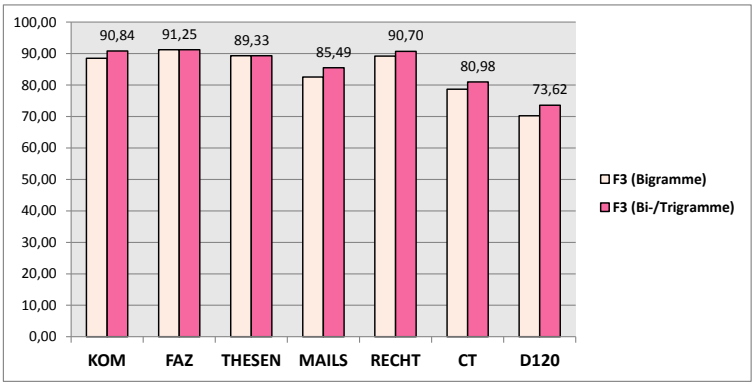


Abbildung 41: Two-Stage Pairwise Similarity Verfahren: Bi-/Trigramme im Vergleich.

Aus der Tabelle und der Abbildung kann beobachtet werden, dass in fünf aus sieben Fällen die Ergebnisse von F_3 mit Bigrammen gegenüber F_3 mit Bi- und Trigrammen höher ausfallen (wenn auch nur mimimal). Im Schnitt ist eine Verbesserung von 1.78 % zu erkennen, was zunächst sinnvoll erscheint. Ein Problem, was jedoch weder aus der Tabelle noch aus der Abbildung hervorgeht, betrifft die Laufzeit. Im ersten Fall werden 1312 Features verwendet, beim zweiten Fall dagegen insgesamt 3698 Features (und damit mehr als das Doppelte), was sich dementsprechend negativ auf die Laufzeit des Verfahrens auswirkt. Beim zweiten Fall dauerte der Test deutlich länger (ca. 8 Stunden mehr) und lieferte dabei keine signifikante Verbesserung. Somit stellt sich die Frage, ob hier ein Kompromiss zwischen Laufzeit und minimaler Verbesserung bzgl. der Erkennungsgenauigkeiten sinnvoll erscheint.

10.2.13 Two-Stage Pairwise Similarity Verfahren: Fünf Parameter im Vergleich

In diesem Experiment wird das Two-Stage Pairwise Similarity Verfahren anhand von fünf Parametern auf sämtliche Korpora angewendet. Die Parameter sind hierbei nach den zwei Stufen des Verfahrens wie folgt aufgeteilt:

	***Param*1**	***Param*2**	***Param*3**	***Param*4**	***Param*5**
Stufe: 1, $n-$Gramm Größe:	2	6	7	3	5
Stufe: 1, $k-$häufigsten $n-$Gramme:	60	100	90	160	120
Stufe: 1, Ähnlichkeitsfunktion:	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Dice}(\cdot,\cdot)$	$sim_{Overlap}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$
Stufe: 1, SPLIT:	30 %	50 %	20 %	40 %	30 %
Stufe: 2, Feature-Kategorien:	F_1, F_2, F_5, F_9	F_1	F_2, F_7, F_{13}	F_2, F_4, F_5, F_8	F_1, F_2, F_3, F_4
Stufe: 2, Distanzfunktion:	$dist_{Tscheby}(\cdot,\cdot)$	$dist_{Euclid}(\cdot,\cdot)$	$sim_{Cosine}(\cdot,\cdot)$	$dist_{Euclid}(\cdot,\cdot)$	$dist_{Manhattan}(\cdot,\cdot)$

Tabelle 47: Two-Stage Pairwise Similarity Verfahren: Parametrisierung.

Das Ziel dieses Experiments ist es festzustellen, ob die Ergebnisse des vorherigen Experiments übertroffen werden können. Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
***Param*1**	41.89	50.00	61.33	47.14	50.79	21.97	21.20	42,05
***Param*2**	75.06	73.75	55.33	62.80	59.51	28.65	47.22	57.47
***Param*3**	56.85	**85.00**	75.33	65.49	71.27	54.31	45.84	64.87
***Param*4**	64.94	**92.5**	**86.00**	78.30	**89.33**	71.56	63.42	78.01
***Param*5**	77.42	**87.5**	**90.66**	**91.98**	**94.71**	**84.33**	**88.99**	**87.94**

Tabelle 48: Two-Stage Pairwise Similarity Verfahren: Fünf Parameter im Vergleich.

Bei den fünf Parametrisierungen fällt vor allem *Param*5 auf, welches für jeden Korpus brauchbare Ergebnisse ($> 75\%$) liefert, wozu bisher keines der getesteten Attributions-Verfahren in der Lage war. Die folgende Abbildung verdeutlicht die Ergebnisse aus der Tabelle 60.

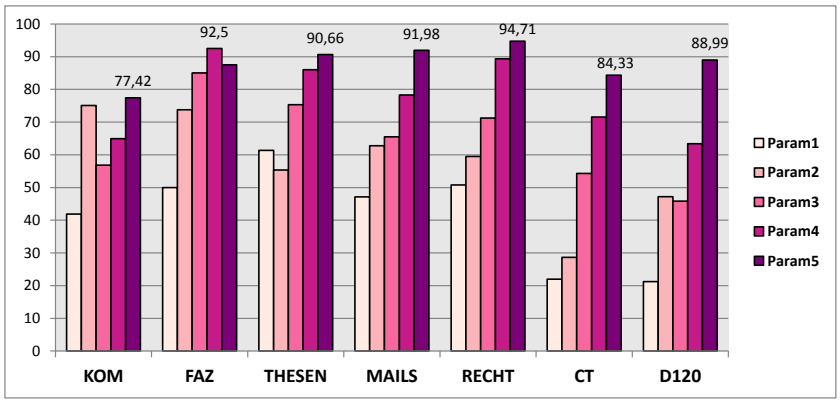


Abbildung 42: Two-Stage Pairwise Similarity Verfahren: Fünf Parameter im Vergleich.

Hieraus lassen sich die folgenden Erkenntnisse herleiten:

- Wie aus Tabelle 60 und Abbildung 10.2.13 ersichtlich ist, liefert $Param5$ das beste Ergebniss. Als wichtigster Grund wird hierfür die in $Param5$ enthaltene Feature-Kategorie F_3 vermutet, welche in (nahezu) jedem vorherigen Experiment, die diskriminierendste Feature-Kategorie darstellte. Die übrigen Einstellungen in $Param5$ fallen auch ins Gewicht, jedoch nicht so stark. Die Ähnlichkeitsfunktion $sim_{Jaccard}(\cdot,\cdot)$ dürfte hier ebensowenig Einfluss haben, wie der Parameter k (Anzahl der häufigsten $n-$Gramme). Die Einstellung, die die zweitwichtigste Rolle beim $Param5$ einnimmt, ist `SPLIT` mit 30 %. Hierdurch verringert sich der Umfang hinsichtlich der möglichen Autoren, sodass in der zweiten Stufe des Verfahrens weniger Fehlentscheidungen möglich sind.

- Das schlechteste Ergebnis in diesem Experiment wird anhand von $Param1$ erzielt. Als Ursache hierfür wird die Tschebyscheff-Distanzfunktion vermutet. Diese berechnet für je zwei Feature-Vektoren die maximale Distanz hinsichtlich der Merkmale darin, sodass alle anderen Distanzen das Ergebniss (und damit die Stilunterscheidung) nicht beeinflussen. Es wird daher festgehalten, dass sich diese Distanzfunktion nicht für eine Autorschafts-Attribution eignet (zumindest nicht anhand des eingesetzten Verfahrens).

- Es fällt auf, dass das Verfahren, selbst für Korpora mit bis zu 100 möglichen Autoren, ebenfalls gute Ergebnisse liefern kann. Der Grund hierfür ist die Vorfilterung von irrelevanten Autoren. Der Wert 20 % bzw. auch 30 % für den Parameter `SPLIT` scheint hier eine gute Wahl zu sein.

Im vorherigen Experiment wurde gezeigt, dass die euklidische Distanz (Minkowski Distanzfunktion mit $\lambda = 2$) im Schnitt das bisher beste Ergebnis erzielt hatte (siehe dazu Tabelle 45). Im folgenden Teilexperiment wird ein Vergleich von diesem, mit dem vom $Param5$ durchgeführt. Das Resultat dazu lautet zunächst:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
Minkowski(2)	88.50	**91.25**	89.33	82.54	89.22	78.67	70.21	84.25
Param5	77.42	**87.5**	**90.66**	**91.98**	**94.71**	**84.33**	**88.99**	**87.94**

Tabelle 49: Two-Stage Pairwise Similarity Verfahren: Vergleich von $Param5$ und dem besten Ergebnis aus der Tabelle 45.

Aus der Spalte mit den Durchschnittswerten geht eindeutig hervor, dass $Param5$ im Schnitt das bessere Ergebnis liefert. Die folgende Abbildung verdeutlicht für welche Korpora genau $Param5$ die höheren Attributions-Ergebnisse erzielt:

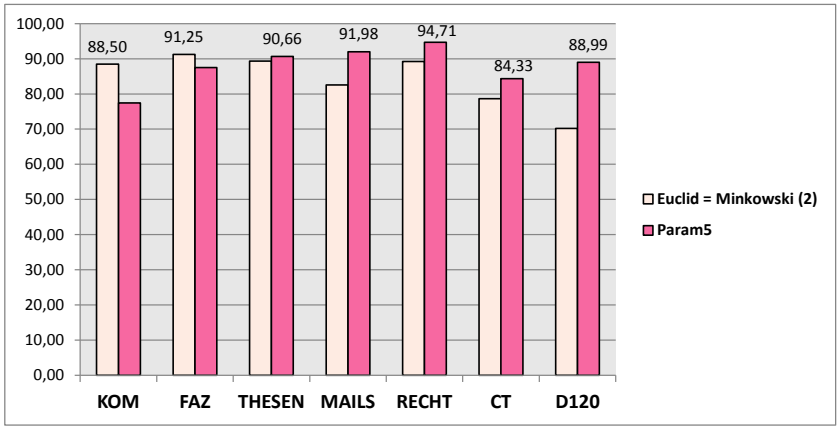


Abbildung 43: Two-Stage Pairwise Similarity Verfahren: Vergleich von *Param*5 und dem besten Ergebnis aus der Tabelle 45.

Hieraus wird ersichtlich, dass *Param*5 für insgesamt fünf aus sieben Korpora die besseren Ergebnisse erzielt und damit das Ergebnis des vorherigen Experiments (siehe Tabelle 45) deutlich übertrifft. Eine interessante Beobachtung dabei ist, dass die besseren Ergebnisse die *Param*5 erzielt, Korpora mit einer Größenordnung von ≥ 26 Autoren betreffen.

10.2.14 Gegenüberstellung: ML-Verfahren vs. Eigene Verfahren

In diesem Experiment werden die besten Ergebnisse der ML-Verfahren gegen die besten Ergebnisse der eigenen Verfahren verglichen. Das Ziel hierbei ist die Feststellung, welches Verfahren im Schnitt das beste Attributions-Ergebnis liefert. Die folgende Tabelle zeigt zunächst die besten Ergebnisse hinsichtlich der ML-Verfahren aus den vorherigen Experimenten:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
$\boldsymbol{k-NN}$	65.52	82.50	69.33	53.80	49.76	44.61	35.27	57.26
$\boldsymbol{NB}$	68.97	**91.25**	**86.67**	72.90	**83.90**	66.16	55.67	**75.07**
$\boldsymbol{SVM}$	**82.76**	**95.00**	**88.00**	79.75	76.59	54.25	41.36	73.96

Tabelle 50: ML-Verfahren im Vergleich.

Aus der Tabelle können die folgenden Erkenntnisse festgehalten werden:

- Im Schnitt liefert der NB-Klassifikator das beste Ergebniss dicht gefolgt vom SVM. Der $k-$NN Klassifikator liefert mit Abstand das schlechteste Ergebnis.
- SVM liefert für kleinere Korpora (≤ 32 Autoren) brauchbare Ergebnisse. Bei mehr als 32 Autoren klingen jedoch die Ergebnisse signifikant ab. NB erzielt dagegen für die größeren Korpora (≥ 50 Autoren) bessere Ergebnisse als SVM.
- Von allen Korpora werden die Autorschaften in $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Thesen}$ am besten attribuiert. Wie bereits in den vorherigen Experimenten angedeutet, ist dies höchstwahrscheinlich auf die Balanciertheit der beiden Korpora zurückzuführen.

