Leipzig Universität
Fakultät Mathematik und Informatik
Studiengang Informatik

Authorship Boosting
Ein Verfahren zur Bekämpfung der Autorschaftsverschleierung

Masterarbeit

Tobias Wenzel
geb. am: 22.05.1992 in Bad Kreuznach

Matrikelnummer: 3733301

1. Gutachter: jr. Prof. Dr. Martin Potthast
2. Gutachter: Dr. Thomas Efer

Datum der Abgabe: 18. Juni 2019

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann. Ich versichere, dass das elektronische Exemplar mit den gedruckten Exemplaren übereinstimmt

Leipzig, 18. Juni 2019

. . . . . .

Tobias Wenzel

Zusammenfassung

In der vorliegenden Arbeit wird untersucht, ob *Authorship-Boosting* [Bevendorff, 2016] nach einer Authorship-Obfuscation angewandt werden kann, um eine Rückführung der ursprünglichen Textrepräsentation zu erreichen und die darin eingesetzten Techniken in einem Authorship-Verification-Ansatz verwendet werden können. Authorship-Boosting beschreibt die Verstärkung von Stilfeatures eines Autors im Text einer anderen Person. Es wird gezeigt, dass die Boosting-Charakteristik eines Textpaares sich nach Obfuskierung von anderen Fällen unterscheidet, wenn für das Verfahren der ursprünglich für die Obfuskierung verwendete Text herangezogen wird. Der Obfuskierungs-Text ist somit erkennbar und kann als Schlüssel für die Obfuskierung angesehen werden, dessen Geheimhaltung essentiell wichtig für die Sicherheit einer Obfuskierung ist. Das Obfuskierungsverfahren wird dadurch grundsätzlich angreifbar.

Weiter wird untersucht, ob Authorship-Boosting als Komponente eines Authorship-Verification-Ansatz eingesetzt werden kann. Unmasking [Koppel und Schler, 2004] ist eines der effektivsten Authorship-Verification-Verfahren auf längeren Texten. Die Unmasking-Annahme ist, dass sich die Feature-Verteilungen von Textrepräsentationen des gleichen Autors einander schneller annähern, als die Feature-Verteilungen von Textrepräsentationen von verschiedenen Autoren, wenn iterativ die Wörter entnommen werden, die am stärksten zur Unterscheidung beitragen. Eine vergleichbare Annäherung kann erfolgen, indem Konzepte des Authorship-Boosting eingesetzt werden. Die Implementierung der alternativen Strategie erreicht in den Experimenten eine zum ursprünglichen Verfahren kompetitive Performanz.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

List of Algorithms

Abkürzungsverzeichnis

AA Authorship-Attribution

AV Authorship-Verification

AO Authorship-Obfuscation

AB Authorship-Boosting

SVM Support Vector Machine

KLD Kullback-Leibler-Divergenz

UM Unmasking

FR (Iterative) Feature Removal

FL (Iterative) Feature Leveling

Kapitel 1

Einleitung

Die Frage, welche Person ein bestimmtes Dokument geschrieben hat, beschäftigt die Forschung seit einiger Zeit. Ziel der Forschungsarbeiten war oft eine Lösung eines Literatur-Disputs [Bourne, 1897, Koppel und Schler, 2004]. Experten untersuchen dazu den Stil des festzustellenden Dokuments und vergleichen ihn mit anderen Dokumenten des vermeintlichen Autors. Durch die zunehmende Kommunikation über das Internet geraten mit Chats, Mails und Blog-Einträgen auch kürzere Texte in den Fokus der Forschung. Die Zahl der in Frage kommenden Autoren ist in diesem Fall in der Regel wesentlich höher [Koppel et al., 2011, Narayanan et al., 2012]. Ein Forensiker ist z. B. an der Identität eines Absenders einer verdächtigen Nachricht interessiert [Frommholz et al., 2016]. Computergestützte Verfahren sollen dem Forensiker dabei helfen, Autoren zuzuordnen. Man unterscheidet Authorship-Attribution, bei der ein Dokument einem bekannten Autor zugewiesen wird, von Authorship-Verification (AV), bei der festgestellt wird, ob zwei Dokumente vom gleichen Autor stammen. Jedes Authorship-Attribution-Problem lässt sich auf ein Authorship-Verification-Problem reduzieren.

Eine Person kann versuchen, ihren Text so zu verändern, dass Spuren ihrer Autorschaft verwischt werden, um automatische Verfahren und Experten zu täuschen. Das nennt man Authorship-Obfuscation (AO). In Ländern mit eingeschränkter Meinungsfreiheit kann AO eingesetzt werden, um restriktiven Behörden zu entgehen. Die Technik kann auch von Terroristen eingesetzt werden, um einer strafrechtlichen Verfolgung zu entgehen. Im letzten Jahrzehnt sind im Bereich der AO einige Forschungsarbeiten entstanden [Hagen et al., 2017]. Klassische Ansätze sind z. B. die in Rao und Rohatgi [2000] vorgeschlagene Round-Trip Übersetzung oder Austausch von aktiven und passiven Formen [Khosmood und Levinson, 2010]. Es hat sich gezeigt, dass bereits einfache Obfuskierer signifikante Resultate erzielen können [Potthast et al., 2016]. In Bevendorff [2016] wurde ein Obfuskierer

entwickelt, der die Repräsentation des zu verändernden Textes sukzessive durch gezielte Feature-Modifizierungen verfremdet. Eine vorgeschlagene Abwandlung des Algorithmus erlaubt es, das Indiz der Autorschaft zu verstärken oder sogar auf eine andere Person zu lenken. Dazu wird die Repräsentation des zu verändernden Dokuments langsam durch Erhöhen der Häufigkeit von ausgewählten Features an die Repräsentation eines Textes einer anderen Person angenähert. Der Autor bezeichnet das als Authorship-Boosting (AB). In der vorliegenden Arbeit sollen die Eigenschaften des Algorithmus untersucht, erläutert und erweitert werden. Insbesondere soll das Verhalten bei Einsatz von AB als Vorverarbeitung untersucht werden, das im Gegensatz zum ursprünglichen Ansatz eine Rückführung der Repräsentation anstrebt. Nähern sich Texte gleicher Autoren schneller an, ließe sich AB als Vorverarbeitungs-Schritt vor einer AV einsetzen und damit die Güte von bestehenden Lösungen verbessern. Nach Kenntnis des Autors ist ein solches Verfahren bisher nicht formuliert worden. Die Untersuchung in Kombination mit einem einfachen Klassifikator ist demnach ein relevanter Beitrag zur aktuellen Forschung auf diesem Gebiet. Zunächst soll das Verhalten des Klassifikators bei Texten des gleichen und verschiedener Autoren untersucht werden, auf die das Verfahren angewandt wird. Dazu wird die Performanz nach Anwendung des AB untersucht. Im Anschluss soll untersucht werden, ob sich das Verfahren als Gegen-Operation zur Obfuskierung einsetzen lässt und Rückschlüsse auf den Autor des Obfuskierungs-Textes ermöglicht. Es wird sich zeigen, dass der Obfuskierungs-Text als Schlüssel interpretiert werden kann, da sich die beobachtete Boosting-Charakterisik unter Verwendung des ursprünglichen Obfuskierungs-Textes signifikant von der Charakteristik mit anderen Texten unterscheidet.

Der in [Koppel und Schler, 2004] vorstellte Algorithmus *Unmasking (UM)* ist eines der erfolgreichsten AV-Verfahren für längere Texte. Die zugrunde liegende Annahme ist, dass der Stil eines Autors maßgeblich von einer geringen Anzahl an Features charakterisiert wird. Während die Wörter entfernt werden, die am meisten zur Trennung der Klassen beitragen, wird iterativ die Fähigkeit des Klassifikators ermittelt, die Texte zu trennen. Texte gleicher Autoren nähern sich dabei schneller an als Texte verschiedener Autoren. Es wird vermutet, dass durch AB eine ähnliche Annäherung erzielt werden kann. Auf dem AB-Algorithmus basierend soll ein Verfahren entwickelt werden, dass dieser Eigenschaft entspricht. Es wird auf verschiedenen Korpora evaluiert und gegen das ursprüngliche Verfahren verglichen.

Gliederung

Zunächst folgt ein Überblick über der Stand der Technik. Danach werden die benötigten Grundlagen der Authorship-Verification beleuchtet und die in den Versuchen verwendeten Korpora vorgestellt. Danach wird auf die Versuche mit AB als Vorverarbeitung und als Mittel gegen AO eingegangen. Anschließend wird die Entwicklung der auf AB basierenden Unmasking Strategie beschrieben und der Klassifikator evaluiert. Schließlich werden die Erkenntnisse der Arbeit zusammengefasst und ein Ausblick auf kommende Forschungsfragen gegeben.

Kapitel 2

Literaturüberblick

In diesem Abschnitt soll ein Überblick über den Stand der Technik in Authorship-Attribution, Authorship-Verification und Authorship-Obfuscation gegeben werden.

Authorship-Attribution und Authorship-Verification

Die Frage, welche Person einen Text geschrieben hat, beschäftigt die Wissenschaft seit mehr als einem Jahrhundert. Bourne [1897] versucht die Autorschaft der *Federalist Papers* durch Analyse von lingustischen Merkmalen und Kontext der vermeintlichen Person zu klären[1]. Mosteller und Wallace [1964] lösen das Problem erstmals mit Hilfe der Häufigkeiten der Wörter mit einer bayesschen Methode. Mit zunehmender Rechenleistung gewinnen Machine Learning Ansätze an Bedeutung [Koppel und Schler, 2009]. Da die Performanz von Authorship-Attribution-Verfahren bei einer steigenden Menge an Autoren abnimmt [Koppel et al., 2011], bietet sich die Lösung des AV-Problems an, deren Lösung es sich darauf übertragen lässt [Stamatatos, 2016]. Die zugrunde liegenden Modelle und Machine Learning Methoden sind oft ähnlich. Koppel und Schler [2009] geben einen guten Überblick über Authorship-Attribution, Stamatatos [2016] über Authorship-Verification.

Die zu untersuchenden Dokumente werden in der Regel als Vektoren einer Vielzahl von Features (deutsch: Merkmale) beschrieben [Abbasi und Chen, 2008]. Häufig werden Stopwörter [Potthast et al., 2018a, Stein et al., 2008], POS-Tags [Zhao und Zobel, 2007] oder Zeichen-N-Gramme [Koppel und Winter, 2014] eingesetzt[2]. N-Gramme haben sich als besonders effizient herausgestellt [Potha und Stamatatos, 2014]. Alternative Ansätze, wie die Projektion mittels Latent Sematic Indexing (LDI) in Unterräume [Potha und Stamatatos,

[1] Federalist Paper: Zeitungsartikel, die anonym erschienen, um das amerikanische Volk vom Föderalismus zu überzeugen.

[2] In dieser Arbeit wird N-Gramm als Synonym für Zeichen-N-Gramme verwendet.

2018] werden weniger häufig verwendet, erzielen aber ähnliche Erkennungsraten. Aus den Featurehäufigkeiten kann ein Ähnlichkeitsmaß berechnet werden. Häufig wird dazu die Kosinus- oder Min-Max-Ähnlichkeit verwendet [Kestemont et al., 2016, Koppel et al., 2011]. Die Kullback-Leibler-Divergenz (KLD) wird in wenigen Arbeiten erwähnt [Zhao und Zobel, 2007, Zhao et al., 2006]. Als Machine Learning Techniken werden Bayessche Regression [Koppel und Schler, 2009], Decision Trees [Abbasi und Hsinchun Chen, 2005], Random Forest Classifier [Delgado et al., 2014] und besonders Support Vector Machines eingesetzt [Abbasi und Hsinchun Chen, 2005, Diederich et al., 2003].