Die folgende Abbildung verdeutlicht die Ergebnisse:

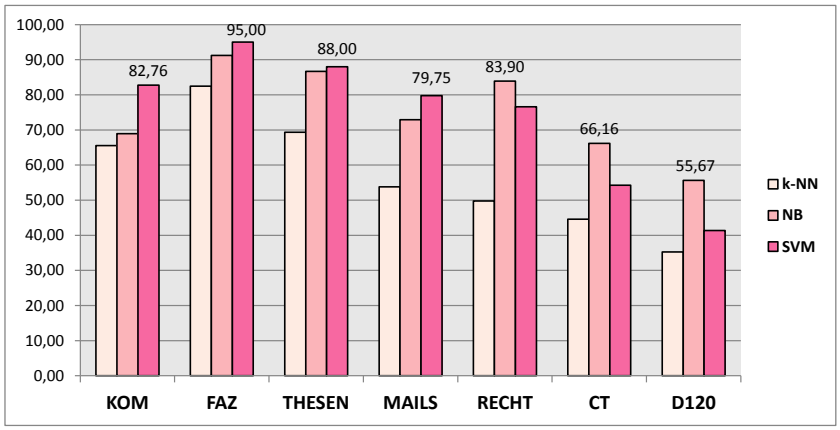


Abbildung 44: ML-Verfahren im Vergleich.

Im Folgenden werden die besten Ergebnisse hinsichtlich der eigenen Verfahren aufgeführt. Hierbei stehen die Abkürzungen PSV für das Pairwise Similarity Verfahren, FIV für das Fragmentwise Intersection Verfahren und TSPSV für das Two-Stage Pairwise Similarity Verfahren:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	∅
PSV	77.50	**85.00**	79.33	69.98	77.42	58.69	46.01	70.56
FIV	69.26	**88.75**	80.00	67.5	77.29	59.31	38.73	68.69
TSPSV	77.42	**87.50**	**90.66**	**91.98**	**94.71**	**84.33**	**88.99**	**87.94**

Tabelle 51: Eigene Verfahren im Vergleich.

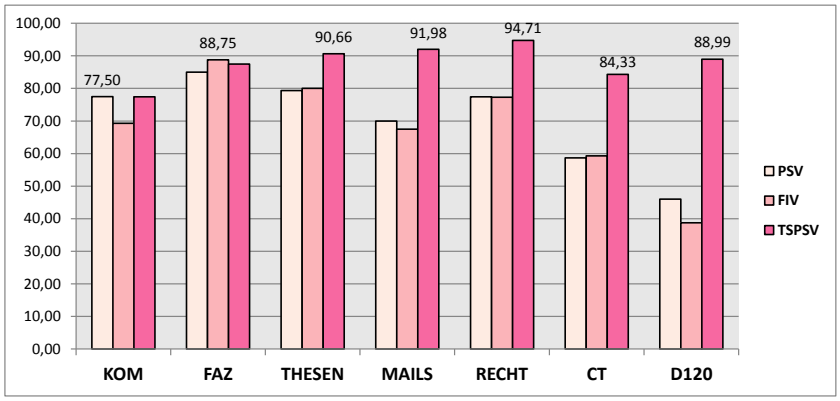


Abbildung 45: Eigene Verfahren im Vergleich.

Sowohl aus der Tabelle als auch aus der Abbildung wird ersichtlich, dass TSPSV mit Abstand die höchsten Ergebnisse hinsichtlich der eigenen Verfahren liefert. Es fällt vor allem auf, dass für größere Korpora signifikant hohe Ergebnisse erzielt werden können, was keines der Attributions-Verfahren in allen vorherigen Experimenten geschafft hat. Die folgende Abbildung illustriert den Vergleich zwischen TSPSV und die beiden ML-Verfahren NB und SVM:

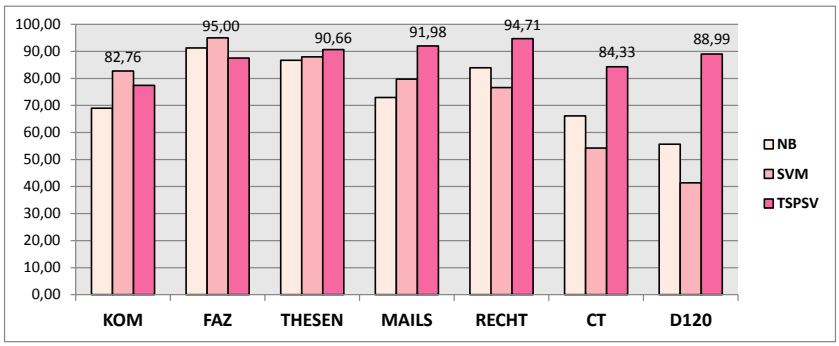


Abbildung 46: Vergleich der besten Attributions-Ergebnisse.

Hieraus können die folgenden Erkenntnisse festgehalten werden:

- Für die kleinsten Korpora ($\leq$ 10 Autoren) liefert der SVM-Klassifikator die höchsten Ergebnisse. Für alle anderen Korpora mit ($\geq$ 15 Autoren) erzielt TSPSV durchgängig die besten Ergebnisse, während die beiden ML-Verfahren NB und SVM mehr und mehr abklingen.
- Im Schnitt erzielt TSPSV 17.38 % höhere Ergebnisse gegenüber NB und 19.25 % gegenüber SVM.
- Keines der aufgeführten Verfahren bediente sich einer Feature-Auswahl. Der Grund hierfür war zum einem zu demonstrieren, dass die Verfahren fähig sind mit verrauschten Features umzugehen und zum anderen um zusätzliche Laufzeit einzusparen, die durch die aufwändige Suche nach relevanten Features entsteht.

10.3 Autorschafts-Verifikation

In diesem Abschnitt werden die beiden Verifikations-Verfahren evaluiert, die innerhalb dieser Arbeit vorgestellt und im Rahmen des Frameworks implementiert wurden.

10.3.1 Local Density Verfahren: Vier Parameter im Vergleich

In diesem Experiment wird das Local Density Verfahren anhand von vier Parameter auf sämtliche Korpora angewendet. Die Parameter lauten dabei:

	Param1	***Param2***	***Param3***	***Param4***
Schwellwert θ:	1	1	0.96	1.12
Abstandsmaß/Distanzfunktion:	Euklidische Norm	Euklidische Norm	Manhatten-Distanz	Euklidische Distanz
Feature-Kategorie(n):	F_1	$F_1 \cup F_3$	$F_1 \cup F_2 \cup F_5$	$F_7 \cup F_{13}$

Tabelle 52: Local Density Verfahren: Parametrisierung.

Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
Param1	64.37	71.32	60.48	**80.20**	70.83	55.87	72.08	67.88
Param2	**78.98**	**82.37**	**80.49**	**86.22**	**84.06**	72.56	**81.46**	**80.88**
Param3	72.73	**80.22**	71.68	**75.92**	**77.99**	73.19	71.83	74.79
Param4	47.95	60.48	53.62	70.57	66.96	47.77	58.01	57.91

Tabelle 53: Local Density Verfahren: Vier Parameter im Vergleich.

Wie aus aus der Spalte mit den Durchschnittswerten ersichtlich ist, liefert $Param2$ das eindeutig beste Verifikations-Ergebnis. Der Grund hierfür ist höchstwahrscheinlich der, dass hier die Feature-Kategorie F_3 enthalten ist. Da $Param2$ bis auf F_3 die selbe Parametrisierung aufweist, kann dies als Indiz dienen, dass dieses Ergebnis dadurch verursacht wird. Die folgende Abbildung zeigt dabei den Vergleich zu den anderen Parametern:

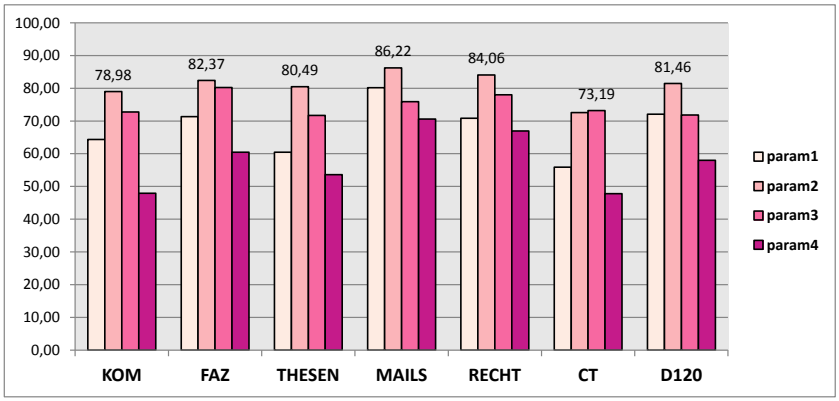


Abbildung 47: Local Density Verfahren: Vier Parameter im Vergleich.

Hier kann beobachtet werden, dass $Param2$ für sechs aus sieben Korpora die höchsten Ergebnisse erzielt, während $Param2$ dagegen für sämtliche Korpora die schlechtesten Ergebnisse vorweist. Hierbei wird jedoch nicht von den Feature-Kategorien F_7 und F_{13} als Verursacher ausgegangen, sondern vielmehr der zu hoch gesetzte Schwellwert $\theta = 1.12$. Dieser lässt offensichtlich viele Fehlklassifikationen zu. Im folgenden Teilexperiment wird $Param2$ wiederverwendet, wobei hier jedoch die $k = 3$ nächsten Nachbarn für die Klassifikation betrachtet werden, anstatt des direkten Nachbarn ($k = 1$). Das Resultat dazu lautet:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
$\boldsymbol{Param2,\ (k=3)}$	**55.27**	10.92	18.99	**49.65**	24.54	**37.47**	20.04	30.98

Tabelle 54: Local Density Verfahren: $Param2$ und $k = 3$.

Die Ergebnisse zeigen, dass die Wahl der $k = 3$ nächsten Nachbarn, zu einer deutlichen Verschlechterung der Klassifikation führen (womit eigentlich nicht gerechnet wurde). Als Ursache dafür wird starkes Rauschen innerhalb des Clusters $\mathcal{C}_{\mathcal{A}_j}$ vermutet, der dazu führt, dass der Nenner in $\frac{d_1}{d_2} \leq \theta$ zu klein wird, sodass dadurch θ überschritten und $\mathcal{A}_j$ damit nicht akzeptiert wird.

10.3.2 Cluster-Based Maximum Similarity Verfahren: Vier Parameter im Vergleich

In diesem Experiment wird das Cluster-Based Maximum Similarity Verfahren anhand von vier Parameter auf sämtliche Korpora angewendet. Die Parameter lauten dabei:

	$\boldsymbol{Param1}$	$\boldsymbol{Param2}$	$\boldsymbol{Param3}$	$\boldsymbol{Param4}$
k (bei $k-$Means):	3	3	5	2
Iterationen (bei $k-$Means):	15	8	10	6
$\ell-$häufigste Textfragmente:	80	160	90	60
$n-$Gramm Größe:	7	5	6	4
Ähnlichkeitsfunktion:	$sim_{Dice}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Cosine}(\cdot,\cdot)$	$sim_{Overlap}(\cdot,\cdot)$
Distanzfunktion:	$dist_{Manhatten}(\cdot,\cdot)$	$dist_{Euclid}(\cdot,\cdot)$	$dist_{Euclid}(\cdot,\cdot)$	$dist_{Manhatten}(\cdot,\cdot)$
Feature-Kategorie(n):	F_1	$F_1 \cup F_7 \cup F_9$	$F_2 \cup F_5 \cup F_9$	$F_2 \cup F_4$

Tabelle 55: Cluster-Based Maximum Similarity Verfahren: Parametrisierung.

Die folgende Tabelle zeigt dazu das Resultat:

	$\mathcal{K}_{KOM}$	$\mathcal{K}_{FAZ}$	$\mathcal{K}_{Thesen}$	$\mathcal{K}_{Mails}$	$\mathcal{K}_{Recht}$	$\mathcal{K}_{CT}$	$\mathcal{K}_{D120}$	$\varnothing$
$\boldsymbol{Param1}$	58.75	**65.77**	**66.57**	**71.40**	**74.28**	**65.76**	**68.55**	**67.30**
$\boldsymbol{Param2}$	49.71	61.46	61.18	58.34	**62.65**	52.98	59.22	57.93
$\boldsymbol{Param3}$	43.39	**62.85**	56.80	60.15	**64.65**	51.43	**63.94**	57.60
$\boldsymbol{Param4}$	43.90	58.00	54.17	51.26	59.50	47.91	29.42	49.17

Tabelle 56: Cluster-Based Maximum Similarity Verfahren: Vier Parameter im Vergleich.

Aus der Tabelle kann ein linearer Abstieg der Ergebnisse entnommen werden. Als Ursache hierfür werden die kleineren $n-$Gramm Größen vermutet, die allesamt kleiner sind als der in $Param1$. Die folgende Abbildung verdeutlich die Ergebnisse:

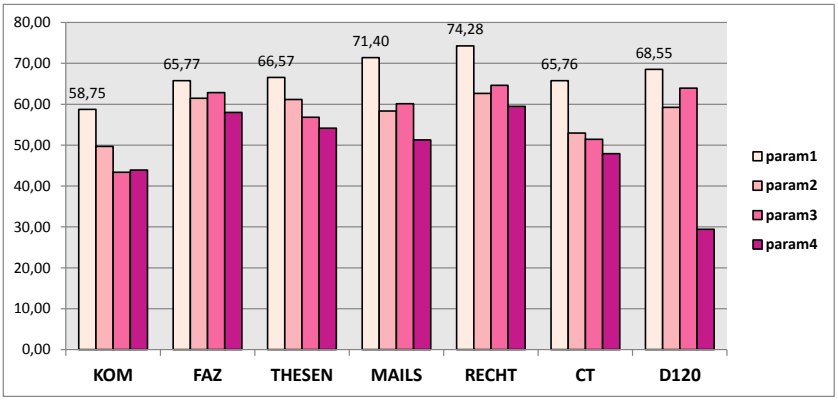


Abbildung 48: Cluster-Based Maximum Similarity Verfahren: Vier Parameter im Vergleich.

Hieraus geht hervor das $Param4$ in fast allen Fällen (sechs aus sieben Korpora) das schlechteste Ergebnis aufweist. Als Grund werden hier die $\ell = 60$-häufigste Textfragmente vermutet, die wahrscheinlich nicht ausreichen, um $\mathcal{A}_j$ verifizieren zu können.

10.4 Intrinsische Exploration

In der folgenden Experimentierreihe werden zwei Experimente hinsichtlich des Cluster-Based Distinction Verfahren durchgeführt. Da es sich dabei um ein intrinsisches Verfahren handelt, existiert hier keine Trainingsmenge, sodass an dieser Stelle keine Evaluierungsstrategie verwendet wird (anders als bei der Autorschafts-Attribution bzw. Verifikation). Die Korpora die in den folgenden Experimenten verwendet werden, unterscheiden sich zudem von denen, aus den vorherigen Experimenten. Diese müssen hinsichtlich der Dokumente derart modifiziert werden, sodass jedes Dokument einen fremden Stil und damit eine stilistische Inkonsitenz aufweist. Hierfür wird die folgende Vorgehensweise beschrieben, mit der die Korpora entsprechend modifiziert werden:

Zunächst werden aus einem Korpus $\mathcal{K}$ sämtliche darin befindliche Dokumente kopiert und künstlich „plagiiert“, um dadurch eine stilistische Inkonsitenz zu erzeugen. In jedes dieser Dokumente wird ein bestimmter Textabschnitt eines fremden Autors eingefügt, der dabei die Bezeichnung `PLAG-SENTENCES` trägt. Genauer betrachtet stellt `PLAG-SENTENCES` eine Menge von Sätzen eines anderen Autoren dar, die in das ursprüngliche Dokument eingefügt werden. Jeder plagiierter Satz $\sigma \in$ `PLAG-SENTENCES` hat dabei eine Mindestlänge, die durch `PLAG-SENT-LENGTH` ausgedrückt wird. Nachdem sämtliche Dokumente auf diese Weise plagiiert wurden, werden diese zurück in $\mathcal{K}$ verschoben. Der Korpus ändert sich dadurch in $\mathcal{K}'$ und enthält dabei insgesamt $2|\mathcal{K}|$ Dokumente. In Worten: Der neue Korpus enthält die doppelte Menge an Dokumente gegenüber dem Original-Korpus. Anhand dieser modifizierten Korpora werden im Folgenden die insgesamt drei Experimente durchgeführt.

Anmerkung: Als Zerlegungsstrategie für die Experimente wurde die Near-Same-Length Strategie gewählt.

10.4.1 Cluster-Based Distinction Verfahren: Anwendung unterschiedlicher Parameter auf dem FAZ-Korpus

In diesem Experiment wird das Cluster-Based Distinction Verfahren anhand von insgesamt fünf Parameter auf den modifizierten[1] Korpus $\mathcal{K}'_{FAZ}$ angewendet. Hierbei wurde `PLAG-SENTENCES` $= 8$ und `PLAG-SENT-LENGTH` $= 30$ gewählt. Die weiteren fünf Parameter lauten:

[1]Siehe dazu den vorherigen Abschnitt.

	Param1	***Param2***	***Param3***	***Param4***	***Param5***
1, $n-$Gramm Größe:	2	6	4	7	3
$k-$häufigsten $n-$Gramme:	40	80	60	50	60
Ähnlichkeitsfunktion:	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$	$sim_{Jaccard}(\cdot,\cdot)$
Anz. der Slices:	9	5	7	5	9

Tabelle 57: Cluster-Based Distinction Verfahren: Parametrisierung.

Das Ziel dieses Experiments war feststellen zu können, wie sich unter anderem die unterschiedliche $n-$Gramm Größen auf die Ergebnisse auswirken. Die folgende Tabelle zeigt dazu das Resultat. Hierbei stellt *Retrieve* den $F_1-Measure$ Wert dar, der angibt wieviele relevante Dokumente gefunden werden konnten. *Found* stellt dagegen den prozentualen Anteil des „plagiierten Textes“ dar, der gefunden werden konnte:

	Param1	***Param2***	***Param3***	***Param4***	***Param5***	∅
Retrieve	**57.75**	47.73	49.14	**52.46**	59.0	**53.22**
Found	16.53	**19.15**	11.65	17.73	13.8	15.77

Tabelle 58: Cluster-Based Distinction Verfahren: Anwendung auf dem FAZ-Korpus.

Aus der Tabelle geht hervor, dass $Param5$ die meisten Dokumente findet, die eine stilistische Inkonsistenz aufweisen. Allerdings ist $Param2$ dagegen in der Lage, den größten Umfang des plagiierten Textes tatsächlich zu finden.

10.4.2 Cluster-Based Distinction Verfahren: Anwendung von drei Parameter auf dem Thesen-Korpus

In diesem Experiment wird das Cluster-Based Distinction Verfahren anhand einer einheitlichen Parametrisierung (bis auf die $k-$häufigsten $n-$Gramme) auf das modifizierte Korpus $\mathcal{K}'_{Thesen}$ angewendet. Hierbei wurde `PLAG-SENTENCES` $= 7$ und `PLAG-SENT-LENGTH` $= 25$ gewählt. Die Parametrisierung lautet wie folgt:

	Param1	***Param2***	***Param3***
1, $n-$Gramm Größe:	3	3	3
$k-$häufigsten $n-$Gramme:	40	70	100
Ähnlichkeitsfunktion:	$sim_{Cosine}(\cdot,\cdot)$	$sim_{Cosine}(\cdot,\cdot)$	$sim_{Cosine}(\cdot,\cdot)$
Anz. der Slices:	9	9	9

Tabelle 59: Cluster-Based Distinction Verfahren: Parametrisierung.

Das Ziel dieses Experiments war feststellen zu können, welchen Einfluß (zunehmende) $k-$häufigsten $n-$Gramme auf die Ergebnisse haben. Die folgende Tabelle zeigt dazu das Resultat:

	Param1	***Param2***	***Param3***	∅
Retrieve	**56.90**	46.43	53.98	**52.44**
Found	**10.15**	9.69	7.85	9.23

Tabelle 60: Cluster-Based Distinction Verfahren: Anwendung auf dem Thesen-Korpus.

Die Ergebnisse aus der Tabelle sind nicht offensichtlich. So liefert $Param1$ anhand der 40-häufigsten Trigramme das höchste Ergebnis mit 56.90 %. $Param2$ enthält dagegen die 70-häufigsten Trigramme weist jedoch damit ein schlechteres Ergebnis auf, während $Param3$ mit den 100-häufigsten Trigramme wieder zu einem höheren Ergebnis führt. Es kann allenfalls festgehalten werden, dass der Umfang des tatsächlich plagiierten Textes mit den zunehmenden $k-$häufigsten $n-$Gramme sinkt. Eine weitere Beobachtung ist hier nicht erkennbar.

10.5 Sonstige Experimente

In diesem Abschnitt werden sonstige Experimente bzw. Beobachtungen erläutert, die sich nicht in den vorherigen Abschnitten einordnen ließen.

10.5.1 Stil-Diskriminierung anhand von $n-$Gramme

In fast jedem Experiment wurde gezeigt, dass $n-$Gramme eine sehr gute Diskriminierungskraft besitzen. In diesem Abschnitt wird erläutert, warum gerade diese so mächtig sind.

$n-$Gramme haben einige Eigenschaften, die sie von anderen Features unterscheiden. Genauer betrachtet stellen sie selbst keine spezifischen Features dar. Vielmehr können $n-$Gramme als eine Familie von Features verstanden werden, die in Abhängigkeit von n definiert werden. Was $n-$Gramme besonders macht ist die Tatsache, dass sie in der Lage sind viele unterschiedliche Features einzufangen. Zu diesen zählen beispielsweise:

- **Zeichenbasierte Features:** Buchstaben, Präfixe, Suffixe, Fugenelemente, Morpheme, etc.
- **Tokenbasierte Features:** Kollokationen, Wortzusammensetzungen (mit/ohne Bindestrich) einfache Phrasen, Suffix-Wort-Präfix Fragmente, etc.

In den Experimenten wurden die besten Ergebnisse mit 2−Gramme und 6−Gramme erzielt. Diese enthalten dabei zum einen Fragmente von inhaltsbasierten Wörtern und zum anderen Funktionswörter[1] (bzw. Teile davon). Im Folgenden werden einige Beispiele für 6−Gramme betrachtet:

`Darmst`, `n␣soll`, `häftig`, `Publik`, `ikatio`, `en.␣De`, `mit␣de`, `ntlich`, `ericht`, `nd␣die`, ...

Es fällt auf, dass die 6−Gramme unter anderem ganze Funktionswörter wie etwa `soll` oder `die` als auch Stammmorpheme von Nomen wie etwa `Publik` enthalten. Zusätzlich dazu werden auch Wort-Präfix Fragmente wie etwa `mit␣de` eingefangen, die spezifisch für manche Autoren sein können. Wenn ein Autor beispielsweise die folgenden Wortkombinationen häufig einsetzt: `mit dem, mit der, mit dessen, mit denjenigen, mit dementsprechenden,...` so werden diese allesamt durch das 6−Gramm `mit␣de` dargestellt, welches eines von mehreren typischen Stil-Repräsentanten eines Autors wiederspiegeln kann. Liegen genügend solcher Repräsentanten vor, können die jeweiligen Autorenstile voneinander leicht unterschieden werden. Der Grund hierfür ist, dass es eher unwahrscheinlich ist, dass mehrere Autoren exakt die selben Stil-Repräsentanten aufweisen.

Bei 2−Gramme werden sehr häufig Funktionswort-Fragmente sowie Präfixe/Suffixe von inhaltsbasierten Wörtern eingefangen. In den Experimenten wurde z.B. bei vereinzelten Autoren der Einsatz des Umlaut-Suffix `aß` beobachtet, der bei vielen anderen Autoren in Form von `as` vorkam. Die Vermutung hierbei ist, dass diejenigen Autoren, die `aß` verwendet haben, noch an die alte Rechtschreibung gewöhnt sind, und so z.B. `Faß` statt `Fass` und andere Wörter mit diesem Umlaut in ihren Texten verwenden. Der folgende Auszug zeigt einige 2−Gramme, die innerhalb der vorgestellten Korpora zu einer guten Diskriminierung der Autorenstile geführt haben: `de`, `di`, `ei`, `es`, `ge`, `mi`, `je`, ...