Dank der Shared Tasks von PAN[3] gewinnen die Themen Authorship-Attribution, Authorship-Verification [Juola und Stamatatos, 2013, Stamatatos et al., 2014, 2015], Authorship-Obfuscation [Hagen et al., 2017, Potthast et al., 2018b], an Popularität. Die Shared Tasks bieten Platformen zum Testen und Vergleichen von entwickelten Lösungen aus den Bereichen Authorship-Attribution (AA), AV und AO. Jeder Task enthält einen Korpus mit Problemen, auf denen das einzureichende Verfahren trainiert werden kann. Die Lösungen werden auf einem zurückgehalten Test Korpus mittels der Evaluations-Plattform TIRA[4] evaluiert [Gollub et al., 2012], was es ermöglicht, die eingereichten Lösungen der fortlaufenden Shared Tasks zu vergleichen. In den Jahren 2014 und 2015 wurde aus den eingereichten Klassifikatoren ein Ensemble-Klassifikator der eingereichten Klassifikatoren entwickelt, der die Performanz der einzelnen Verfahren bis auf wenige Ausnahmen übertraf[5]. Im Folgenden werden die Ansätze der Gewinner von in den Jahren 2013, 2014 und 2015 vorgestellt. Probleme des AV-Task von 2013 setzten sich aus mehreren bekannten und einem unbekannten Text zusammen. Seidman [2013] entwickelt eine Variante der *Imposter Methode* [Koppel und Winter, 2014]. Der Algorithmus sucht im Internet nach ähnlichen Dokumenten, den Impostern, und vergleicht die Ähnlichkeit der Dokumentenpaare gegen die der Imposter. Der resultierende *Score* der Dokumenten-Paare wird aufaddiert. Wenn der Score des Paars höher ist als der der Imposter, wird es der Klasse *gleicher Autor* zugeordnet. Khonji und Iraqi [2014] entwickeln den Ansatz weiter, indem sie das Min-Max Feature zur Berechnung des Scores anpassen, je nachdem wie ähnlich sich die Vektoren sind und ein größeres Set an Features verwenden. Ihre Laufzeit ist vergleichsweise lang.

Bagnall [2015] entwickelt einen AV-Ansatz, in dem ein rekurrentes neuronales

[3]Plagiarism Action Network/ Conference and Labs of the Evaluation Forum, kurz PAN/CLEF. Website: `https://pan.webis.de`.

[4]Testbed for Information Retrieval Algorithms, kurz TIRA. Website: `http://tira.webis.de`.

[5](Hard Voting) Ensemble Klassifikatoren verwenden verschiedene Klassifikatoren oder Untermengen des Trainings-Sets und votieren für die Klasse mit den meisten Stimmen.

Netz (RNN) für jeden Autor ein Sprachmodell lernt. Das Modell, das die meisten Übereinstimmungen mit dem Modell des untersuchten Textes zeigt, wird ausgewählt. Bagnall [2016] wendet den Ansatz auf das Authorship Clustering Problem an. Beim Authorship Clustering Problem ist nicht bekannt, wie viele Autoren es gibt.

AV und AA können in der Praxis eingesetzt werden, um Autoren von beleidigenden Nachrichten in sozialen Medien und terroristischen Texten zu enttarnen [Abbasi und Hsinchun Chen, 2005, Frommholz et al., 2016]. Historiker können sie einsetzen, um Hinweise auf die Autorschaft von Pseudonymen zu bekommen [Juola, 2013, 2015, Kestemont et al., 2016]. AV-Ansätze, die auf kürzeren Texte wie Chats oder Foren arbeiten, verwenden in der Regel komplexere Modelle [Rocha et al., 2017]. Abbasi und Hsinchun Chen [2005] analysieren Texte des Ku Klux Klan und Al-Quaeda in Web Foren und diskutieren die Unterschiede der verwendeten Features bei arabischen Texten. Aufgrund der geringen Datengrundlage, 20 Posts von 20 Autoren, sollten die guten Ergebnisse mit Vorsicht betrachtet werden. Lambers und Veenman [2009] benutzen ein kompressions-basiertes Verfahren. Der Ansatz ist, dass ähnliche Texte besser komprimiert werden. Die Autoren berechnen die Distanz zu Prototyp-Texten aus dem Trainings-Korpus und vergleichen ihn mit dem unbekannten Text. Koppel et al. [2011] untersuchen AA an einem Korpus aus 10 000 Blogeinträgen, wobei der korrekte Autor nicht vorhanden sein muss. Diese Problemstellung wird als Open Set Authorship-Attribution bezeichnet. Zur Lösung verwenden sie einen ähnlichkeitsbasierten Ansatz mit zufällig ausgewählten Features. Frommholz et al. [2016] entwickeln ein System, dass vor Cyberstalking schützen soll (Anti Cyberstalking Text-based System). AA wird hier eingesetzt, um bekannte Täter zu identifizieren, zuzuordnen oder Profile zu erstellen.

Unmasking [Koppel und Schler, 2004] ist einer der effektivsten AV-Algorithmen, wenn die zu untersuchenden Texte Buchlänge haben. Grundannahme ist, dass Texte gleicher Autoren durch eine geringe Anzahl an Features unterschieden werden können. Die Texte werden zunächst in Chunks (deutsch: Textabschnitte) aufgeteilt. Ein Klassifikator versucht die Mengen der Chunk-Repräsentationen zu trennen lassen. Dann werden die Features entfernt, mit denen sich die Mengen am leichtesten trennen. Chunk-Repräsentationen gleicher Autoren lassen sich schneller nicht mehr auseinanderhalten als Texte verschiedener Autoren. Ein *Meta-Klassifikator* lernt die Abnahme der Fähigkeit die Chunks der Chunk-Repräsentationen zu trennen und ermittelt dadurch, ob der Autor gleich ist. Die Effektivität von UM wurde von Kestemont et al. [2012] anhand eines Korpus aus Texten aus Dramen und Prosa untersucht. Die Autoren bestätigen die Effektivität von UM auf längeren Prosa Dokumenten, und bemerken eine weniger hohe Effektivität auf Texten, die Dramen

entnommen sind. Loose [2011] entwickelt ein Verfahren für UM für kurze Texte. Stein et al. [2008] stellen die Verbindung von AV zur Plagiatsprüfung her und erweitern theoretische Grundlagen von Unmasking. Es werden zudem alternative Strategien zu Meta Analyse vorgeschlagen und entwickelt. Potthast et al. [2018a] setzt das Verfahren zur Trennung von *Fake- und Real News*, sowie Satire gegen Fake und Real News ein. Die zu trennenden Mengen setzen sich aus den Dokumenten des jeweiligen Gebiets zusammen. Eine stärkere Abnahme der Trennbarkeit indiziert eine stärkere Ähnlichkeit des Themengebiets.

Beim Einsatz von AA und AV in der Praxis kann die Menge der Texte, die von unterschiedlichen Autoren stammen, beliebig groß werden. Im Gegensatz dazu existiert oft nur eine kleine Anzahl an Texten des gesuchten Autors (vgl. Abbasi und Hsinchun Chen [2005], Juola [2015], Koppel und Schler [2004]). Stamatatos [2008] zerlegt die Texte des Korpus in Textsegmente, die der Textmenge des jeweiligen Autors entsprechen und generiert dadurch ein ausgeglicheneres Klassifikations-Modell. Die Texte der Autoren werden konkateniert, nach Zeilen getrennt und anschließend zufällig mit Zurücklegen gezogen. Statt eines Klassifikators können auch mehrere Klassifikatoren eingesetzt werden, wobei der beste automatisch ausgewählt wird [Delgado et al., 2014]. In Escalante et al. [2009] wird dazu eine *Particle Swarm Model Selection* eingesetzt. Die Partikel enthalten die verwendeten Präprozesse, die Zusammensetzung der verwendeten Features sowie den Klassifikator. Zusammen werden diese als numerische Vektoren kodiert. Das Verfahren minimiert dabei den Klassifikationsfehler der Modelle.

Authorship-Obfuscation

Die oben vorgestellten Authorship Identification-Verfahren können von von repressiven Regierungen missbraucht werden, um die Meinungsfreiheit einzuschränken. Narayanan et al. [2012] untersuchen unterschiedliche Authorship Identification Techniken anhand eines Korpus aus 100 000 Blogeinträgen. Sie folgern aus der Effektivität ihrer Untersuchungen, dass AA und AV Risiken für Menschen darstellen, die auf Anonymität angewiesen sind. Authorship-Obfuscation (AO) beschreibt den Versuch, Spuren der Autorschaft in Texten unkenntlich zu machen, sodass weder computergestützte Verfahren noch Experten den Autor zuweisen können [Potthast et al., 2016]. Im Folgenden wird eine Auswahl an AO-Ansätzen vorgestellt, die gegen AV eingesetzt werden können. Potthast et al. [2016] gibt einen guten Überblick über den Stand der Technik. Kacmarcik und Gamon [2006] entwickeln einen AO-Ansatz und testen ihn gegen UM und andere AV Verfahren. Sie folgern aus ihren Experimenten, dass Klassifikatoren, die Repräsentationen mit wenigen Features verwenden, anfälliger bezüglich AO sind. Kacmarcik und Gamon [2006] schlägt

vor, eine *Round-Trip* Übersetzung, d. h. beispielsweise englisch → französisch → englisch, als Element von AO-Ansätzen zu verwenden. Khosmood und Levinson [2009] entwerfen ein Framework, dass Texte mit einer vorgegebenen Menge an Modifikationen dem Stil eines anderen Autors annähert, bis ein Schwellwert unterschritten ist.

Potthast et al. [2016] evaluieren drei AO-Ansätze gegen 44 AV-Verfahren und folgern, dass computergestützte AV-Ansätze bereits durch einfache AO-Ansätze in ihren Klassifikationen beeinflusst werden können. Sie definieren darüber hinaus die Metriken *Safety*, *Sensibleness* und *Soundness*, mit denen sich der Einfluss von AO-Ansätzen auf AV ermitteln und vergleichen lässt. Die Safety wird als Effektivität gegen AV-Verfahren definiert und entspricht damit der Fähigkeit, AV zu täuschen. Sie wird aus den Differenzen der Evaluationsmetriken auf dem Test-Korpus $\mathcal{D}_{\text{test}}$ (siehe Abschnitt 3.3) und dem obfuskierten Korpus $o(\mathcal{D}_{\text{test}})$ berechnet[6]. Soundness wird definiert als die Fähigkeit des Obfuskierers, den Text derart zu modifizieren, dass inhaltlich keine Veränderungen vorgenommen werden. Sensibleness wird definiert als die Fähigkeit Texte zu obfuskieren, die grammatikalisch richtig und unauffällig sind. Von den drei Metriken kann nur Sefaty automatisiert ermittelt werden, wohingegen Soundness und Sensibleness manuell evaluiert werden müssen.

Safety, Soundness und Sensibleness werden in den PAN Shared Tasks eingesetzt, um die eingereichten Obfuskierern zu evaluieren [Hagen et al., 2017, Potthast et al., 2018b]. Im Jahr 2017 verwendeten die eingereichten Lösungen regelbasierte und Sequence-to-Sequence Ansätze. Bakhteev und Khazov [2017] tauschen Kurz- und Langformen aus und ersetzen Synonyme. Castro-Castro et al. [2017] nehmen mittels eines Wörterbuchs und semantischen Ressourcen syntaktische und semantische Veränderungen vor und verkürzen Sätze nach syntaktischen Regeln. Der in Mihaylova et al. [2016] vorgestellte Ansatz ist auch im Jahr 2018 auf den meisten Korpora am effektivsten [Potthast et al., 2018b]. Der Ansatz verwendet ähnliche Text Transformationen wie in den zuvor vorgestellten Ansätzen und fügt zudem Rauschen ein. In Potthast et al. [2018b] werden Definitionen für AV und AO formuliert und eine Erweiterung der Auswertungen vorgestellt. Die Autoren entwickeln mit dem *obfuscator world ranking* eine Maßzahl, die es erlaubt, für einen AO-Ansatz die Metrik Safety für alle vorhandenen AV-Korpora respektive aller vorhandenen AV und deren AV Probleme zu berechnen. Die Maßzahl gewichtet Probleme höher, die von wenigen AV-Lösungen korrekt erkannt werden. Probleme, die von vielen AO-Ansätzen effektiv obfuskiert werden können, werden weniger stark mit einbezogen. Die Autoren stellen zudem fest, dass alle bisherigen Ansätze entweder sehr effektiv gegen computergestützte AV-Ansätze sind und aufgrund dessen viele textuelle Auffälligkeiten erzeugen oder sehr wenige Modifikationen

[6]Für die Metrik C@1 ergibt sich der Wert für einen AV-Ansatz y aus $\Delta_{\text{C@1}}(o, y, \mathcal{D}_{\text{test}}) = \text{C@1}(y, o(\mathcal{D}_{\text{test}})) - \text{C@1}(y, \mathcal{D}_{\text{test}})$.

vornehmen und deswegen wenig effektiv gegen AV sind.

Kapitel 3

Grundlagen

Im folgenden Kapitel werden die benötigten Grundlagen der Authorship-Verification, Authorship-Obfuscation, sowie Authorship-Boosting und Machine Learning beleuchtet und die Korpora der Versuche vorgestellt.

3.1 Authorship-Verification als Klassifikationsproblem

Die in dieser Arbeit verwendete Notation orientiert sich an Potthast et al. [2016]. Ein Authorship Identification Problem wird als Tupel $\langle d_u, D_A \rangle$ definiert mit d_u als Text mit unbekannter und D_A als ein Set mit Texten bekannter Autorschaft. Wenn D_A nur einen Autor enthält, so handelt es sich um ein Authorship-Verification Problem, andernfalls um ein Authorship-Attribution Problem. Abbildung 3.1 zeigt den Unterschied schematisch. Sei nun $\gamma(\langle d_u, D_A \rangle) \in A \cup \{\emptyset\}$ die wahre Zuordnung, die einem unbekannten Dokument d_u seinen Autor aus dem Set der bekannten Autoren A zuweist. Die Zuweisung von $\emptyset$ bedeutet, dass sich der wahre Autor nicht im Set der bekannten Autoren A befindet. Dann ist die Lösung des AV, eine Approximation $\nu \approx \gamma$ zu finden. Die Approximation ν wird auf einer Teilmenge der vorhanden Daten $\mathcal{D}_{\text{train}} \subset \mathcal{D}$ trainiert und auf $\mathcal{D}_{\text{test}} \subset \mathcal{D}$ evaluiert. In der Regel gilt $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$ und $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$

AV lässt sich als Ein-Klassen-Klassifikationsproblem definieren, in dem die Klasse mit bekannten Autorenschaften der Zielklasse entspricht. Alle anderen Texte werden als *Outlier* bezeichnet. Die Lösung des AV Problems ist schwerer als Authorship-Attribution, da die Outlier-Klasse alle Dokumente beinhaltet, die nicht vom Autor verfasst wurden. Es ist demnach schwer, die Klasse der Outlier zu beschreiben, ohne die Zielklasse mit einzubeziehen.

Zur Lösung des Problems im Feld der Informatik werden Machine Learning Ansätze eingesetzt. Beispiele für Algorithmen sind Naive Bayes Klassifikatoren, Decision Trees und

(a) Authorship-Attribution: Von welchem Autor ist das Dokument?

(b) Authorship-Verification: Ist die Person der Autor beider Texte?

Abbildung 3.1: Schematische Darstellung von Authorship-Attribution und Authorship-Verification. Die Bücher symbolisieren jeweils einen Text. Die Pfeile symbolisieren eine Zuordnung, die für die gestrichelten Pfeile ermittelt werden soll.

Neuronale Netze (vgl. Abschnitt 2). Im Feld der AV erzielt die Support Vector Machine häufig gute Ergebnisse [Koppel und Schler, 2009] und eignet sich infolgedessen als Klassifikator für die in der Arbeit verwendeten AV-Verfahren. Im Folgenden wird die Funktionsweise der Support Vector Machine erläutert.

3.2 Support Vector Machine (SVM)

Support Vector Maschines zeigen in AV-Lösungen eine hohe Effektivität und werden aufgrund dessen häufig eingesetzt. Die Grundlagen wurden in Vapnik und Chervonenkis [1963] beschrieben und über einen längeren Zeitraum entwickelt [Boser et al., 1992, Cortes und Vapnik, 1995] u. a. Im Folgenden werden die Grundlagen des Verfahrens, sowie Klassifikation und Auswertungen erläutert.

In Abbildung 3.2 ist eine Support Vector Machine (SVM) schematisch für zwei Features dargestellt. Die Grundidee des Algorithmus ist eine Hyperebene H_0 in den Featureraum zu legen, die ihn so aufteilt, dass neue Daten $\mathbf{x} \in \mathcal{D}_{\text{test}}$ mittels einer Entscheidungsfunktion bei linearer Separierbarkeit $\nu(\mathbf{w}^T\mathbf{x} + b) \Rightarrow \{0, 1\}$ der korrekten Klasse zugewiesen werden können. Der Klassifikator teilt den Featureraum im linearen Fall in zwei Teilbereiche auf. Während der Trainingsphase wird die Hyperebene H_0 ermittelt, sodass der Abstand zu den nächstliegenden Featurevektoren, den Support Vektoren, maximal wird. Je größer der Abstand gewählt werden kann, desto besser können die Klassen getrennt werden und demnach Elemente $\mathbf{x} \in \mathcal{D}_{\text{test}}$ korrekt zugewiesen werden. Anders als bei Neuronalen

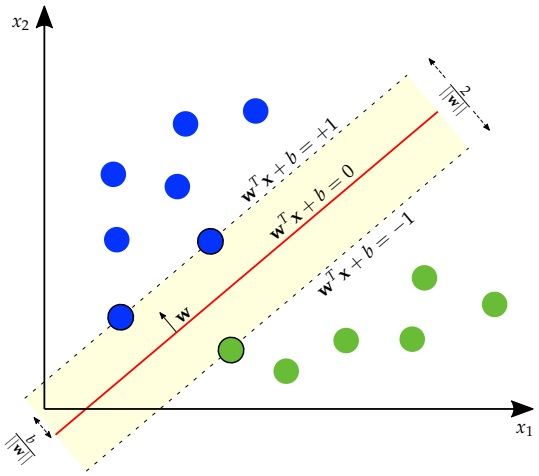


Abbildung 3.2: Schematische Darstellung der Support Vector Machine nach einer Graphik von Larhmam, veröffentlicht unter CC BY-SA 4.0 Lizenz. Die SVM legt eine Hyperebene H_0 (rot Line) in den Raum, welche die Klassen trennt. Die Hyperebenen H_1 und H_2 (gestrichelte Linien) schneiden die nächstliegenden Featurevektoren (Kreise mit Rand).

Netzen werden für das Modell nur die Punkte mit einbezogen, die sich nahe an der Entscheidungsgrenze befinden. Die Hyperebene folgt der Form:

$$H_0 : \mathbf{w}^T\mathbf{x} + b = 0. \tag{3.1}$$

Dabei ist $\mathbf{w}$ ein Gewichtsvektor, der orthogonal zu H_0 liegt und b ein Bias. Damit lässt sich eine Entscheidungsregel basierend auf dem Abstand zur Hyperebene H_0 als Kriterium definieren:

$$\begin{aligned} \mathbf{w}^T\mathbf{x} + b \geq 0 \text{ then } + \\ \mathbf{w}^T\mathbf{x} + b < 0 \text{ then } - \end{aligned} \tag{3.2}$$

Wenn der Wert der Entscheidungsfunktion eines Samples $\mathbf{x} \in \mathcal{D}_{\text{test}}$ größer gleich 0 ist, wird es der positiven Klasse zugewiesen, andernfalls der negativen Klasse. Der Betrag des Werts der Entscheidungsfunktion kann als Sicherheit der Klassifizierung interpretiert werden. Der Abstand zwischen den Elementen $\mathbf{x} \in \mathcal{D}_{\text{train}}$ und der Hyperebene H_0 wird während der Trainingsphase maximiert, sodass die neuen Daten $\mathbf{x} \in \mathcal{D}_{\text{test}}$ möglichst gut zugeordnet werden können. Dazu werden zwei weitere Hyperebenen definiert:

$$H_1 : \mathbf{w}^T\mathbf{x} + b = +1, \tag{3.3}$$
$$H_2 : \mathbf{w}^T\mathbf{x} + b = -1. \tag{3.4}$$

Die nächstliegenden Samples aus $\mathbf{x} \in \mathcal{D}_{\text{train}}$, auch genannt Support-Vektoren, sollen die Ebenen schneiden. Um H_0 zu bestimmen, werden die Parameter $\mathbf{w}$ und b benötigt. Es soll gelten, dass für $\mathbf{x} \in \mathcal{D}_{train}$ gilt:

$$\mathbf{w}^T\mathbf{x}_+ + b \geq +1, \tag{3.5}$$
$$\mathbf{w}^T\mathbf{x}_- + b \leq -1. \tag{3.6}$$

D. h. kein Sample des Trainings-Sets liegt zwischen H_1 und H_2. Die Definition von $y_i, i = 1, \ldots, |x|$ als $+1$ für positiv-Samples und -1 für negativ-Samples, erlaubt die Gleichungen (3.5) und (3.6) zusammenzufassen[1]:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geq 0. \tag{3.7}$$

Ein Featurevektor $\mathbf{x}_0 \in H_1$ hat zu H_2 den Abstand $\frac{2}{||w||}$ (siehe Abbildung 3.2). Daraus folgt, dass der Betrag von $\mathbf{w}$ minimiert werden muss. Durch Wegfall der Konstanten kann die Lösung der *Hard Margin* SVM als Minimierung von $||\mathbf{w}||$ beschrieben werden[2]. Das Problem lässt sich als quadratisches Problem formulieren, für dessen Lösung effiziente Lösungsalgorithmen existieren. Oft ist eine lineare Separierung nicht möglich, da die geforderten Beschränkungen in Gleichung (3.7) nicht eingehalten werden können und es somit zu einem unzulässigen Optimierungsproblem kommt, für das keine Lösung existiert. Mit dem Zulassen von Verletzungen ζ_i kann darauf reagiert werden (*Soft Margin*):

$$\begin{aligned} \min_{\mathbf{w},\, b,\, \zeta} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\zeta_i \\ \text{s. t.} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 + \zeta_i \geq 0 \qquad \zeta_i \geq 0 \\ & \qquad\qquad\qquad\qquad\qquad\qquad i = 1, \ldots, m. \end{aligned} \tag{3.8}$$

[1] Daraus folgt, dass zwischen H_1, H_2 gilt: $y_i(\mathbf{x}_i\mathbf{w} + b) = 0$.

[2] Hard Margin SVM Objective. Zwischen H_1 und H_2 werden keine Verletzungen von Gleichung (3.7) zugelassen: $\min_{\mathbf{w},\, b} \frac{1}{2}||\mathbf{w}||^2$ s. t. $y_i(\mathbf{x}_i\mathbf{w} + b) - 1 \geq 0$ for $i = 1, \ldots, m$.

Tabelle 3.1: Konfusionsmatrix.

Prädiziert \ Werte	Positiv	Negativ
Positiv	t_p	f_p
Negativ	f_n	t_n

Die Konstante C ist hierbei ein Regularisierungs-Hyperparameter, welcher einstellt, welches Gewicht der Verletzung der Beschränkungen beigemessen wird. In dieser Arbeit werden die Klassen `SVC` [Chang und Lin, 2011] und `LinearSVC` [Fan et al., 2008] aus der Python-Bibliothek scikit-learn [Pedregosa et al., 2011] verwendet.