Nach einer eigenen Vermutung kann hier geschlossen werden, dass je kleiner n gewählt wird, desto höher die Wahrscheinlichkeit ist, dass der jeweilige $n-$Gramm ein Fragment eines Wortes darstellt, welches keine Semantik in sich trägt. Wenn n dagegen größer gewählt wird, gilt die Umkehrung (der $n-$Gramm enthält ein Wort/Fragment der eine Semantik aufweist, z.B. Adjektive oder Verben).

Fazit: Innerhalb der Arbeit wurden register- und genreübergreifende Korpora eingesetzt, deren Autoren anhand von 2−Gramme und 6−Gramme gut bis sehr gut attribuiert werden konnten. Auch wenn Stil unabhängig vom Genre ist, kann festgehalten werden, dass sich dieser durchaus anhand von genrebehafteten $n-$Grammen diskriminieren lässt.

[1] Zur Erinnerung: Funktionswörter weisen keine Semantik auf.

10.5.2 Stil-Diskriminierung anhand von Funktionswörtern

In diesem Experiment soll gezeigt werden, aus welchem Grund Funktionswörter (Feature-Kategorie F_4) Autorenstile diskrimieren können. Zur Veranschaulichung wurde hierfür der Korpus $\mathcal{K}_{KOM}$ ausgewählt, da hier die Anzahl der Autoren überschaubar ist. Das folgende Parallele-Koordinaten Diagramm zeigt die Anwendung der zehn häufigsten Funktionswörter im Deutschen, in Bezug auf die fünf Autoren innerhalb von $\mathcal{K}_{KOM}$:

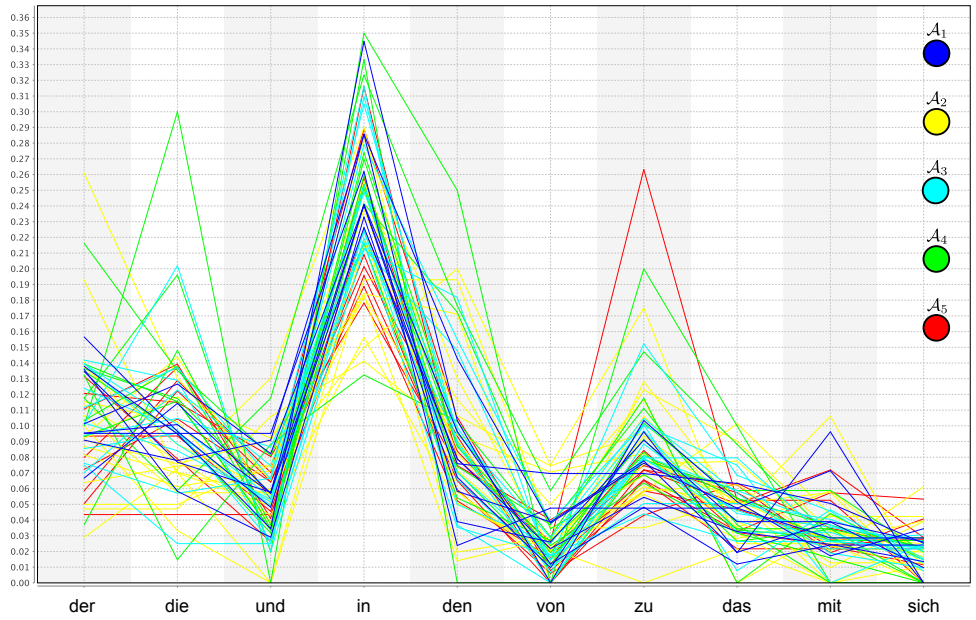


Abbildung 49: Verteilung der zehn häufigsten Funktionswörter im Deutschen (bezogen auf die fünf Autoren in $\mathcal{K}_{KOM}$).

Jede Achse in diesem Diagramm spiegelt ein einzelnes Feature dar. Die horizontalen Linien repräsentieren dagegen die Feature-Vektoren bzw. die jeweiligen Autorenstile. Jedes Dokument wird hinsichtlich seines Autors durch eine Farbe abgebildet.

Bei einer genaueren Betrachung fällt hier zunächst auf, das die Verteilung annähernd gleich ist, da die Linien in der selben Struktur verlaufen. Dennoch lassen sich die einzelnen Autorenstile unterscheiden, da beispielsweise die Funktionswörter `die` und `zu`, über eine (wenn auch nicht hohe) Varianz verfügen. In diesem Szenario ist die Varianz[1] verhältnismäßig klein, wenn jedoch der gesamte Spektrum an Funktionswörtern betrachtet wird, verstärkt sich diese zunehmend.

Im Rahmen dieser Arbeit wurden insgesamt 759 Funktionswörter für die Feature-Kategorie F_4 verwendet. Ihre Anwendung lieferte nach der Feature-Kategorie F_3 häufig die zweithöchsten Ergebnisse. Der Grund hierfür ist zum einem der, dass Funktionswörter in fast jeden Text enthalten sind und zum anderen, dass Autoren diese öfters in unterschiedlichen Häufigkeiten verwenden, sodass ihre Stile dadurch gut diskriminiert werden können.

[1] Prinzipiell gilt: Je weniger horizontale Linien ein und denselben Punkt auf einer Achse schneiden, desto mehr Varianz enthält das Feature f_i, der diese Achse darstellt.

11 Zusammenfassung

Das primäre Ziel dieser Arbeit war die Entwicklung eines Autorschaftsanalyse-Systems, welches Verfahren für die Autorschafts-Attribution, Autorschafts-Verifikation und die intrinsische Exploration umsetzt. Die Verfahren, die im Rahmen eines Frameworks implementiert wurden, basieren teilweise auf Konzepten und Modellen aus existierenden Arbeiten. Die einzelnen Unterschieden zwischen den eigenen und den bestehenden Verfahren wurden dabei im Detail genannt. Einige der Verfahren basieren auf Klassifikatoren und Clustering-Verfahren, die Standardalgorithmen des maschinellen Lernens darstellen. Hierbei wurden zwei Klassifikatoren sowie ein Clustering-Verfahren implementiert und für das Framework entsprechend angepasst. Diese dienten zum einem als Baselines und zum anderen als eine Basis, um die eigenen Verfahren erweitern zu können.

Neben der Implementierung des Frameworks war ein weiteres wichtiges Ziel, bestehende Stilmerkmale zu untersuchen und neue Stilmerkmale zu finden, die sich vor allem für die deutsche Sprache eignen, da diese die Ausgangsprache der implementierten Verfahren darstellen. Hierbei wurde großen Wert darauf gelegt, dass diese domänenübergreifend eingesetzt werden können, um dadurch möglichst unabhängig vom Register oder Genre zu sein. Um die Domänenunabhängigkeit bewerten zu können, wurden insgesamt sieben Korpora zusammengestellt, die sich durch Register, Genre, Jargon und weitere stilistische Varietäten unterscheiden, insbesondere auch durch das sogenannte Class-Imbalance. In einem gesonderten Kapitel wurde darauf eingegangen, mit welchen Mitteln Stilmerkmale aus den Dokumenten bzw. aus den sprachlichen Ebenen der Texten entnommen werden können, um dadurch Autorenstile zu approximieren. Unter anderem wurden unterschiedliche Natural Language Processing Werkzeuge eingesetzt, um die Texte mit unterschiedlichen Annotationen zu versehen. Die Annotationen wurden dabei als Annäherungen der sprachlichen Ebenen aufgefasst, da sich diese nicht vollständig maschinell abbilden lassen, zumindest nicht mit den Standard NLP-Werkzeugen.

Sämtliche Verfahren wurden anhand der Features und Korpora in einer umfangreichen Experimentenreihe evaluiert. Hierbei konnten zahlreiche Erkenntnisse festgehalten werden. Die wohl wichtigste Erkenntniss betrifft die $n-$Gramme Features. Diese konnten als die diskriminierendsten Stilmerkmale identifiziert werden und sind mit Abstand allen anderen Features überlegen. Selbst die in der Literatur öfters eingesetzen Funktionswörter, waren bei weitem nicht so mächtig wie die $n-$Gramme, wenn es darum ging Autorenstile über Register und Genre hinweg unterscheiden zu können. Belegt werden konnte dies anhand von mehreren Experimenten in unterschiedlichen Ausführungen. Der Grund warum $n-$Gramme so erfolgreich sind, wurde dabei ebenfalls kurz angerissen. Es kann festgehalten werden, dass eine erfolgreiche Autorschaftsanalyse alleine nur mit diesen Features ermöglicht werden kann. Das bestmögliche Ergebnis, welches für sämtliche sieben Korpora erzielt werden konnte, lieferte das Two-Stage Pairwise Similarity Verfahren mit einer Erkennungsgenauigkeit von 87.94 % im Durchschnitt. Das Verfahren als auch das Ergebnis übertrief alle anderen Attributions-Verfahren, inklusive die der drei Klassifikatoren $k-$NN, NB und SVM, die als Baselines in den Experimenten verwendet wurden. Für die Autorschafts-Attribution wurde somit festgestellt, dass dieses Verfahren als praxistauglich eingestuft werden kann, vor allem weil es nicht nur im Schnitt die besten Ergebnisse liefert, sondern dazu für größere Korpora (mit einer Größenordnung von 100 Autoren) brauchbare Ergebnisse liefern kann.

Neben diskriminiereden Features wurde zudem der Einfluß von Metriken untersucht, die die Basis mehrerer Verfahren darstellen. Hierbei wurde beispielsweise gezeigt, dass die Wahl von spezifischen vektorbasierten Distanzfunktionen zu einer deutlichen Verbesserung der Ergebnisse führen kann. Anders dagegen verhalten sich mengenbasierte Ähnlichkeitsfunktionen. Hier konnten, ausgehend von vier verschiedenen Funktionen, keinerlei Unterschiede bei den durchgeführten Experimenten festgestellt werden. In vielen Attributions-Experimenten konnte festgestellt werden, dass die balancierten Korpora $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Thesen}$ die besten Ergebnisse gegenüber allen anderen Korpora erzielen konnten. Es empfiehlt sich daher in realen Attributions-Szenarien Korpora stets hinsichtlich der Anzahl von Dokumenten je Autor als auch hinsichtlich der Dokumentlängen zu vereinheitlichen.

12 Ausblick

Die vorliegende Arbeit bietet zahlreiche Ideen für zukünftige Arbeiten. Unter anderem könnten in naher Zukunft die Implementierung der theoretischen Ansätzen, von denen insgesamt fünf Ideen vorgestellt wurden, angegangen werden. Allen voran wäre hierbei die Umsetzung des Pre-Profiling-Attribution Verfahres am realistischsten, da hierfür bereits einige Ansätze (wenn auch nur aus englischen Sprachraum) existieren. Auch die Umsetzung des Ensemble Metric Scheme Ansatzes könnte realisiert werden. Zuvor müsste hier jedoch erforscht werden, welche Metriken miteinander sinnvoll kombiniert werden könnten, da eine wahllose Kombination von Metriken zu nicht interpretierbaren Ergenissen führen würde.

Neben den theoretischen Ansätzen, können auch die implementierten Verfahren, um weitere Funktionalitäten erweitert werden. Für das Two-Stage Pairwise Similarity Verfahren würde es sich z.B. anbieten, die Vorfilterung der Trainingsmenge noch weiter zu optimieren, oder jedoch auch die zweite (aufwändige) Stufe des Verfahrens, in der die Attribution durchgeführt wird, zu beschleunigen.

Für die Verifikations-Verfahren könnten umfangreichere Ausreißer-Erkennungsmechanismen eingesetzt, oder auch andere Bedingungen für die Akzeptanz bzw. Ablehung des zu verifizierenden Autors festgelegt werden. Innerhalb dieser Arbeit wurden nur einfache Bedingungen betrachtet (wie z.B. die Cluster-Struktur der Trainingsdokumente).

Für die intrinsische Exploration könnte das Verfahren selbst verbessert werden, da die Retrieve-Phase (das Finden von stilistischen Inkonsistenzen) nur knapp die 50-Prozent-Marke erreicht. Hier wäre analog zu den Verifikations-Verfahren der Einsatz von ausgereiften Ausreißer-Erkennungsmechanismen von zentraler Bedeutung. Weiterhin könnte auch die Postprocessing-Komponente optimiert werden, um dadurch insbesondere eine multiple Autorschafts-Attribution ermöglichen zu können. Im Rahmen der Experimente konnte leider nur ein kleiner Umfang der künstlich erzeugten Stil-Inkonsistenzen ermittelt werden.

Aus den Experimenten innerhalb dieser Arbeit gingen neben Erkenntnissen auch einige Herausforderungen hervor. Die wichtigste betrifft dabei die Parametrisierungen der meisten Autorschaftsanalyse-Verfahren, vor allem die der Autorschafts-Verifikation und die der intrinsischen Exploration. Da hier zahlreiche Einstellungsmöglichkeiten gewählt werden können, kann nicht beurteilt werden, ob in den durchgeführten Experimenten die bestmögliche Parametrisierung gefunden werden konnte. Des Weiteren ergaben manche Parameter semantisch gesehen ein Sinn, rechnerisch jedoch nicht, was zu einer fehlenden Interpretation geführt hat. Es wird davon ausgegangen, dass für diese Verfahren eine weitere Optimierung gefunden werden kann. In einer weiterführenden Arbeit könnten hierfür beispielsweise genetische bzw. evolutionäre Verfahren entwickelt werden, welche eine optimale Parametersuche ermöglichen können. Interessant wären hierbei Verfahren, die vor allem ein globales Optimum garantieren können (z.B. SVM-basierte Verfahren). Die Hoffnung ist hierbei, dass anhand einer geeigneten Parametrisierung, die Verfahren (als auch die Ergebnisse) nicht nur optimiert, sondern auch besser interpretiert werden können.

Literatur

[1] Ahmed Abbasi and Hsinchun Chen. Applying Authorship Analysis to Extremist-Group Web Forum Messages. *IEEE Intelligent Systems*, 20(5):67–75, 2005. (Zitiert auf Seite 74)

[2] AclWeb. The Association for Computational Linguistics, Pennsylvania, USA. 2012. http://aclweb.org. (Zitiert auf Seiten 74 und 76)

[3] ACM. The Association for Computing Machinery, Inc., New York, USA. 2012. http://www.acm.org. (Zitiert auf Seiten 74 und 76)

[4] Daniel Ammann. *Stumme Zeugen: Fussspuren, Fingerabdruck und moderne Forensik.* ph|akzente, Pädagogische Hochschule Zürich, 2011. (Zitiert auf Seite 24)

[5] Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. Gender, Genre, and Writing Style in Formal Written Texts. *TEXT*, 23:321–346, 2003. (Zitiert auf Seite 109)

[6] Ioannis Avraam and Ioannis Anagnostopoulos. A Comparison over Focused Web Crawling Strategies. In *Proceedings of the 2011 15th Panhellenic Conference on Informatics*, PCI '11, pages 245–249, Washington, DC, USA, 2011. IEEE Computer Society. (Zitiert auf Seite 89)

[7] Max Barthel. *Der Mensch am Kreuz: Roman nach dem Tagebuch eines katolischen Pfarrers.* Bücherkreis Berlin, 1929. (Zitiert auf Seiten 25, 29 und 66)

[8] C. Biemann, G. Heyer, U. Quasthoff, and M. Richter. The Leipzig Corpora Collection - Monolingual Corpora of Standard Size. In *Proceedings of Corpus Linguistic 2007*, Birmingham, UK, 2007. (Zitiert auf Seite 63)

[9] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity Measures for Categorical Data: A Comparative Evaluation Abstract. In *SIAM Data Mining Conference*, 2008. (Zitiert auf Seite 36)

[10] Thorsten Brants, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut, and Hans Uszkoreit. Das NEGRA-Annotationsschema. Negra project report, Universität des Saarlandes, Saarbrücken, 1997. (Zitiert auf Seite 87)

[11] Prof. Dr. Christian Breiteneder. Multimedia 2 - Vorlesungsunterlagen (Kapitel 6), Interactive Media Systems Group, Institute for Software Technology and Interactive Systems, Vienna University of Technology. `http://www.ims.TUWien.ac.at`, 2009. Letzter Zugriff: 29.07.2012. (Zitiert auf Seite 35)

[12] Steven Burrows. *Source Code Authorship Attribution.* PhD thesis, School of Computer Science and Information Technology, RMIT University, Melbourne, Australia, November 2010. (Zitiert auf Seite 90)

[13] CiteSeer. The College of Information Sciences and Technology, The Pennsylvania State University, USA. 2012. http://citeseerx.ist.psu.edu/index. (Zitiert auf Seiten 74 und 76)

[14] Clab-Team. Chunk Parsing. *Institut für Computerlinguistik, Universität Zürich*, November 2008. (Zitiert auf Seite 31)

[15] Simon Clematide. Einführung in die Computerlinguistik I (Vorlesungsskript), Institut für Computerlinguistik, Universität Zürich. `http://http://www.cl.uzh.ch/people/team/siclemat.html`, 2010. Letzter Zugriff: 17.08.2012. (Zitiert auf Seiten 30 und 31)

[16] Prof. Jürgen Cleve. Data Mining (Vorlesungsskript), Lehrgebiet: Grundlagen der Informatik / Künstliche Intelligenz, Fakultät für Wirtschaftswissenschaften, Hochschule Wismar. `http://www.wi.hs-wismar.de/~cleve/vorl/dmining/dmcolor.pdf`, 2011. Letzter Zugriff: 26.07.2012. (Zitiert auf Seiten 41 und 42)

[17] Xavier Roche & Other Contributors. HTTrack Website Copier, Offline Browser. `http://www.httrack.com`, 2011. Letzter Zugriff: 23.12.2011. (Zitiert auf Seite 120)

[18] I. Cramer, M. Finthammer, A. Kurek, L. Sowa, M. Wachtling, and T. Claas. Experiments on lexical chaining for german corpora: Annotation, extraction, and application. *Journal for Language Technology and Computational Linguistics (JLCL).*, 23, 2008. (Zitiert auf Seite 112)

[19] Pádraig Cunningham and Sarah Jane Delany. k-Nearest Neighbour Classifiers. Technical report ucd-csi-2007-4, University College Dublin Padraig, Dublin Institute of Technology, Dublin, 2007. (Zitiert auf Seite 41)

[20] M. Dash and H. Liu. Feature Selection for Classification. *Intelligent Data Analysis*, 1:131–156, 1997. (Zitiert auf Seiten 68 und 69)

[21] Dr. Matthias Schubert Dr. Peer Kröger. Knowledge Discovery in Databases (Vorlesungsskript), Lehr- und Forschungseinheit für Datenbanksysteme, Institut für Informatik, Ludwig Maximilians Universität München. `http://www.dbs.ifi.lmu.de/Lehre/KDD`, 2006. Letzter Zugriff: 29.07.2012. (Zitiert auf Seiten 50 und 54)

[22] Prof. Dr. Raimund H. Drommel. Webseite: *Anonyme Briefe enthalten mehr unbeabsichtigte Hinweise auf ihre Autoren, als allgemein angenommen*, howpublished = `http://www.sprachdetektiv.de`, note = Letzter Zugriff: 13.02.2012, year = 2012. (Zitiert auf Seite 24)

[23] Prof. Dr. Raimund H. Drommel. Anonymschreiben – Sprachprofiling und vergleichende Autorschaftsbestimmung (Detektiv-Kurier), Neue Aspekte der Zusammenarbeit zwischen Detektiven und Gutachtern. `http://http://www.sprachdetektiv.de/downloads/Anonymschreiben.pdf`, 2001. Letzter Zugriff: 26.06.2012. (Zitiert auf Seite 24)

[24] Stefan Eberhardt. Support Vector Machines For Pattern Recognition. *Hauptseminar Statistische Lerntheorie, Abteilung Neuroinformatik, Universität Ulm*, 2007. http://www.informatik.uni-ulm.de/ni/Lehre/WS06/SLTHS/ausarbeitungen/Eberhardt.pdf. (Zitiert auf Seiten 44 und 50)

[25] Tatjana Eitrich. *Support-Vektor-Maschinen und ihre Anwendung auf Datensätze aus der Forschung.* Berichte des Forschungszentrums Jülich. Zentralinstitut für Angewandte Mathematik, Forschungszentrum Jülich GmbH - Zentralbibliothek, Jülich, 2003. (Zitiert auf Seite 44)

[26] Elsevier. Elsevier B.V. Amsterdam, Netherlands. 2012. http://www.elsevier.com. (Zitiert auf Seiten 74 und 76)

[27] Peter Fankhauser and Elke Teich. *Exploring Lexical Patterns in Text: Lexical Cohesion Analysis with WordNet.* Univ.-Verl., Potsdam, Januar 2005. (Zitiert auf Seite 111)

[28] Claudia Felsch. *Frequentielle Lesbarkeitsanalyse.* GRIN Verlag GmbH, 2007. (Zitiert auf Seite 63)

[29] Thomas Finley and Thorsten Joachims. Supervised Clustering with Support Vector Machines. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 217–224, New York, NY, USA, 2005. ACM. (Zitiert auf Seite 51)

[30] Ronny Fischer. *Netzwerk-Forensik: Digitale Beweismittelsicherung in Netzwerken.* Präsentationen des IT-Security Forum #5. GO OUT Production GmbH, 2006. (Zitiert auf Seite 80)

[31] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, Carole E. Chaski, and Blake Stephen Howald. Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. *IJDE*, 6(1), 2007. (Zitiert auf Seite 95)

[32] Prof. Dr. Christina Gansel. Philologische Methoden (Vorlesungsskript), Ernst-Moritz-Arndt-Universität Greifswald Philosophische Fakultät (Institut für Deutsche Philologie). `http://http://www.phil.uni-greifswald.de/fileadmin/mediapool/general_studies/FolienV2.pdf`, 2008. Letzter Zugriff: 11.08.2012. (Zitiert auf Seite 23)

[33] Antonio Miranda García and Javier Calle Martín. Function Words in Authorship Attribution Studies. *Literary and Linguistic Computing*, 22:49–66, 04/2007 2007. (Zitiert auf Seite 84)

[34] BUCHFUNK Hörbuchverlag GbR. Deutsch24: Grammatik-Glossar. `http://www.deutsch24.net/moodle`, 2012. Letzter Zugriff: 15.03.2012. (Zitiert auf Seiten 10, 20 und 21)

[35] Eugenie Giesbrecht and Stefan Evert. Is Part-of-Speech Tagging a Solved Task? An Evaluation of POS Taggers for the German Web as Corpus. *Web as Corpus Workshop WAC5*, page 27, 2009. (Zitiert auf Seiten 30 und 116)

[36] IFB Verlag Deutsche Sprache GmbH. Der Anglizismen-INDEX. `http://www.vds-ev.de/anglizismenindex`, 2012. Letzter Zugriff: 08. August 2012. (Zitiert auf Seite 63)

[37] Felix Golcher. *Ein Einblick in die statistische Stilometrie*. Humboldt-Universität zu Berlin, Institut für deutsche Sprache und Linguistik, Korpuslinguistik, 2007. (Zitiert auf Seiten 23 und 80)

[38] Felix Golcher. *Logarithmus muss: Kritik an einer verbreiteten Methodik der Stilometrie*. Humboldt-Universität zu Berlin, Institut für deutsche Sprache und Linguistik, Korpuslinguistik, 2010. (Zitiert auf Seite 80)

[39] PD Dr. Karin Haenelt. Ähnlichkeitsmaße für Vektoren (Kursfolien), Information Retrieval. `http://http://kontext.fraunhofer.de/haenelt/kurs/InfoRet/termine_folien.php`, 2007. Letzter Zugriff: 06.02.2012. (Zitiert auf Seiten 10 und 37)

[40] Lam Ngoc Hai. Automatisches Erkennen von Problemen mit Produkten in Social Media (Twitter, Blogs, Facebook, etc.). Studentische Arbeit, Supervisor: Dipl.-Medien-Inf. David Urbansky. Master's thesis, Technische Universität Dresden, Faculty of Computer Science, Dresden, 2011. (Zitiert auf Seite 30)

[41] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005. (Zitiert auf Seiten 10, 41, 42, 44, 51, 55, 69 und 70)

[42] Lena Heine. *Linguistics@schools: Abenteuer Sprachwissenschaft; Kooperationsmöglichkeiten zwischen Schule und Hochschule*. Peter Lang, 2010. (Zitiert auf Seite 81)