3.3 Klassifikations-Güte

Die Güte von Klassifikatoren wird daran gemessen, wie hoch die Fähigkeit ist, Klassen korrekt zuzuweisen. Tabelle 3.1 fasst alle möglichen Fälle für eine Klasse zusammen. Es existiert eine Vielzahl an Maßzahlen, die jeweils Vor- und Nachteile haben und sich gegenseitig ergänzen können. Üblicherweise werden Precision und Recall für die zu klassifizierenden Klassen angegeben. Precision gibt die Reinheit der als korrekt klassifizierten Samples t_p an $\left(\frac{t_p}{t_p+f_p}\right)$. Eine Precision von 1 bedeutet, die Klasse wird nicht falsch zugewiesen. Recall dagegen gibt den Anteil der Samples der Klasse an, die korrekt klassifiziert werden $\left(\frac{t_p}{t_p+f_n}\right)$. Ein Recall von 1 bedeutet, dass alle Elemente der Klasse korrekt klassifiziert wurden. Die Werte lassen sich als harmonischer Mittelwert je Klasse zusammenfassen

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{3.9}$$

Als Güte für den gesamten Klassifikator wird Accuracy angegeben. Sie gibt die Trefferquote über alle Klassen $\frac{n_c}{n}$ mit n_c als Anzahl der korrekt klassifizierten Samples an und n als Summe aller Samples. Eine Accuracy von 1 bedeutet, alle Samples werden korrekt klassifiziert. Wenn die Klassen ungleich verteilt sind und alle Samples einer Klasse zugewiesen werden, wird die Accuracy hoch, obwohl der Klassifikator nicht gut zuordnet. Wenn die Klassen der Elemente in $\mathcal{D}$ ungleich verteilt sind, vlg. Koppel und Schler [2004], sollte *crossvalidiert* werden, um zu vermeiden, dass die Aufteilung das Ergebnis nicht verfälscht. Samples aus $\mathcal{D}_{train}$ und $\mathcal{D}_{test}$ werden dabei über N Iterationen ausgetauscht und die Accuracy als Mittelwert angegeben. In den PAN Workshops können Elemente

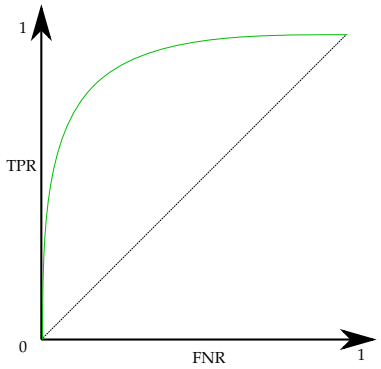


Abbildung 3.3: Schematische Darstellung von ROC Verläufen. Die schwarze Diagonale zeigt den Verlauf bei Evaluation eines Klassifikators, der zufällige Entscheidungen trifft. Der perfekte Klassifikator ist ein Punkt in der linken oberen Ecke. Die grüne Kurve zeigt den Verlauf bei Evaluation eines guten Klassifikators.

ausgelassen werden, der Wert C@1 wird leicht abweichend bestimmt [Peñas und Rodrigo, 2011]:

$$\text{C@1} = \frac{1}{n}\left(n_c + \frac{n_c}{n} \cdot n_u\right). \tag{3.10}$$

Die Zahl n_u entspricht der Anzahl der Samples für die keine Entscheidung getroffen wurde. Das Maß C@1 weist unbeantworteten Fällen dieselbe Accuracy zu dem Rest. Werden alle Samples klassifiziert $n_u = 0$ oder kein Sample klassifiziert $n_u = n$ entspricht das Maß der Accuracy. Damit werden Klassifikatoren bevorzugt, die ein Auslassen der Antwort einer falschen Antwort vorziehen.

Der ROC AUC Score (Area Under the Receiver Operating Characteristic Curve, kurz: AUC) [P. Bradley, 1997] wird seit 2014 in den PAN Workshops angegeben. Für ROC werden die Wahrscheinlichkeiten der Klasse benötigt. Der Schwellenwert, bei dem ein Sample einer Klasse zugewiesen wird, wird iterativ erhöht und True Positive Rate (kurz: TPR, auch: Recall) gegen die False Positive Rate (kurz: FPR) $\left(\frac{f_p}{f_p+t_n}\right)$ aufgetragen (siehe Abbildung 3.3). Diagonalen in der graphischen Darstellung deuten darauf hin, dass zufällig entschieden wird (vgl. schwarze Linie). Nach unten gekrümmte Kurven deuten darauf hin, dass der Klassifikator die Ergebnisse falsch interpretiert. Gute Klassifikatoren haben direkt einen hohe Recall, erst mit Steigerung der Schwellenwerts erhöht sich die False Positve Rate (vgl.

grüne Linie). Als Messwert wird die Fläche unterhalb der Kurve angegeben:

$$\text{AUC} = \int_{x=0}^{1} \text{TPR}(\text{FPR}^{-1}(x))\, dx. \tag{3.11}$$

Das Maß sagt aus, wie hoch die Wahrscheinlichkeit ist, dass der Klassifikator ein zufällig gewähltes positives Samples höher wertet als ein negatives Sample. In dieser Arbeit werden die Maße C@1 und AUC kombiniert als Güte angegeben (vgl. PAN Shared Tasks 2013, 2014 und 2015). Ein idealer Klassifikator hat demnach einen Wert von 1.

3.4 Ähnlichkeitsmaße

Es existieren eine Reihe an Metriken, um Vektor-Textrepräsentationen zu vergleichen. In ähnlichkeitsbasierten Klassifikatoren werden sie als Features eingesetzt. Im Vorfeld wurde eine Reihe von Metriken und deren Kombinationen empirisch an ausgewählten Korpora getestet. An dieser Stelle werden nur die gewählten Metriken vorgestellt. Eine oft eingesetzte Metrik ist die Kosinusähnlichkeit. Sie wird für zwei Vektoren **a** und **b** der Länge m als

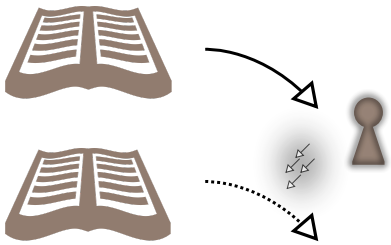
$$\cos(\theta) = \frac{p \cdot q}{\|p\|_2 \|q\|_2} = \frac{\sum_{i=1}^{m} a_i \cdot b_i}{\sum_{i=1}^{m} a_i^2 \cdot \sum_{i=1}^{m} b_i^2} \tag{3.12}$$

definiert und nimmt Werte im Bereich $[-1, 1]$ an. Das Argument θ entspricht dabei dem von **a** und **b** eingeschlossenen Winkel. Das Maß gibt damit an, wie stark zwei Vektoren in die gleiche Richtung zeigen. Nimmt es den Wert 0 an, liegen sie orthogonal und sind unabhängig. Im Rahmen dieser Arbeit liegt der Wertebereich im Intervall $[0, 1]$, da die Featurehäugikeiten a_i, b_i nur positive Werte annehmen können.

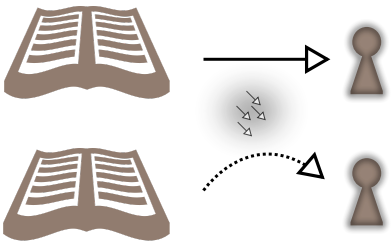
In dieser Arbeit soll ein Fokus auf die Kullback-Leibler-Divergenz (KLD) [Kullback und Leibler, 1951] (auch relative Entropie) gelegt werden, da diese einige nützliche Eigenschaften besitzt (siehe Abschnitt 3.5) und sich in vergangenen Arbeiten als effektiv herausgestellt hat [Bevendorff, 2016, Zhao und Zobel, 2007, Zhao et al., 2006]. Seien P, Q diskrete Wahrscheinlichkeitsverteilungen der Länge m der Features der Textpaars. Dann ist die KLD definiert als

$$\text{KLD}\,(P||Q) = \sum_{i=1}^{m} P[i] \log \frac{P[i]}{Q[i]}. \tag{3.13}$$

Eine alternative Definition ist durch $\text{KLD}\,(P||Q) = H(P, Q) - H(P)$ mit $H(P, Q)$ als



(a) Authorship-Obfuscation: Obfuskiere Autorschaft von Autor A.



(b) Authorship-Boosting: Verschiebe Autorschaft von Autor A auf Autor B.

Abbildung 3.4: Schematische Darstellung von Authorship-Obfuscation und Authorship-Boosting . Die Bücher symbolisieren jeweils einen Text des Dokumentenpaars. Die Autorschaft des oberen Dokuments, als Pfeil mit durchgezogener Linie dargestellt, ist bekannt. Die als gestrichelter Pfeil dargestellten Hinweise auf die Person werden mittels AO von der Person abgelenkt, bzw. mit AB auf die untere Person verschoben.

Crossentropy von P und Q und $H(P)$ als Entropy von P gegeben. Die KLD ist immer positiv nach oben unbeschränkt, da $H(P,Q) = H(P)$ gilt, wenn die Verteilungen gleich sind. In der Praxis liegen die Werte meist unter 1 [Bevendorff, 2016]. Die Summanden können negative Werte annehmen. Die KLD ist nicht symmetrisch. Um dem entgegenzuwirken kann Smoothing, vgl. Zhao et al. [2006], oder die symmetrische Jensen-Shannon-Divergence [Fuglede und Topsoe, 2004] eingesetzt werden [Bevendorff, 2018][3].

3.5 Boosting und Obfuskierung

In dieser Arbeit wird zwischen Authorship-Obfuscation (AO), kurz: Obfuskierung und Authorship-Boosting (AB) kurz: Boosting, unterschieden. Ersteres verschiebt Indizien der Autorschaft des Textes weg vom ursprünglichen Autor a, letzteres verschiebt diese von a auf einen zweiten Autor b (siehe Abbildung 3.4). Ein Obfuskierungs-Verfahren o kann als

$$o(\langle d_u, D_a \rangle) = (\langle \widetilde{d_u}, D_a \rangle) \tag{3.14}$$

formalisiert werden. Der Parameter $\widetilde{d_u}$ ist das veränderte Dokument. Beide Verfahren streben an, die Entscheidung von AVLösungen für Dokumentenpaare der Klasse

[3]Die Jensen-Shannon Divergence ist definiert als $\text{JSD}\,(P||Q) = \frac{1}{2}\text{KLD}\,(P||M) + \frac{1}{2}\text{KLD}\,(Q||M)$ mit $M = \frac{1}{2}P + Q$.

gleicher Autor $v(\langle d_u, D_a \rangle), u = a$ zu $\emptyset$ bzw. zu *unterschiedliche Autoren* werden zu lassen. Der veränderte Text soll den in Potthast et al. [2016] definierten Qualitätsmerkmalen entsprechen:

1. *Safety*: Robustheit gegen Verifikation und de-Obfuskierung,
2. *Soundness*: Textuelle Gleichheit des obfuskierten Textes,
3. *Sensibleness*: Grammatikale Richtigkeit und Unauffälligkeit.

Dies kann durch verschiedene Ansätze erreicht werden (siehe Abschnitt 2). Die Grundlagen des in dieser Arbeit verwendete Obfuskierers und des Boosting Verfahrens wurden in Bevendorff [2016] entwickelt und implementiert. Die in Abschnitt 3.4 vorgestellte Kullback-Leibler-Divergenz spielt in beiden Algorithmen eine wichtige Rolle: Die Parameter C_P und C_Q kodieren den Stil der Dokumente P bzw. Q. Die KLD wird als Maß verwendet, wie sehr sich die Stile unterscheiden. Stammen Texte vom gleichen Autor, wurde eine ähnliche Kodierung verwendet und der KLD-Wert liegt niedriger, als für Texte unterschiedlicher Autoren. Um ein Dokument zu obfuskieren, wird ein neues Dokument erzeugt und der KLD-Wert bezüglich des ursprünglichen Dokuments erhöht. Aufgrund der Qualitätsmerkmale kann der Text nicht komplett ausgetauscht oder gelöscht werden (sound). Zudem muss darauf geachtet werden, dass grammatikalisch richtige Änderungen vorgenommen werden, die unauffällig (sensible) aber effektiv (safe) sind. Um die KLD zu erhöhen, müssen ihre Summanden erhöht werden. Seien p, q die Wahrscheinlichkeiten von $P[i], Q[i]$. Dann ergibt sich für die Änderung von q bezüglich der Summanden die partielle Ableitung:

$$\frac{\partial}{\partial q}\left(p \log_2 \frac{p}{q}\right) = -\frac{p}{q \ln_2}. \tag{3.15}$$

Die Grenzwertbetrachtung zeigt, dass der Summand gegen unendlich strebt, wenn q gegen Null strebt:

$$\lim_{q \to 0} \frac{p}{q} = \infty. \tag{3.16}$$

Demnach kann die Summe der KLD durch Entnahme von Wörtern erhöht werden. Algorithmus 1 zeigt eines der in Bevendorff [2016] entwickelten Verfahren. Der Algorithmus wurde für ein Modell aus Zeichenfolgen (auch: N-Gramme) entwickelt, kann aber auf andere Features erweitert werden. Als Eingabe dienen die Vektoren der Verteilung p des zu obfuskierenden Textes und der Verteilung q, die einem weiteren Text der gleichen Person entnommen werden kann. Zunächst werden die Features so nach Größe der Summanden

Algorithm 1 Authorship-Obfuscation (Reduction) nach Bevendorff [2016]. Seien p und q die Vektoren mit den Häufigkeiten der Features in den Dokumenten d_k und d_u und max_iterations die Anzahl der Features, die modifiziert werden sollen.