[43] Gerhard Heyer, Uwe Quasthoff, and Thomas Wittig. *Text Mining: Wissensrohstoff Text – Konzepte, Algorithmen, Ergebnisse*. W3L-Verlag, 2008. (Zitiert auf Seiten 20, 22 und 36)

[44] Dr. Anke Holler. *Pro Seminar: Satzübergreifende Phänomene (Anaphorik und Koreferenz)*. Ruprecht- Karls Universität Heidelberg, 2005. (Zitiert auf Seite 111)

[45] Prof. David I. Holmes. Webseite: *Department of Mathematics and Statistics, College of New Jersey, USA*, howpublished = `http://mathstat.pages.tcnj.edu/information/faculty-profiles`, note = Letzter Zugriff: 05.02.2012, year = 2012. (Zitiert auf Seite 163)

[46] John Houvardas and Efstathios Stamatatos. N-gram feature selection for authorship identification. In *AIMSA*, pages 77–86, 2006. (Zitiert auf Seite 80)

[47] Farkhund Iqbal, Liaquat A. Khan, Benjamin C. M. Fung, and Mourad Debbabi. E-Mail Authorship Verification for Forensic Investigation. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1591–1598, New York, NY, USA, 2010. ACM. (Zitiert auf Seite 89)

[48] Shunichi Ishihara. A Forensic Authorship Classification in SMS Messages: A Likelihood Ratio Based Approach Using N-gram. In *Proceedings of the Australasian Language Technology Association Workshop 2011*, pages 47–56, Canberra, Australia, December 2011. (Zitiert auf Seite 89)

[49] Jing Jiang and ChengXiang Zhai. Instance Weighting for Domain Adaptation in NLP. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271, Prague, Czech Republic, June 2007. Association for Computational Linguistics. (Zitiert auf Seite 87)

[50] Prof. Xiaoyi Jiang and Kai Rothaus. Klassifikation mit Distanzfunktionen (Vorlesungsskript der Vorlesung: Mustererkennung, Fachgebiet Computer Vision, Westfälische Wilhelms-Universität Münster. `http://cvpr.uni-muenster.de/teaching/ws08/mustererkennungWS08/script/ME02.pdf`, 2008. Letzter Zugriff: 12.08.2012. (Zitiert auf Seiten 10, 36, 52 und 54)

[51] Thorsten Joachims. Diplomarbeit: Einsatz eines intelligenten, lernenden Agenten für das World Wide Web, Fachbereich Informatik, Universität Dortmund. `http://www.cs.cornell.edu/People/tj/publications/joachims_96a.pdf`, 1996. Letzter Zugriff: 12.07.2012. (Zitiert auf Seite 80)

[52] Fernando Pereira John Blitzer, Ryan McDonald. Domain Adaptation with Structural Correspondence Learning. *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing EMNLP 06*, (July):120, 2006. (Zitiert auf Seite 87)

[53] Prof. Patrick Juola. Webseite: *Department of Mathematics and Computer Science Duquesne University, USA*, howpublished = `http://www.mathcs.duq.edu/~juola`, note = Letzter Zugriff: 05.02.2012, year = 2012. (Zitiert auf Seite 163)

[54] Dr. Sascha Kiefer. Literaturwissenschaft (Vorlesungsskript), Fachrichtung 4.1 Germanistik, Universität des Saarlandes. `http://www.uni-saarland.de/fak4/fr41/germanistik/pdfs/studieninfos/zp/zplw_script.pdf`, 2006. Letzter Zugriff: 21.05.2012. (Zitiert auf Seiten 109 und 110)

[55] Kim Kluckhohn. Kleines Glossar zur „Einführung in die Sprachwissenschaft“, Institut für Germanistik, Universität Leipzig. `http://www.uni-leipzig.de/~kluck/a1/glossar.htm`, 2012. Letzter Zugriff: 24.07.2012. (Zitiert auf Seiten 18, 19 und 20)

[56] Dr.-Ing. Jörn Kohlhammer and Dr. Tatiana von Landesberger. Informationsvisualisierung und Visual Analytics (Vorlesungsfolien), Kapitel: Datenrepräsentierung, Fraunhofer IGD/TU Darmstadt. `http://www.gris.tu-darmstadt.de/teaching/courses/ws1112/infvis/`, 2012. Letzter Zugriff: 30.07.2012. (Zitiert auf Seiten 10 und 64)

[57] Moshe Koppel, Navot Akiva, Idan Dershowitz, and Nachum Dershowitz. Unsupervised Decomposition of a Document into Authorial Components. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1356–1364, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. (Zitiert auf Seite 74)

[58] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Authorship Attribution in the Wild. *Lang. Resour. Eval.*, 45(1):83–94, March 2011. (Zitiert auf Seite 74)

[59] Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Yaron Winter. The „Fundamental Problem“ of Authorship Attribution. *English Studies*, 93(3):284–291, 2012. (Zitiert auf Seiten 85 und 86)

[60] Prof. Moshe Koppel. Webseite: *Department of Computer Science, Bar-Ilan University, Israel*. 2012. http://u.cs.biu.ac.il/ koppel. (Zitiert auf Seite 163)

[61] R. Layton, P. Watters, and R. Dazeley. Authorship Attribution for Twitter in 140 Characters or Less. In *Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Second*, pages 1–8. IEEE, 2010. Peter. (Zitiert auf Seite 74)

[62] L. Lemnitzer and H. Zinsmeister. *Korpuslinguistik: eine Einführung*. Narr Studienbücher. Narr, 2010. (Zitiert auf Seite 21)

[63] PD Dr. Hartmut Lenk. *Textsorten - Analysemethoden und Klassifikationen.* Germanistisches Institut der Universität Helsinki, 2006. (Zitiert auf Seite 20)

[64] IEEE Xplore Digital Library. Institute of Electrical and Electronics Engineers, New York City, USA. 2012. http://ieeexplore.ieee.org. (Zitiert auf Seiten 74 und 76)

[65] E. D. Liddy. *Natural Language Processing.* Encyclopedia of Library and Information Science, Morgan Kaufmann Publishers Inc., 2001. (Zitiert auf Seite 27)

[66] Dr. Martina Liedke-Göbel. *Angewandte Linguistik.* Einführung in die Germanistische Linguistik. Gabriele Graefen/ Martina Liedke:, 2007. (Zitiert auf Seite 80)

[67] Dr. Martina Liedke-Göbel. Linguistik und Grammatik (Syntax). `www.SprachenTrainer.net/mliedke/LuG/ab_syntax.doc`, 2012. Letzter Zugriff: 12.04.2012. (Zitiert auf Seite 62)

[68] Fabian Loose. Paarweise Autorenschaftsverifikation von kurzen Texten. Masterarbeit, Bauhaus-Universität Weimar, Fakultät Medien, Medieninformatik, May 2011. (Zitiert auf Seiten 59 und 60)

[69] Martin Lorenz and Jörg Kapfer. Einführung in Reguläre Ausdrücke, Computerlinguistik der Universität Erlangen. `http://www.linguistik.uni-erlangen.de/~cnweber/lectures/wacl/pdf/regexCLUE.pdf`, 2006. Letzter Zugriff: 12.02.2012. (Zitiert auf Seite 27)

[70] Eneldo Loza Mencía. Paarweises Lernen von Multilabel-Klassifikationen mit dem Perzeptron-Algorithmus. Master's thesis, TU Darmstadt, Knowledge Engineering Group, March 2006. Diplom. (Zitiert auf Seite 33)

[71] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008. (Zitiert auf Seiten 41, 42, 43, 44, 48 und 52)

[72] Schütze-H. Manning, C. *Foundations of Statistical Natural Language Processing.* MIT Press, Cambridge, MA, 1999. (Zitiert auf Seiten 10 und 38)

[73] Florian Markowetz. Genomische Datenanalyse - Klassifikation mit Support Vector Machines (Vorlesungsfolien), Max-Planck-Institut für Molekulare Genetik, Berlin Center for Genome Based Bioinformatics, Berlin. `http://lectures.molgen.mpg.de/statistik03/docs/Kapitel_16.pdf`, 2003. Letzter Zugriff: 21.06.2012. (Zitiert auf Seiten 10, 44 und 49)

[74] G.R. Mc Menamin and D. Choi. *Forensic Linguistics: Advances in Forensic Stylistics.* CRC Press, 2002. (Zitiert auf Seite 24)

[75] Jonathan Milgram, Mohamed Cheriet, and Robert Sabourin. „One Against One“ or „One Against All“. Which One is Better for Handwriting Recognition with SVMs? In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), October 2006. Université de Rennes 1, Suvisoft. http://www.suvisoft.com Université de Rennes 1. (Zitiert auf Seite 50)

[76] Shlomo Argamon Eran Messeri Moshe Koppel, Jonathan Schler. Authorship Attribution with Thousands of Candidate Authors. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 659–660, New York, NY, USA, 2006. ACM. (Zitiert auf Seiten 85 und 86)

[77] Arjun Mukherjee and Bing Liu. Improving Gender Classification of Blog Authors. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 207–217, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. (Zitiert auf Seite 83)

[78] Angela Orebaugh and Jeremy Allnutt. Data Mining Instant Messaging Communications to Perform Author Identification for Cybercrime Investigations. *Engineering*, pages 99–110, 2010. (Zitiert auf Seiten 89 und 90)

[79] K. Karteeka Pavan, Allam Appa Rao, A. V. Dattatreya Rao, and G. R. Sridhar. Robust Seed Selection Algorithm for k-Means Type Algorithms. *CoRR*, abs/1202.1585, 2012. (Zitiert auf Seite 54)

[80] Daniel Pavelec, Luiz S. Oliveira, Edson J. R. Justino, and Leonardo Vidal Batista. Using Conjunctions and Adverbs for Author Verification. *J. UCS*, 14(18):2967–2981, 2008. (Zitiert auf Seite 81)

[81] Dr. Thomas Petzoldt. Datenanalyse mit R - Ausgewählte Beispiele, Institut für Hydrobiologie, TU Dresden. `http://tu-dresden.de/die_tu_dresden/fakultaeten/fakultaet_forst_geo_und_hydrowissenschaften/fachrichtung_wasserwesen/ihb/studium/elements.pdf`, 2009. Letzter Zugriff: 19.08.2012. (Zitiert auf Seite 36)

[82] Prof. Dr. phil. Karl Heinz Wagner and Susanne Hackmack. Einführung in die Sprachwissenschaft (Vorlesungsfolien), Sprach- und Literaturwissenschaften (Fachbereich 10), Universität Bremen. `http://www.fb10.uni-bremen.de/khwagner/grundkurs1/pdf/grund.pdf`, 1998. Letzter Zugriff: 19.08.2012. (Zitiert auf Seiten 18 und 19)

[83] Anat Rachel Shimoni Prof. Moshe Koppel, Shlomo Argamon. Automatically Categorizing Written Texts by Author Gender, 2003. (Zitiert auf Seite 83)

[84] Jonathan Schler James W. Pennebaker Prof. Moshe Koppel, Shlomo Argamon. Mining the Blogosphere: Age, Gender and the Varieties of Self-Expression. *First Monday*, 12(9), 2007. (Zitiert auf Seite 83)

[85] Victor Raskin, Christian F. Hempelmann, and Katrina E. Triezenberg. Semantic Forensics: an Application of Ontological Semantics to Information Assurance. In *Proceedings of the 2nd Workshop on Text Meaning and Interpretation*, TextMean '04, pages 105–112, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. (Zitiert auf Seite 81)

[86] Dr. Uwe Reichel. Maschinelles Lernen II, Kapiel: Instanzbasiertes Lernen, Institut für Phonetik und Sprachverarbeitung, LMU München. `http://www.phonetik.uni-muenchen.de/~reichelu/kurse/machine_learning/machine_learning.html`, 2008. Letzter Zugriff: 12.05.2012. (Zitiert auf Seiten 10 und 64)

[87] Christof Rumpf. Einführung in die Computerlinguistik, Heinrich-Heine-Universität, Institut für Sprache und Information (Abteilung für Computerlinguistik), Düsseldorf. `http://user.phil-fak.uni-duesseldorf.de/~rumpf/talks/clintro2.pdf`, 2008. Letzter Zugriff: 21.08.2012. (Zitiert auf Seite 31)

[88] Dr. Gerold Schneider. Token, Types, Häufigkeiten und automatische Wortartenerkennung (Vorlesungsfolien), Kapitel: 6, Institut für Computerlinguistik der Universität Zürich. `http://files.ifi.uzh.ch/cl/gschneid/LexMorphVorl/Lexikon06.pdf`, 2000. Letzter Zugriff: 12.08.2012. (Zitiert auf Seite 19)