Input: vec p, vec q, int max_iterations
Output: vec q

```
1: counter := 0
2: vec qs := sorted_desc(q, key := func(n): p[n]/q[n])
3: while counter < max_iterations do
4:    for each feature in qs do
5:       if qs[feature] > 1 then
6:          q[feature]−−
7:          break
8:       end if
9:    end for
10:   counter++
11: end while
```

sortiert. Dies stellt sicher, dass durch wenige Veränderungen eine hohe Wirkung erzielt werden kann und die Obfuskierung den Qualitätsmerkmalen entspricht. Der Reihe nach wird in q die Anzahl derjenigen Features reduziert, deren Anzahl über 1 liegt. Eine Reduktion eines Features q auf 0 würde zu einer Abnahme der KLD führen, da nur Features mit eingehen, die in beiden Texten vorkommen. Das geschieht für eine vorgegebene Anzahl an Iterationen max_iterations. Aufgrund der Beschränkung $q_i \geq 1$ kann die tatsächliche Anzahl der Iterationen darunter liegen. In Bevendorff [2018] wurde der Algorithmus um die Elemente einer heuristischen Suche und adaptives Stoppen bei ausreichender Obfuskierung erweitert. Für die adaptive Obfuskierung wird aus den Paaren der Klasse *unterschiedliche Autoren* das 50. Perzentil der Jensen-Shannon Distanzen[4] respektive ihrer Textlänge berechnet. Dieser Schwellwert wird mit $\epsilon_{0.5}$ bezeichnet. Die Obfuskierung für ein Dokument wird darauf folgend so lange fortgesetzt, bis der Schwellwert $\epsilon_{0.5}$ überschritten wird. Die Wahl eines höheren Perzentils resultiert in einer stärkeren Obfuskierung.

Eine alternative Variante fügt Features hinzu, anstatt sie zu entnehmen (siehe Algorithmus 6). N-Gramme, die nicht in q, aber in p vorkommen, werden auf 1 gesetzt. Die Sortierung von p sorgt dafür, dass N-Gramme vorne stehen, die stärker zur Summe betragen. Daraus folgt, dass neue Summanden hinzugefügt werden, die die KLD ebenfalls steigen lassen.

[4]Die Jensen-Shannon Distanz ist definiert als $\mathrm{JS}_\Delta(P, Q) = \sqrt{2 \cdot \mathrm{JSD}\,(P||Q)}$.

Algorithm 2 Authorship-Boosting nach Bevendorff [2016]. Seien p und q die Vektoren mit den Häufigkeiten der Features in den Dokumenten d_k und d_u und iterations die Anzahl der Features, die modifiziert werden sollen.

Input: vec p, vec q, int iterations
Output: vec q

```
counter := 0
while counter < iterations do
    vec qs := sorted_desc(q, key := func(n): p[n]/q[n])
    feature := first(qs)
    q[feature]++
    counter++
end while
```

Bevendorff schlägt vor, den Algorithmus so anzupassen, dass die KLD bzgl. q nicht maximiert, sondern bezüglich der Verteilung eines anderen Autors minimiert wird. Der Autor nennt das Authorship-Boosting . Damit verschiebt sich die Verteilung von p in Richtung der q. Die Grenzwertbetrachtung der partiellen Ableitung zeigt, dass der Summand kleiner wird, wenn q gegen unendlich geht:

$$\lim_{q \to \infty} \frac{p}{q} = 0. \tag{3.17}$$

Demnach kann die Summe der KLD durch gezieltes Hinzufügen von Wörtern reduziert werden. Algorithmus 2 zeigt eine vereinfachte Form des Algorithmus. q steht für den Vektor der Verteilung eines Textes, der nicht vom gleichen Autor stammt. Die Sortierung sorgt dafür, dass die Summanden mit dem höchsten Einfluss benutzt werden. In jedem Durchgang wird die Anzahl des Features mit dem höchsten Einfluss auf die KLD erhöht. Das Verfahren kann robuster formuliert werden, indem jeweils ein zufälliges Feature aus den m einflussreichsten erhöht wird.

3.6 Unmasking

Der in Koppel und Schler [2004] beschrieben Algorithmus Unmasking ist zum Zeitpunkt des Verfassens der Arbeit eine der effektivsten Methoden der AV für längere Texte. Das zugrunde liegende Verfahren soll im folgenden Abschnitt erklärt werden. Die Autoren stellen die These auf, dass Unterschiede von Textpaaren gleicher Autoren sich in einer relativ kleinen Anzahl an Features wiederspiegeln. Wenn zwei Werke vom gleichen Autor

Algorithm 3 Unmasking nach Koppel und Schler [2004]. Seien d_k und d_u die Texte der Eingabedokumente, D_u und D_k die zugehörigen Modelle der Chunks und $\mathbf{w}$ der Vektor mit den Gewichtungen der Features innerhalb der SVM.

Input: string d_k, string d_u, int iterations
Output: vec cv_accuracies

$features :=$ most_frequent_words(d_k, d_u, N)
$D_k :=$ chunk(d_k)
$D_u :=$ chunk(d_u)
$X := D_k\{features\} \cup D_u\{features\}$
counter $:= 0$
cv_accuracies $:= \{\}$
while counter $<$ iterations **do**
    cv_accuracies := cv_accuracies $\cup$ cv_accuracy(SVC, X)
    $features :=$ sorted_desc$(|\mathbf{w}_i|)[:6]$
    $X := X\{!features\}$
    counter++
end while

sind, dann zeigen sich die Unterschiede in einer geringen Anzahl an Features.

Das Verfahren ist in Algorithmus 3 gezeigt. Zunächst werden die Texte d_k, d_u in Chunks unterteilt. Aus jedem Chunk wird ein Modell aus den $N = 250$ häufigsten Wörtern in beiden Texten erstellt. Die Modelle der Chunks erhalten eine Zuweisung $y_i \in \{0, 1\}$ abhängig davon, in welchem Dokument sie vorkommen. Anschließend wird auf dieser Menge eine SVM 10-fach crossvalidiert[5]. Die Höhe der ermittelten Accuracy gibt an, wie gut sich die Sets der Chunk-Modelle auseinander halten lassen. Zunächst ist dieser Wert hoch, nahe 1 (vgl. Abbildung 3.5). Dann werden die Features entfernt, mit denen sich die Klassen am besten unterscheiden lassen. Das sind die, für deren Gewichtung $\mathbf{w}_i$ gilt

$$\begin{aligned} (f_1, \ldots, f_N) := \arg\max_{f_i, \forall i} \quad & \sum_{i=1}^{N} |\mathbf{w}_i| \cdot f_i \\ \text{s. t.} \quad & \sum_{i}^{N} f_i = c. \\ & f_i \in [0, 1] \end{aligned} \tag{3.18}$$

Die Autoren wählen $c = 6$ Features pro Iteration aus. Durch das Entfernen der Features

[5] In dieser Arbeit wird die Klasse `LinearSVC` [Fan et al., 2008] aus scikit learn [Pedregosa et al., 2011], Version 0.20.0, verwendet. Die unterliegende C Implementierung nutzt einen Zufallszahlengenerator zur Auswahl der Features der SVM. Die Ergebnisse unterscheiden sich deswegen und werden als Mittel von mehreren Durchgängen gezeigt.

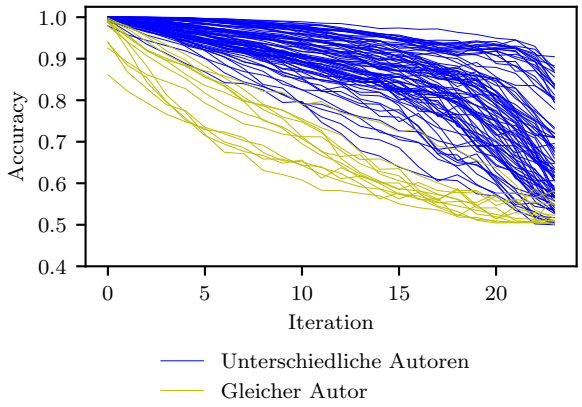


Abbildung 3.5: Reproduziertes Ergebnis aus [Koppel und Schler, 2004] auf dem KS04 Korpus. Accuracy: 0.978, F1-Score: 0.920.

nähern sich die Sets der beiden Dokumente einander an. Der Vorgang wird eine vorgegebene Anzahl an Iterationen wiederholt und die Accuracy-Werte werden gespeichert. Die so entstandenen Kurven können bereits sehr gut visuell unterschieden werden (vgl. Abbildung 3.5).

Im zweiten Schritt wird ein Meta-Klassifikator eingesetzt, der die Kurven von gleichen und verschiedenen Autoren trennen soll. Als Feature wird für jede Iteration i die Accurary, der Unterschied zur nächsten und übernächsten Accuracy, sowie der i-höchste Abfall zwischen einer und zwei Iterationen verwendet. Die SVM lernt wie sich die Kurven und deren Steigungen von gleichen Autoren im Unterschied zu Texten von unterschiedlichen Autoren unterscheiden lassen. Kurven von Texten desselben Autors fallen schneller ab.

Aufgrund der 10-fachen Crossvalidierung bei Verwendung von 250 Features, eignet sich das Verfahren nur für längere Texte. Die Anzahl der Wörter sollte dabei bei 10 000 liegen [Sanderson und Guenter, 2006]. In Loose [2011] wurde eine Unmasking-Variante entwickelt, die weitere Features und Ähnlichkeitsmaße mit einbezieht. In Bevendorff [2018] wurde ein Verfahren vorgeschlagen, das es ermöglicht, auch das ursprüngliche Unmasking zu verwenden. Dazu werden die Texte als Wordpool betrachtet und so lange zufällig Wörter gezogen, bis eine vorgegebene Chunk-Anzahl erreicht ist. Sofern die Methode eingesetzt wird, wird die Chunk-Anzahl auf 25 festgelegt. Um den Zufallseinwirkungen entgegenzuwirken wird über mehrere Durchgänge gemittelt. Weiter wird vorgeschlagen, die erzeugte Folge der Crossvalidierungs-Accuracy Werte in eine monoton fallende Folge umzuwandeln (monotonisieren), sodass für alle Werte eines Dokumentenpaars gilt:

$$\mathrm{cv}[i] > \mathrm{cv}[j] \quad \forall \quad 1 \leq i < \ldots < j < N. \tag{3.19}$$

Durch die Monotonisierung wird eine implizite Glättung vorgenommen.

Tabelle 3.2: Korpus Übersicht. Die Wortanzahl W wird als Durchschnitt über den Korpus gezeigt. N entspricht der Anzahl an Problemen im Korpus. Der Korpus JB16 BO 1 entspricht den Problemen der Klasse *gleicher Autor* des Test-Korpus von JB16 und erweitert die Probleme um einen weiteren Text des gleichen Autors. Der Korpus JB16 BO 2 ist eine Erweiterung des Test-Korpus von JB16 um einen weiteren Text eines anderen Autors.

Korpus	Trainings-Korpus		Test-Korpus		Gesamt		Quelle
	N	W	N	W	N	W	
KS04	–	–	–	–	202	10 000	[Koppel und Schler, 2004]
PAN14	100	3 137	200	6 404	300	4 770	[Stamatatos et al., 2014]
PAN15	100	366	500	526	600	446	[Stamatatos et al., 2015]
KW14	1 000	500	500	500	1500	500	[Koppel und Winter, 2014]
JB16	192	3 878	80	3 880	274	3 879	[Bevendorff, 2016]
JB16 BO 1&2	–	–	40	3 866	40	3 866	Diese Arbeit

3.7 Korpora

Im folgenden Abschnitt sollen die Korpora vorgestellt werden, die zur Evaluation eingesetzt werden. Ein AV Korpus ist eine Zusammenstellung von Dokumentenpaaren, deren Autorschaft bekannt ist. Üblicherweise existiert ein Trainings- und Test-Korpus. Eine vollständige Auflistung der im Rahmen der Arbeit verwendeten Korpora ist in Tabelle 3.2 gezeigt. Es existiert eine Anzahl an Korpora, die bei PAN Shared Tasks eingesetzt werden. Für die Evaluation wurden der PAN14 Novel Korpus (kurz PAN14) und der englische PAN15-Korpus ausgewählt (kurz PAN15), da zahlreiche Klassifikatoren auf ihnen aufgewertet und verglichen worden sind [Stamatatos et al., 2014, 2015]. Gleiche und ungleiche Autoren kommen gleich häufig vor. PAN14 ist eine Zusammenstellung aus 100 Werken des Autors Howard Phillips Lovecraft und von Fans geschriebenen Fortsetzungen. Das verwendete Vokabular ist ausgefallen, thematisch liegen die Texte z. T. nahe beieinander. Das birgt die Gefahr, dass Klassifikatoren lernen, Autoren anhand Themen einzuordnen. Trainings- und Testsets werden zu $|\mathcal{D}_{\text{train}}| = 100$ und $|\mathcal{D}_{\text{test}}| = 200$ Problemen aufgeteilt. Die Testdokumente sind mit 6 104 Wörtern durchschnittlich doppelt so lang wie die Trainingsdokumente mit 3 138. Für die Zusammenstellung der Probleme wurden die Dokumente mehrfach verwendet, sodass der Trainings-Korpus aus 57 und der Test-Korpus aus 43 Dokumenten besteht. Zudem existieren fünf Probleme, deren Texte lediglich vertauscht worden sind. Der PAN15-Korpus enthält kurze Dialoge aus Bühnenstücken ohne genauere Begrenzung des Themas. Trainings- und Testsets werden zu $|\mathcal{D}_{\text{train}}| = 100$ und $|\mathcal{D}_{\text{test}}| = 500$ Problemen geteilt. Die Testdokumente sind mit 526 Wörtern durchschnittlich etwas länger als die Trainingsdokumente mit 366. Beide PAN Korpora bilden keine optimale Basis zur Entwicklung neuer AV-Verfahren, bieten aber eine Möglichkeit zum direkten Vergleich und werden aufgrund dessen miteinbezogen. Bei der

Entwicklung neuer Korpora besteht dementsprechend weiterer Forschungsbedarf.

In Bevendorff [2016] wurde ein Korpus aus Auszügen aus Büchern des Project Gutenberg erstellt, der viele Themen abdeckt und damit ein realistischeres Szenario widerspiegelt[6]. Er wird im Folgenden als JB16 bezeichnet. Die Strukturierung der Fälle erfolgt nach dem PAN-Schema. Train und Testsets werden zu $|\mathcal{D}_{\text{train}}| = 182$ und $|\mathcal{D}_{\text{test}}| = 80$ geteilt. Anders als in den zuvor vorgestellten Korpora wird jedes Dokument nur einmal verwendet. Die durchschnittliche Wort Anzahl liegt in beiden Korpora bei 3 880. In Abschnitt 4 wird dieser Korpus in den Fällen mit gleichem Autor um einen dritten Text mit gleichem bzw. verschiedenen Autor erweitert, der ebenfalls Project Gutenberg entnommen wurde und eine vergleichbare durchschnittliche Wort-Anzahl hat (JB16 BO 1&2).

Im Rahmen dieser Arbeit sollen die entwickelten Klassifikatoren im ersten Teil der Arbeit mit denen der in Koppel und Winter [2014] verglichen werden, denen sie nachempfunden sind (siehe Abschnitt 4.1). Aufgrund dessen wurde der verwendete Korpus aus Schler et al. [2006] reproduziert. Die aus Foren entnommenen Texte sind mit 500 Wörtern ebenfalls sehr kurz und enthalten zahlreiche Rechtschreibfehler und Unregelmäßigkeiten. Trainings- und Testsets werden zu $|\mathcal{D}_{\text{train}}| = 1\,000$ und $|\mathcal{D}_{\text{test}}| = 500$ geteilt. Auch hier werde alle Dokumente nur einmal verwendet, mit der zusätzlichen Einschränkung, dass jeder Autor nur in einem Problem vorkommt. Im Folgenden wird der Korpus mit KW14 bezeichnet.

Ein weiterer Korpus ist Koppel und Schler [2004] entnommen. Im Folgenden wird er mit KS04 bezeichnet. Er besteht aus Büchern von neun Autoren des Project Gutenberg. Die durchschnittliche Wort Anzahl beträgt 10 000. Die Texte eignen sich aufgrund der Länge besser für UM.

Die Auswahl der Korpora ermöglicht die Evaluierung eines breiten Spektrums von Theaterstücken, Büchern, sowie Forumsbeträgen. KW14 und PAN15 werden wegen der Kürze der Texte nur zur Evaluation der Baseline-Klassifikatoren verwendet. Wegen der ungleichen Verteilung in KS04 von gleichen und ungleichen Textpaaren wird nur crossvalidiert.

[6] Project Gutenberg: `https://www.gutenberg.org/`. *Anmerkung*: Zum Zeitpunkt der Arbeit ist die Seite in Deutschland seit dem 1. März 2018 nicht mehr abrufbar. Aufgrund dessen wurden nur Dokumente verwendet, die nicht mehr urheberrechtlich geschützt sind und sich bereits vor der Sperrung in Besitz des Autors befanden.

Kapitel 4

Boosting als Klassifikationverbesserung

Das in Abschnitt 3.5 vorgestellte Authorship-Boosting soll in Dokumenten Indizien der Autorschaft einer Person verstärken, die ungleich der des tatsächlichen Autors ist. Dazu wird die Repräsentation des zu boostenden Dokuments langsam durch Erhöhen der Häufigkeit von ausgewählten Features an eine dazu verschiedene Repräsentation angenähert. In den folgenden Versuchen soll mit einem einfachen Klassifikator überprüft werden, ob nach einer Vorverarbeitung durch Authorship-Boosting ähnliche Autoren leichter als *gleicher Autor* klassifiziert werden, ohne das die Zahl der False Positives zu hoch wird (siehe Abschnitt 4.2). Des Weiteren wird untersucht, ob Obfuskierungen rückgängig gemacht werden können (siehe Abschnitt 4.3). Für die Versuche wird vorrangig der JB16-Korpus verwendet, da sich dieser Korpus, wie in Abschnitt 3.7 erläutert, am besten zur Auswertung eignet. Es müssen keine Veränderungen der Texte vorgenommen werden. Die Veränderungen der Repräsentationen entspricht nicht dem Original, wenn AB als Präprozess eingesetzt wird, bzw. sie stellt nur eine Approximation dessen dar, wenn sie nach der AO angewandt wird:

$$\text{boosting}(\langle d_u, D_a \rangle) = (\langle \widehat{d_u}, D_a \rangle). \tag{4.1}$$

Die Versuche werden wie folgt durchgeführt: Nachdem der Klassifikator auf dem Trainings-Korpus $\mathcal{D}_{\text{train}}$ trainiert und auf dem Test-Korpus $\mathcal{D}_{\text{test}}$ evaluiert wurde, werden in den Textrepräsentationen der unbekannten Texte aus $\mathcal{D}_{\text{test}}$ je eine Anzahl von n Features nach Algorithmus 2 geboostet. Wenn nicht anders definiert, ist $n = 10$. Aus den veränderten Repräsentationen wird ein neues Modell erstellt und der Klassifikator auf diesem Modell erneut evaluiert. Dieser Vorgang wird wiederholt durchgeführt, wobei Unterschiede zum ursprünglichen Modell für jede Iteration ausgegeben werden, um Änderungen in der Evaluation zu untersuchen. Hierfür werden die Anzahl der True Positives und False

Positives der Klasse *gleicher Autor* ermittelt. Da sich die Repräsentation annähern, können Zuweisungen der Klasse *unterschiedliche Autoren*, d. h. False Negatives und True Negatives, nicht hinzukommen.

Wie in Abschnitt 3.2 beschrieben, wird während des Trainings der SVM eine Hyperebene bestimmt, die die Samples der beiden Klassen, *gleicher Autor* und *unterschiedliche Autoren* trennt. Als Entscheidungsfunktion wird der Abstand zur Hyperebene bestimmt, wobei das Vorzeichen die Klasse bestimmt. Der Wert der Entscheidungsfunktion für ein unbekannte Sample $\mathbf{w}^T\mathbf{x} + b$ kann als Konfidenz interpretiert werden. Je höher der Wert im positiven Bereich liegt, desto höher ist die Konfidenz, dass es sich um den gleichen Autor handelt. Analog gilt, je höher der Wert im negativen Bereich liegt, desto höher ist die Konfidenz, dass es sich um unterschiedliche Autoren handelt. Die Auswirkung des Authorship Boostings auf die Features des Klassifikators werden beispielhaft graphisch dargestellt zwischen den Iterationen und anhand der Abstände zur Hyperebene sichtbar gemacht. Tabellarische Ansichten finden sich in Anhang B.

4.1 Baseline-Klassifikator

Im Folgenden werden Auswahl und Aufbau des verwendeten Klassifikators vorgestellt. Es existiert ein breites Spektrum an Verfahren, die für die Untersuchungen der Arbeit herangezogen werden können (siehe Abschnitt 2). Um Authorship-Boosting bestmöglich untersuchen zu können, wird ein einfaches AV-Verfahren entworfen. Es wird ein Klassifikator benötigt, das leicht interpretierbar ist und sich trotzdem mit dem Stand der Technik vergleichen lässt.

In Koppel und Winter [2014] werden zwei Methoden beschrieben, die als *Baseline*, eingesetzt werden. Die in dieser Arbeit verwendeten Klassifikatoren orientieren sich an diesen Methoden. Zunächst werden die Textpaare eingelesen und normalisiert. Beim Normalisieren werden z. B. Kommata, Zahlen oder Rollen in Dramen entfernt bzw. angeglichen. Der Einfluss von seltenen Zeichen und zufälligen Unterschieden soll dadurch verringert werden. Als Textrepräsentation werden die relativen Häufigkeiten (tf) der häufigsten 100 000 4-Gramme im Korpus mit ihrer inversen Dokumentenhäufigkeit (idf) multipliziert verwendet[1]. Diese Repräsentation wurde der von 3-Grammen vorgezogen, um

[1]Die inverse Dokumentenhäufigkeit für einen Term t ist definiert als $\text{idf}_t = \log\left(\frac{N}{\sum_{D:t\in D} 1}\right)$. Koppel und Winter geben nicht an, auf welchem Korpus das Maß berechnet wurde. In dieser Arbeit wurden der Brown-Korpus [Francis und Kučera., 1979], der Reuters-Korpus [Lewis, 2004] und der Gutenberg-Korpus verwendet. Letzter hat keine Übereinstimmungen mit JB16.

einen besseren Vergleich mit den von Koppel und Winter [2014] verwendeten Basline-Klassifikatoren zu erzielen. Die Performanz auf den Korpora liegt leicht unter der von 4-Grammen[2]. In dieser Arbeit wurde auch die Performanz auf tf untersucht. Die tf ist korpusabhängig, da keine Normalisierung bezüglich hochfrequenter Features vorgenommen wird.