[89] Prof. Bernhard Schröder. Hauptpraktikum Computerlinguistik (Skript), Arbeitsbereich für Sprache und Kommunikation (Institut für Kommunikationswissenschaften), Universität Bonn. `http://www.sk.uni-bonn.de/lehre/informationen-materialien/informationen-und-materialien-kopho/materialien-1/schroder/hp-cl-b-1-im-ws-2000-2001/skript.pdf`, 2001. Letzter Zugriff: 22.08.2012. (Zitiert auf Seite 31)

[90] PD Dr. Matthias Schubert and Dr. Arthur Zimek. Knowledge Discovery in Databases (Vorlesungsskriptfolien), Database Systems Group, Department of Computer Science, Ludwig-Maximilians-Universität, München. `http://www.dbs.ifi.lmu.de/Lehre/KDD/WS0910/skript/kdd-5-clustering.pdf`, 2010. Letzter Zugriff: 21.08.2012. (Zitiert auf Seite 53)

[91] Jörg Schuster. Einführung in die Linguistik. `http://www.cis.uni-muenchen.de/people/schuster/cl1/skript.pdf`, 2003. Letzter Zugriff: 19.08.2012. (Zitiert auf Seite 22)

[92] Charles C. Tappert Seung-Seok Choi, Sung-Hyuk Cha. A Survey of Binary Similarity and Distance Measures. *Journal on Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010. (Zitiert auf Seiten 10, 36 und 37)

[93] Jonathon Shlens. A Tutorial on Principal Component Analysis, Systems Neurobiology Laboratory, Salk Insitute for Biological Studies La Jolla, CA 92037 and Institute for Nonlinear Science, University of California, San Diego La Jolla, CA 92093-0402. `http://www.snl.salk.edu/~shlens/pub/notes/pca.pdf`, 2005. Letzter Zugriff: 21.05.2012. (Zitiert auf Seite 70)

[94] Shlomo Levitan Shlomo Argamon. Measuring the Usefulness of Function Words for Authorship Attribution. 2005. (Zitiert auf Seiten 84 und 141)

[95] Bernhard Sowinski. *Deutsche Stilistik*. Fischer-Taschenbücher. Fischer Taschenbuch Verlag, 1973. (Zitiert auf Seite 23)

[96] Caroline Sporleder. Evaluation,Proseminar: Maschinelles Lernen und Experimentelles Design, Computational Linguistics, Universität des Saarlandes. `http://www.coli.uni-saarland.de/courses/MLED11/slides/evaluation.pdf`, 2011. Letzter Zugriff: 11.08.2012. (Zitiert auf Seite 57)

[97] SpringerLink. Springer-Verlag GmbH, Heidelberg, Germany. 2012. http://www.springerlink.com. (Zitiert auf Seiten 74 und 76)

[98] Efstathios Stamatatos and Moshe Koppel. Plagiarism and Authorship Analysis: Introduction to the Special Issue. *Language Resources and Evaluation*, 45(1):1–4, 2011. (Zitiert auf Seite 88)

[99] Prof. Efstathios Stamatatos. A Survey of Modern Authorship Attribution Methods. *J. Am. Soc. Inf. Sci.*, 60(3):538–556, 2009. (Zitiert auf Seiten 14, 59, 62, 73, 85, 91, 92, 94 und 98)

[100] Prof. Efstathios Stamatatos. Webseite: Dept. of Information and Communication Systems Engineering, University of the Aegean, Greece. `http://www.icsd.aegean.gr/lecturers/stamatatos`, 2012. Letzter Zugriff: 19.03.2012. (Zitiert auf Seite 163)

[101] Ulla Fix Steffen Pappert, Melanie Schröter. *Verschlüsseln, Verbergen, Verdecken in öffentlicher und institutioneller Kommunikation*. Philologische Studien und Quellen. Erich Schmidt, 2008. (Zitiert auf Seite 80)

[102] Benno Stein, Nedim Lipka, and Peter Prettenhofer. Intrinsic Plagiarism Analysis. *Lang. Resour. Eval.*, 45(1):63–82, March 2011. (Zitiert auf Seiten 73 und 81)

[103] Susanne Strecker and Stephanie Bösel. Füllwörter richtig vermeiden. `http://www.schreiblabor.com/orthography/filler`, 2012. Letzter Zugriff: 08. August 2012. (Zitiert auf Seite 63)

[104] David Martinus Johannes Tax. *One-Class Classification. Concept Learning In the Absence of Counter-Examples*. PhD thesis, Delft University of Technology, 2001. (Zitiert auf Seiten 99 und 100)

[105] Radka ŠTEPNICKOVÁ. Phraseologismen und die Fachsprache des Reiseführers. Vergleich der deutschen Ausgabe in Deutschland und Tschechien. Diplomová práce, Masarykova univerzita, Filozofická fakulta, 2006. (Zitiert auf Seite 80)

[106] Mercan Topkara, Cüneyt M. Taskiran, and Edward J. Delp. Natural Language Watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents*, pages 441–452, 2005. (Zitiert auf Seite 90)

[107] Jun. Prof. Dr. Achim Tresch. Machine Learning: Metriken für Nearest Neighbour-Verfahren/Lineare Diskriminanzfunktionen, (Lecture Slides). http://www.treschgroup.de/lecturestalks.html, Oktober 2007. (Zitiert auf Seiten 10 und 36)

[108] Dale Schuurmans Shaojun Wang Vlado Keselj, Fuchun Peng. Language Independent Authorship Attribution using Character Level Language Models. In *EACL '03: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, pages 267–274, Morristown, NJ, USA, 2003. Association for Computational Linguistics. (Zitiert auf Seite 82)

[109] N. Cercone C. Thomas Vlado Keselj, Fuchun Peng. N-Gram-based Author Profiles for Authorship Attribution. *Computational Linguistics*, 3:255–264, 2003. (Zitiert auf Seiten 80, 82, 88 und 95)

[110] Carmen Voigt. *Wer war es?* Linguistisches Kolloquium. Universität Erfurt, 2011. (Zitiert auf Seite 80)

[111] Wikipedia. Liste deutscher Redewendungen — Wikipedia, Die freie Enzyklopädie. `http://de.wikipedia.org/w/index.php?title=Liste_deutscher_Redewendungen&oldid=106538689`, 2012. Letzter Zugriff: 12.08.2012. (Zitiert auf Seite 63)

[112] Wiktionary. Liste der Anglizismen - Wiktionary. `http://de.wiktionary.org/wiki/Wiktionary:Deutsch/Liste_der_Anglizismen`, 2012. Letzter Zugriff: 19.08.2012. (Zitiert auf Seite 63)

[113] Prof. Dr. Erika Worbs. Phraseologismen, Zitate, Geflügelte Worte. Anmerkungen zu den Quellen der modernen Phraseologie. *Dieter Huber/Erika Worbs (Hg.). Ars transferendi. Sprache - Übersetzung - Interkulturalität. Peter Lang Verlag, Frankfurt am Main*, pages 261–272, 1998. (Zitiert auf Seite 63)

[114] Takahiro Yamada, Kyohei Yamashita, Naohiro Ishii, and Kazunori Iwata. Text classification by combining different distance functions withweights. In *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, SNPD-SAWN '06, pages 85–90, Washington, DC, USA, 2006. IEEE Computer Society. (Zitiert auf Seite 114)

[115] Ying Zhao and Justin Zobel. Searching With Style: Authorship Attribution in Classic Literature. In Gillian Dobbie, editor, *Thirtieth Australasian Computer Science Conference (ACSC2007)*, volume 62 of *CRPIT*, pages 59–68, Ballarat Australia, 2007. ACS. (Zitiert auf Seite 74)

[116] Zobel Justin Zhao, Ying. Effective and Scalable Authorship Attribution using Function Words. In *Proceedings of the Second Asia conference on Asia Information Retrieval Technology*, AIRS'05, pages 174–189, Berlin, Heidelberg, 2005. Springer-Verlag. (Zitiert auf Seite 84)

13 Anhang

13.1 Struktur der beiliegenden CD

Die CD die mit dieser Arbeit abgegeben wurde weist die folgende Verzeichnisstruktur auf:

- **Thesis Implementation:** Dieses Verzeichnis enthält den gesamten Quellcode des Frameworks, welches im Rahmen dieser Arbeit entwickelt wurde.
- **Wordlists:** Dieses Verzeichnis enthält mehrere Wortlisten, mit deren Hilfe Feature-Kategorien umgesetzt wurden.
- **Corpora:** Dieses Verzeichnis enthält die sieben Korpora die im Rahmen der Experimente kompiliert wurden. Die Korpora liegen sowohl in der reinen Textform, als auch in einer annotierten Form vor.
- **3rd Party Libraries:** Dieses Verzeichnis enthält sämtliche externe Werkzeuge/Bibliotheken, die innerhalb des Frameworks referenziert werden.
- **Feature-Matrices:** Dieses Verzeichnis enthält 7×13 Feature-Matrizen, mit denen die ML-Verfahren evaluiert wurden (7 Korpora, 13 Feature-Kategorien).

13.2 Bekannte Autoren auf dem Gebiet der Autorschaftsanalyse

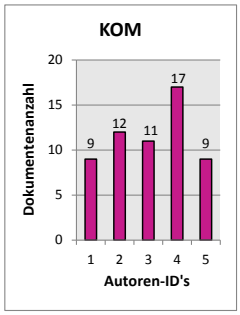
Auf Seite 73 wurde erwähnt, dass vier Autoren im Rahmen einer Recherche aufgefallen sind, die in den höchsten Rängen der Ergebnisse präsent waren. Im Folgenden werden diese vier Autoren vorgestellt, die auf dem Gebiet der Autorschaftsanalyse Pionierarbeit geleistet haben:

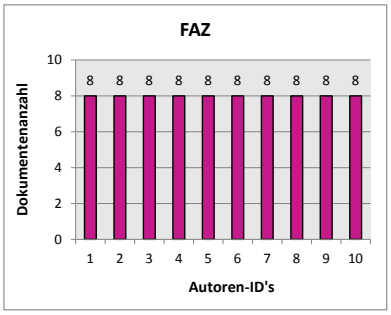
1. **Efstathios Stamatatos,** ist Assistant-Professor an der „University of the Aegean“, Griechenland. Stamatatos forscht seit über einem Jahrzehnt auf dem Gebiet der Autorschaftsanalyse (nicht nur in geschriebener Sprache) und hat über 40 wissenschaftliche Ausarbeitungen auf diesem Gebiet vorzuweisen, [100].
2. **Moshe Koppel,** ist Professor an der „Bar-Ilan University“, Israel. Koppel gilt als einer der aktivsten Forscher auf dem Gebiet der Autorschaftsanalyse. Er blickt zurück auf über 50 wissenschaftliche Beiträge auf diesem Gebiet, die teilweise einen religiösen Fokus aufweisen, [60].
3. **Patrick Juola,** ist Associate-Professor an der „Duquesne University“, Pennsylvania, USA. Juola hat ca. 40 wissenschaftliche Ausarbeitungen und 1 Buch in dem Gebiet der Autorschaftsanalyse publiziert. Dabei verbindet Juola unterschiedliche Disziplinen miteinander, wie beispieslweise Korpuslinguistik, Psycholinguistik, NLP, Kryptographie oder auch Pädagogik, [53].
4. **David I. Holmes,** ist Professor an dem „College of New Jersey“, USA. Holmes hat ca. 100 wissenschaftliche Ausarbeitungen sowie einige[1] Bücher auf dem Gebiet der Autorschaftsanalyse publiziert. Er blickt auf eine dreißig Jahre lange Erfahrung auf dem Gebiet der Stilometrie und hat an vielen Projekten rund um die Disziplin der Autor-Attribution mitgewirkt. Die Projekte reichen dabei von der Analyse englischer Literatur, Sprachstörungen, Psychologie bis hin zur Untersuchungen militärischer Geschichte. Allen Projekten galt die Gemeinsamkeit, anonyme Autoren von Manuskripten, Briefen oder Büchern ausfindig zu machen, [45].