Der erste Klassifikator lernt die Unterschiede der absolute Differenz, $|\mathrm{d}_k - \mathrm{d}_u|$ zwischen den Repräsentationen der Texten des gleichen Autors und Texten unterschiedlicher Autoren. Dieser Ansatz wird in dieser Arbeit als Absolute Difference Classifier (AD-C) bezeichnet. Der zweite Klassifikator verwendet die Kosinusähnlichkeit (Cos) der Repräsentationen. In dieser Arbeit wird zusätzlich die KLD als Feature verwendet. Im Folgenden wird dies mit Similarity Based Classifier (SIM-C) bezeichnet. Für beide Ansätze wird eine SVM verwendet[3].

[2]Die Ergebnisse des Klassifikators SIM-C mit 3-Grammen liegen mit C@1 · AUC auf JB16: 0.41, PAN14: 0.21 und PAN15: 0.30 unterhalb der Ergebnisse mit 4-Grammen, vgl. Tabelle 4.1.

[3]Für diesen Klassifikator wird die Klasse `SVC` [Chang und Lin, 2011] aus scikit learn [Pedregosa et al., 2011], Version 0.20.0, verwendet.

Tabelle 4.1: Evaluation der Baseline-Klassifikatoren. Die jeweils besten Ergebnisse je Korpus sind fett gedruckt. Die Ergebnisse der kursiv gedruckten Klassifikatoren entsprechen den Baseline-Klassifikatoren von Koppel und Winter [2014].

Korpus	Klassifikator	idf	C@1	AUC	C@1 · AUC
PAN14	Absolute Difference Classifier	✗	0.57	0.50	0.29
PAN14	Absolute Difference Classifier	✓	0.56	0.50	0.28
PAN14	Similarity Based Classifier	✗	0.54	0.66	0.35
PAN14	Similarity Based Classifier	✓	0.53	**0.69**	0.36
PAN14	Cosinus Classifier	✗	0.52	0.66	0.33
PAN14	Cosinus Classifier	✓	**0.60**	0.69	**0.39**
PAN15	Absolute Difference Classifier	✗	0.53	0.50	0.26
PAN15	Absolute Difference Classifier	✓	0.51	0.50	0.25
PAN15	Similarity Based Classifier	✗	0.56	0.69	0.38
PAN15	Similarity Based Classifier	✓	0.58	**0.71**	**0.41**
PAN15	Cosinus Classifier	✗	0.60	0.27	0.16
PAN15	Cosinus Classifier	✓	**0.64**	0.32	0.20
JB16	Absolute Difference Classifier	✗	0.51	0.31	0.16
JB16	Absolute Difference Classifier	✓	0.51	0.31	0.16
JB16	Similarity Based Classifier	✗	**0.70**	0.71	**0.50**
JB16	Similarity Based Classifier	✓	0.68	**0.72**	0.38
JB16	Cosinus Classifier	✗	0.51	0.31	0.16
JB16	Cosinus Classifier	✓	0.60	0.68	0.41
KW14	Absolute Difference Classifier	✗	0.50	0.26	0.37
KW14	Absolute Difference Classifier	✗	0.50	0.70	0.13
KW14	*Absolute Difference Classifier*	✓	**0.80**	–	–
KW14	Similarity Based Classifier	✗	0.73	0.80	0.59
KW14	Similarity Based Classifier	✓	0.75	**0.81**	**0.61**
KW14	*Cosinus Classifier*	✓	0.71	–	–
KW14	Cosinus Classifier	✗	0.72	0.78	0.57
KW14	Cosinus Classifier	✓	0.73	0.80	0.59

Tabelle 4.1 zeigt eine Übersichtstabelle der Evaluation der Klassifikatoren auf den in Abschnitt 3.7 vorgestellten Korpora. Auf KS04 wird auf Grund seiner Größe und des Fehlens des Test-Sets verzichtet. Die Accuracy des in Koppel und Winter [2014] beschriebenen *AD-C* konnten nicht reproduziert werden. Die Nachimplementierung

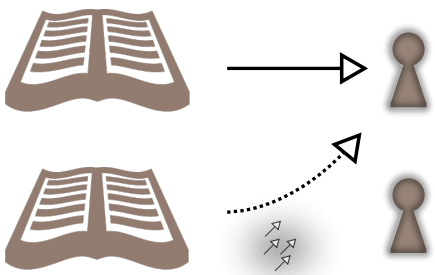


Abbildung 4.1: Schematische Darstellung von Authorship-Boosting als Präprozess. Die Bücher symbolisieren jeweils einen Text des Dokumentenpaars. Die Autorschaft des oberen Dokuments, als Pfeil mit durchgezogener Linie dargestellt, ist bekannt. Die als gestrichelter Pfeil dargestellten Hinweise auf die untere Person werden mittels AB auf die obere Person verschoben.

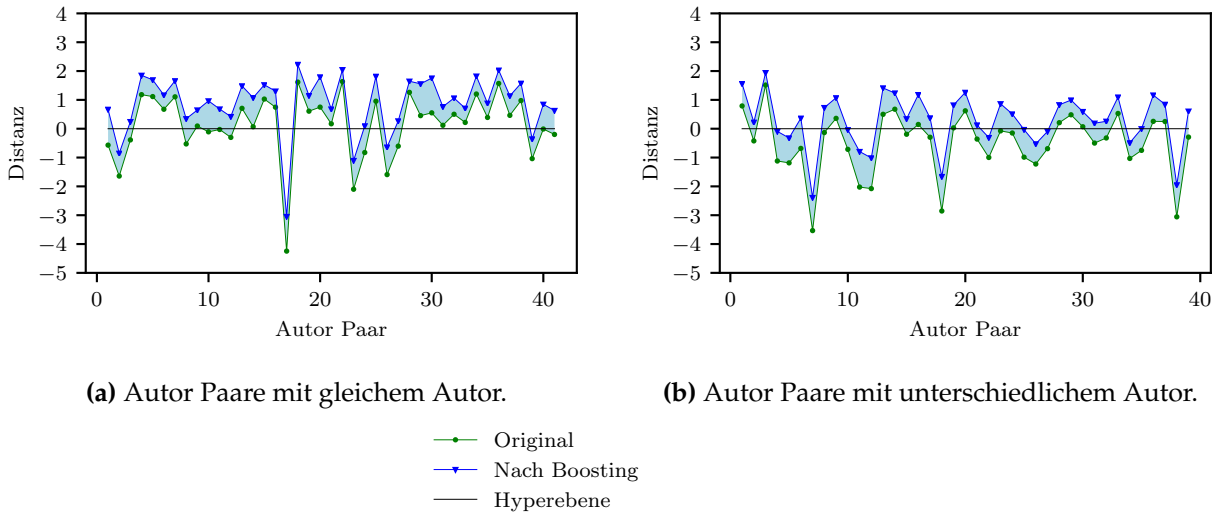
schneidet korpusübergreifend am schlechtesten ab. Die Autoren lassen einige Details ihrer besten Implementierung offen, was es erschwert, ein vergleichbares Ergebnis zu erzielen. Die höchsten Accuracy-Werte der Klassifikatoren SIM-C und Cos liegen zwischen 0.60 und 0.75. Je nach Korpus führt die Hinzunahme von idf zu einer besseren Lösung. Die Werte des AUC-Scores, sowie die Werte des Produkts C@1 · AUC, bestätigen die verhältnismäßig bessere Performanz des ähnlichkeitsbasierten Klassifikators SIM-C auf allen Korpora. Auffällig ist, das die Kombination von Kosinusähnlichkeit und Kullback-Leibler-Divergenz, mit Ausnahme des PAN14-Korpus, eine höhere Performanz zeigt. Die Hinzunahme der inversen Dokumenten-Frequenz führt bei den ähnlichkeitsbasiserten Klassifikatoren SIM-C und Cos, mit Ausnahme des JB16-Korpus zu einer leichten Steigerung. Bei Verwendung der absoluten Differenzen führt die Hinzunahme zu einer Abnahme des Scores.

Als Klassifikator für die nachfolgenden Versuche wird der Klassifikator SIM-C auf tf idf Features als Baseline ausgewählt, da er korpusübergreifend die beste Performanz zeigt.

4.2 Authorship-Boosting als Präprozess

Wenn sich Texte von gleichen Autoren durch AB schneller annähern, als solche von unterschiedlichen, könnte man AB als Präprozess einsetzen, um das Klassifikationsergebnis zu verbessern[4]. Im Folgenden wird dies untersucht. Dabei erhöht sich die Zahl der richtig

[4]Infolgedessen könnte der Gradient der Steigerung der Entscheidungsfunktion zwischen den Boosting-Iterationen zur Klassifikation eingesetzt werden. Da sich keine signifakten Unterschiede zwischen den Klassen zeigen, wird darauf nicht eingegangen.



(a) Autor Paare mit gleichem Autor.

(b) Autor Paare mit unterschiedlichem Autor.

Abbildung 4.2: Sharpening: Distanz der Samples **x** zur Hyperebene H_0 ohne Boosting (Original) und nach 50 Boost-Iterationen auf dem JB16-Korpus. (**a**) Autor Paare mit gleichem Autor (**b**) Autor Paare mit unterschiedlichem Autor.

erkannten gleichen Autoren (True Positives) und fälschlich zugewiesenen unterschiedlichen Autoren (False Positives). Der Vorgang ist angelehnt an eine Bildschärfung und wird im Folgenden als *Sharpening* (engl. Schärfung) bezeichnet. Dazu wurde AB nach Algorithmus 2 implementiert.

Abbildung 4.2 zeigt die Werte der Entscheidungsfunktion nach fünfzig Boost-Iterationen auf dem JB16-Korpus. Die Mehrzahl der Paare gleicher Autoren, die ohne Boosting nicht korrekt zugeordnet werden konnten, befinden sich direkt unter der Hyperebene (siehe Abbildung 4.2a). Nach 100 Iterationen können bis zu elf weitere richtige Paare hinzugefügt werden. Auffällig ist, dass sich Paare, die sich weniger ähnlich sind, schneller annähern (vlg. Tabelle 4.2). Abbildung 4.2b zeigt Paare unterschiedlicher Autoren (vlg. Tabelle 4.3). Obwohl sich die Mehrzahl der Fälle vor dem Boosting unterhalb der Ebene befindet, übersteigt die Anzahl der False Positives nach wenigen Iterationen die der True Positives. Im Mittel unterscheiden sich die Wertsteigerungen der Entscheidungsfunktion der Klassen *gleicher Autor* und *unterschiedliche Autoren* nicht signifikant (p-Value=0.270). Abbildung 4.3 zeigt die Zunahme der als gleich zugewiesenen Fälle in absoluten Zahlen sowie die Accuracy. Die Fläche zwischen beiden Kurven in Abbildung 4.3a und demnach der Anstieg der Accuracy ist nach zehn Boostdurchgängen am größten (siehe Abbildung 4.3b).