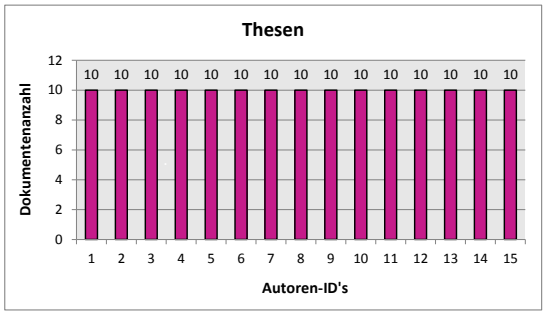
[1] Hier ist es leider unklar wieviele Bücher Holmes rausgebracht hat, da er offensichtlich nicht über eine eigene Webseite verfügt und seine Ausarbeitungen über unterschiedliche Verlage veröffentlicht wurden, sodass eine genau Zahl nicht ermittelt werden konnte.

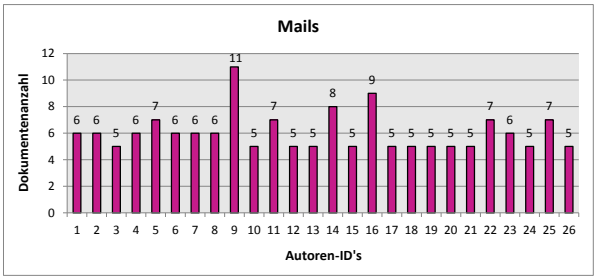
13.3 Dokumentverteilungen der Autoren für sämtliche Korpora

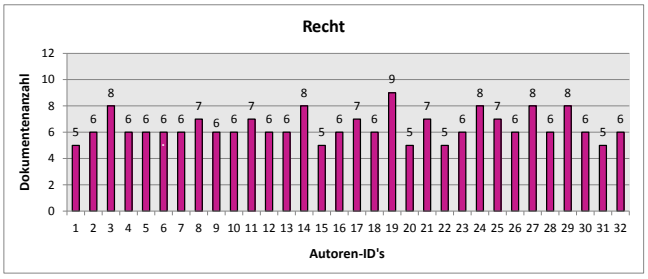
Die im Kapitel vorgestellten Korpora weisen die folgenden Dokumentverteilungen hinsichtlich der Autoren auf:

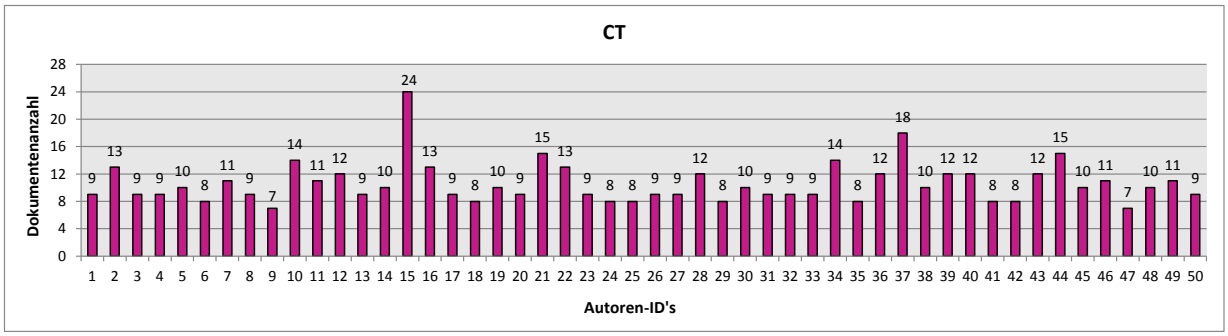


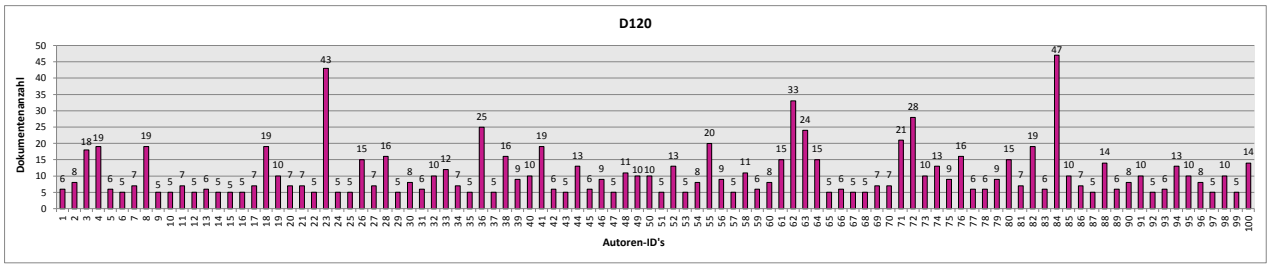












Aus den oberen Diagrammen kann entnommen werden, dass die beiden Korpora $\mathcal{K}_{FAZ}$ und $\mathcal{K}_{Thesen}$ *Instanz-balanciert* sind, während alle andere Korpora dagegen eine *Instanz-unbalancierte* Form aufweisen.

13.4 Rhetorische Stilmittel

Die folgenden rhetorischen Stilmittel wurden aus der Webseite `www.rhetoriksturm.de` entnommen (zuletzt zugegriffen am 04.07.2012):

Stilmittel:	**Beispiel:**
Alliteration	Gleicher Anfangslaut bei einer Reihenfolge (mindestens zwei) von Wörtern. „Fischers Fritze fischt frische Fische.“
Anapher	Wiederholung eines Satzteils am Anfang des Hauptsatzes bzw. des Nebensatzes. „Ich will Fussball spielen. Ich will etwas Essen“
Chiasmus	Überkreuzstellung von einzelnen Begriffen in einem Satz. „Innerhalb des Tages werde ich gewinnen, denn wenn ich nicht gewinne, wird der Tag sehr dunkel für mich“
Ellipse	Auslassung von einzelnen Wörtern, die für den Sinn des Satzes nicht entscheidend sind. „Alles klar?“ (Anstelle von „Ist bei dir alles Klar“?)
Enjambement	Zeilensprung; Der Satz endet nicht mit dem Vers, sondern ist Vers bzw. Strophenübergreifend. „Und so kam der Mann mit zwei Tieren im Schlepptau.“
Hyperbel	Starke Übertreibung. „Ich könnte ein ganzes Pferd verdrücken.“
Inversion	Veränderung der normalen Satzstellung (Subjekt, Prädikat, Objekt) „Der Hans ist des Tim's Bruder.“ (Ohne Inversion: Der Hans ist der Bruder Tim's)
Ironie	Übertriebene Bemerkung, die das Gegenteil meint. „Boah, du hast aber ne tolle Frisur.“ (In diesem Zusammenhang nur durch die sprachliche Betonung zu erkennen)
Klimax	Dreigliedrige Steigerung. „Meine Hütte, Mein Haus, Mein Schloss“
Metapher	Übertragende Bedeutung für ein oder mehrere Wörter, die in einem anderen Zusammenhang benutzt werden, aber das selbe Meinen. „Die Nadel im Heuhaufen suchen.“
Oxymoron	Wörter die sich sinnmäßig ausschließen. „Die Lebenden Toten.“
Personifikation	Unmenschliche Wörter (häufig Substantive) werden mit einem menschlichen Begriff in Verbindung gesetzt. „Das Auge des Gesetztes.“
Symbole	Ein Wort, dass Assoziation auslöst. (durch übertragende Bedeutungen) Grün = Hoffnung, Natur, Frieden
Verdinglichung	„Lebendige“ Nomen werden durch übertragende Bedeutungen in ein kollektiv gefasst und „entlebt“. Ein Haufen um ihn staunt und grinst voll Spott.(A. Wolfenstein, Die Stadt)
Vergleich	Zwei Bildbereiche werden durch das Wort „wie“ miteinander verknüft. „Die Liebe ist wie ein Fass ohne Boden.“

13.5 POS-Tagger: Tagset

Die folgende Tabelle beschreibt das POS-Tagset, welches in Rahmen dieser Arbeit verwendet wurde. Es handelt sich hierbei um das: „Stuttgart/Tübinger Tagset“ (kurz STTS).

Quelle: `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/stts.asc` (zuletzt zugegriffen am 04.07.2012):

POS-Tag	Beschreibung	Beispiele
ADJA	attributives Adjektiv	[das] große [Haus]
ADJD	adverbiales oder prädikatives Adjektiv	[er] fährt [schnell], [er ist] schnell
ADV	Adverb	schon, bald, doch
APPR	Präposition; Zirkumposition links	in [der Stadt], ohne [mich]
APPRART	Präposition mit Artikel	im [Haus], zur [Sache]
APPO	Postposition	[ihm] zufolge, [der Sache] wegen
APZR	Zirkumposition rechts	[von jetzt] an
ART	bestimmter oder unbestimmter Artikel	der, die, das, ein, eine
CARD	Kardinalzahl	zwei [Männer], [im Jahre] 1994
FM	Fremdsprachiges Material	[Er hat das mit „] A big fish [„ übersetzt]
ITJ	Interjektion	mhm, ach, tja
KOUI	unterordnende Konjunktion mit „zu“ und Infinitiv	um [zu leben], anstatt [zu fragen]
KOUS	unterordnende Konjunktion mit Satz	weil, daß, damit, wenn, ob
KON	nebenordnende Konjunktion	und, oder, aber
KOKOM	Vergleichspartikelm, ohne Satz	als, wie
NN	normales Nomen	Tisch, Herr, [das] Reisen
NE	Eigennamen	Hans, Hamburg, HSV
PDS	substituierendes Demonstrativpronomen	dieser, jener
PDAT	attribuierendes Demonstrativpronomen	jener [Mensch]
PIS	substituierendes Indefinitpronomen	keiner, viele, man, niemand
PIAT	attribuierendes Indefinitpronomen ohne Determiner	kein [Mensch], irgendein [Glas]
PIDAT	attribuierendes Indefinitpronomen ohne Determiner	[ein] wenig [Wasser], [die] beiden [Brüder]
PPER	irrefelexives Personalpronomen	ich, er, ihm, mich, dir
PPOSS	substituierendes Possessivpronomen	meins, deiner
PPOSAT	attribuierendes Possessivpronomen	mein [Buch], deine [Mutter]
PRELS	Relativpronomen (substituierend)	[der Hund,] der
PRELAT	Relativpronomen (attribuierend)	[der Mann,] dessen [Hund]
PRF	reflexives Personalpronomen	sich, einander, dich, mir
PWS	substituierendes Interrogativpronomen	wer, was
PWAT	attribuierendes Interrogativpronomen	welche [Farbe], wessen [Hut]

POS-Tag	Beschreibung	Beispiele
PWAV	adverbiales Interrogativ- oder Relativpronomen	warum, wo, wann, worüber, wobei
PAV	Pronominaladverb	dafür, dabei, deswegen, trotzdem
PTKZU	„zu“ vor Infinitiv	zu [gehen]
PTKNEG	Negationspartikel	nicht
PTKVZ	abgetrennter Verbzusatz	[er kommt] an, [er fährt] rad
PTKANT	Antwortpartikel	ja, nein, danke, bitte
PTKA	Partikel bei Adjektiv oder Adverb	zu [schnell]
TRUNC	Kompositions-Erstglied	An- [und Abreise]
VVFIN	finites Verb, voll	[du] gehst, [wir] kommen [an]
VVIMP	Imperativ, voll	komm [!]
VVINF	Infinitiv, voll	gehen, ankommen
VVIZU	Infinitiv mit „zu“, voll	anzukommen, loszulassen
VVPP	Partizip Perfekt, voll	gegangen, angekommen
VAFIN	finites Verb, aux	[du] bist, [wir] werden
VAIMP	Imperativ, aux	sei [ruhig !]
VAINF	Infinitiv, aux	werden, sein
VAPP	Partizip Perfekt, aux	gewesen
VMFIN	finites Verb, modal	dürfen
VMINF	Infinitiv, modal	wollen
VMPP	Partizip Perfekt, modal	[er hat] gekonnt
XY	Nichtwort, Sonderzeichen enthaltend	D2XW3
$,	Komma	,
$.	Satzbeendende Interpunktion	. ? ! ; ;
$(	sonstige Satzzeichen (satzintern)	- [] ()

Tabelle 61: Liste aller verfügbaren POS-Tags und deren Beschreibung

13.6 Parser: Tagset

Das Parser-Tagset welches innerhalb der Arbeit verwendet wurde kann auf der folgenden Webseite aufgerufen werden:

`http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/kanten.html` (zuletzt zugegriffen am 04.07.2012).