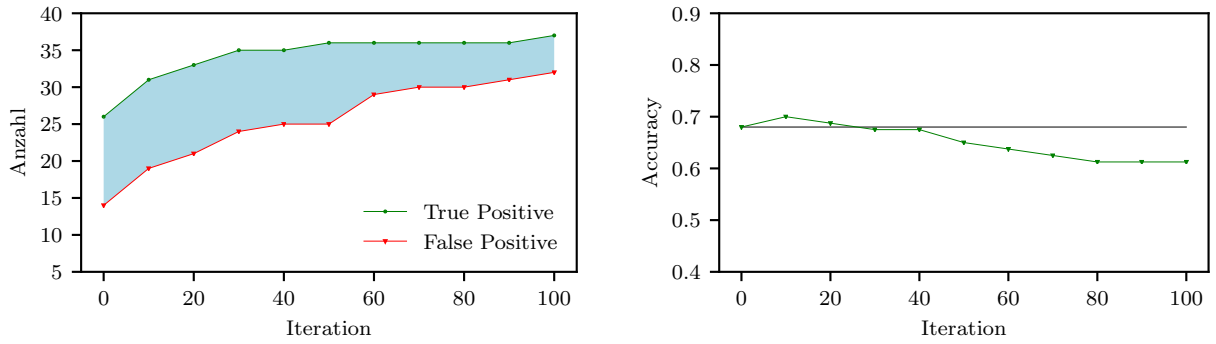
Neben dem JB16-Korpus wurde das Experiment auch für die Korpora PAN14, KW14 und PAN15 durchgeführt. Aufgrund der geringen Textmenge wurde bei KW14 und PAN15 nur ein Feature je Durchgang verändert. Anders als im oben beschriebenen Versuch

Tabelle 4.2: Sharpening: Distanz zur Hyperebene für Textpaare gleicher Autoren auf dem JB16-Korpus. In Abbildung 4.2a ist eine graphische Darstellung nach 40 Iterationen gezeigt. Die Spalten zeigen jeweils die Werte nach i Boost-Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Positives und False Positives.

ID	Original	10	20	30	40	50	60	70	80	90	100
GB004	-0.569	0.018	0.260	0.442	0.566	0.672	0.766	0.846	0.924	0.997	1.062
GB006	-1.642	-1.403	-1.219	-1.086	-0.963	-0.853	-0.750	-0.654	-0.563	-0.479	-0.395
GB007	-0.388	-0.171	-0.043	0.062	0.158	0.245	0.322	0.396	0.471	0.535	0.597
GB008	1.181	1.483	1.593	1.687	1.774	1.852	1.923	1.988	2.052	2.112	2.164
GB009	1.115	1.333	1.455	1.545	1.626	1.695	1.756	1.818	1.879	1.935	1.984
GB010	0.675	0.823	0.928	1.019	1.097	1.168	1.232	1.297	1.360	1.412	1.463
GB011	1.106	1.308	1.416	1.506	1.585	1.662	1.739	1.811	1.874	1.938	2.002
GB012	-0.528	-0.164	0.016	0.143	0.246	0.346	0.433	0.520	0.596	0.670	0.745
GB014	0.094	0.246	0.361	0.468	0.562	0.648	0.722	0.795	0.863	0.924	0.985
GB016	-0.108	0.405	0.594	0.743	0.869	0.965	1.050	1.129	1.199	1.264	1.330
GB019	-0.027	0.225	0.376	0.486	0.588	0.682	0.771	0.852	0.928	0.991	1.054
GB022	-0.301	-0.044	0.118	0.234	0.330	0.421	0.499	0.574	0.639	0.705	0.770
GB023	0.707	1.099	1.235	1.327	1.409	1.486	1.561	1.624	1.688	1.752	1.815
GB024	0.069	0.436	0.629	0.799	0.947	1.075	1.188	1.282	1.372	1.459	1.536
GB025	1.028	1.176	1.276	1.363	1.448	1.522	1.597	1.659	1.721	1.783	1.845
GB027	0.746	0.964	1.070	1.158	1.232	1.304	1.365	1.427	1.489	1.548	1.597
GB031	-4.245	-3.850	-3.602	-3.393	-3.206	-3.045	-2.905	-2.781	-2.664	-2.552	-2.443
GB032	1.617	1.798	1.935	2.047	2.145	2.229	2.305	2.372	2.439	2.506	2.559
GB036	0.608	0.775	0.886	0.977	1.066	1.148	1.224	1.297	1.361	1.424	1.487
GB039	0.749	1.244	1.479	1.609	1.708	1.796	1.869	1.933	1.997	2.061	2.112
GB043	0.171	0.345	0.457	0.537	0.614	0.681	0.744	0.808	0.869	0.920	0.971
GB045	1.635	1.746	1.836	1.914	1.987	2.047	2.107	2.165	2.214	2.262	2.310
GB047	-2.099	-1.730	-1.516	-1.340	-1.214	-1.104	-1.006	-0.916	-0.826	-0.747	-0.668
GB050	-0.825	-0.413	-0.246	-0.115	-0.001	0.098	0.187	0.276	0.356	0.432	0.508
GB051	0.953	1.279	1.529	1.651	1.739	1.816	1.885	1.949	2.013	2.075	2.126
GB053	-1.591	-1.193	-1.006	-0.867	-0.747	-0.638	-0.544	-0.454	-0.363	-0.272	-0.186
GB057	-0.605	-0.278	-0.107	0.033	0.159	0.270	0.360	0.443	0.522	0.594	0.665
GB058	1.262	1.369	1.449	1.521	1.584	1.647	1.705	1.755	1.805	1.856	1.906
GB060	0.453	0.943	1.194	1.351	1.464	1.557	1.643	1.719	1.795	1.859	1.922
GB062	0.550	1.090	1.384	1.545	1.659	1.756	1.839	1.919	1.988	2.056	2.125
GB064	0.118	0.334	0.465	0.575	0.675	0.763	0.849	0.921	0.994	1.067	1.140
GB065	0.501	0.688	0.804	0.900	0.987	1.063	1.138	1.202	1.264	1.326	1.388
GB066	0.217	0.379	0.474	0.557	0.633	0.709	0.784	0.848	0.911	0.974	1.038
GB068	1.203	1.416	1.537	1.643	1.734	1.823	1.902	1.979	2.047	2.111	2.174
GB069	0.391	0.548	0.661	0.745	0.819	0.883	0.947	1.008	1.059	1.110	1.161
GB070	1.570	1.695	1.790	1.875	1.957	2.029	2.097	2.164	2.230	2.284	2.338
GB072	0.463	0.669	0.814	0.934	1.036	1.139	1.230	1.315	1.392	1.468	1.545
GB074	0.976	1.143	1.274	1.383	1.485	1.575	1.655	1.732	1.798	1.863	1.928
GB075	-1.038	-0.808	-0.663	-0.546	-0.448	-0.355	-0.269	-0.191	-0.118	-0.045	0.029
GB081	-0.011	0.262	0.433	0.583	0.722	0.846	0.951	1.040	1.126	1.200	1.272
GB082	-0.199	0.144	0.312	0.439	0.542	0.631	0.717	0.791	0.865	0.937	0.998
True Positive	26	31	33	35	35	36	36	36	36	36	37
False Positive	14	19	21	24	25	25	29	30	30	31	32

Tabelle 4.3: Sharpening: Distanz zur Hyperebene für Textpaare unterschiedlichen Autoren auf dem JB16-Korpus. In Abbildung 4.2b ist eine graphische Darstellung nach 40 Iterationen gezeigt. Die Spalten zeigen jeweils die Werte nach i Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Negatives.

ID	Original	10	20	30	40	50	60	70	80	90	100
GB001	0.790	1.035	1.202	1.339	1.456	1.562	1.655	1.745	1.824	1.894	1.960
GB002	-0.423	-0.202	-0.070	0.043	0.139	0.223	0.303	0.375	0.447	0.518	0.590
GB003	1.515	1.655	1.740	1.814	1.882	1.944	2.005	2.065	2.114	2.163	2.212
GB005	-1.120	-0.713	-0.493	-0.336	-0.206	-0.093	0.010	0.101	0.187	0.264	0.340
GB013	-1.185	-0.825	-0.662	-0.535	-0.423	-0.321	-0.224	-0.131	-0.047	0.029	0.101
GB015	-0.683	-0.150	0.040	0.171	0.275	0.364	0.440	0.514	0.588	0.650	0.711
GB017	-3.534	-3.186	-2.950	-2.749	-2.548	-2.388	-2.238	-2.088	-1.938	-1.787	-1.678
GB018	-0.133	0.147	0.335	0.489	0.621	0.732	0.832	0.926	1.008	1.088	1.155
GB020	0.359	0.654	0.789	0.907	0.995	1.072	1.140	1.209	1.277	1.345	1.404
GB021	-0.715	-0.496	-0.353	-0.237	-0.134	-0.036	0.055	0.141	0.215	0.289	0.350
GB026	-2.024	-1.544	-1.307	-1.113	-0.940	-0.792	-0.668	-0.554	-0.452	-0.350	-0.253
GB028	-2.077	-1.523	-1.354	-1.226	-1.115	-1.013	-0.916	-0.827	-0.737	-0.657	-0.581
GB029	0.499	0.811	0.980	1.141	1.287	1.414	1.524	1.619	1.700	1.781	1.859
GB030	0.677	0.822	0.943	1.055	1.149	1.237	1.314	1.391	1.469	1.537	1.601
GB033	-0.192	0.005	0.108	0.196	0.269	0.343	0.417	0.482	0.543	0.605	0.662
GB034	0.147	0.582	0.780	0.928	1.059	1.181	1.292	1.394	1.486	1.565	1.644
GB035	-0.296	-0.072	0.064	0.180	0.278	0.374	0.460	0.546	0.625	0.699	0.772
GB037	-2.853	-2.471	-2.211	-1.997	-1.809	-1.660	-1.535	-1.424	-1.316	-1.218	-1.119
GB038	0.029	0.360	0.497	0.617	0.723	0.820	0.910	0.999	1.077	1.154	1.230
GB040	0.621	0.897	1.009	1.102	1.184	1.260	1.328	1.392	1.455	1.518	1.581
GB041	-0.360	-0.220	-0.113	-0.026	0.055	0.128	0.200	0.263	0.323	0.383	0.444
GB042	-0.997	-0.760	-0.627	-0.506	-0.403	-0.309	-0.225	-0.148	-0.071	-0.003	0.061
GB044	-0.075	0.301	0.496	0.647	0.766	0.865	0.957	1.044	1.125	1.199	1.270
GB046	-0.149	0.098	0.227	0.331	0.425	0.511	0.596	0.675	0.748	0.821	0.885
GB049	-0.987	-0.589	-0.418	-0.273	-0.144	-0.030	0.071	0.159	0.246	0.331	0.405
GB052	-1.224	-1.017	-0.864	-0.739	-0.629	-0.526	-0.437	-0.348	-0.260	-0.178	-0.108
GB054	-0.692	-0.492	-0.372	-0.266	-0.173	-0.092	-0.019	0.048	0.108	0.168	0.229
GB055	0.209	0.434	0.557	0.658	0.749	0.830	0.908	0.972	1.037	1.102	1.156
GB056	0.483	0.640	0.746	0.837	0.915	0.994	1.059	1.124	1.189	1.254	1.312
GB059	0.068	0.221	0.325	0.422	0.510	0.594	0.670	0.744	0.806	0.869	0.931
GB063	-0.498	-0.256	-0.112	0.003	0.100	0.190	0.273	0.350	0.427	0.497	0.561
GB067	-0.321	-0.153	-0.030	0.073	0.168	0.258	0.336	0.413	0.487	0.551	0.614
GB071	0.529	0.772	0.876	0.963	1.033	1.096	1.160	1.224	1.281	1.332	1.382
GB073	-1.031	-0.875	-0.756	-0.662	-0.575	-0.489	-0.415	-0.341	-0.272	-0.210	-0.148
GB076	-0.751	-0.444	-0.297	-0.186	-0.089	-0.002	0.083	0.167	0.241	0.313	0.386
GB077	0.255	0.692	0.833	0.958	1.069	1.166	1.256	1.343	1.420	1.497	1.574
GB078	0.250	0.457	0.577	0.672	0.765	0.845	0.924	0.996	1.062	1.128	1.192
GB079	-3.056	-2.707	-2.450	-2.254	-2.091	-1.947	-1.818	-1.704	-1.590	-1.476	-1.373
GB080	-0.291	0.064	0.239	0.386	0.504	0.607	0.698	0.786	0.863	0.941	1.011
True Negative	25	20	18	15	14	14	10	9	9	8	7



(a) Anzahl der True und False Positives.

(b) Accuracy des Klassifikators.

Abbildung 4.3: Sharpening: Die Graphiken zeigen die Evaluation nach iterativem Boosting auf dem JB16-Korpus (**a**) als Anzahl der True und False Positives (**b**) und als Accuracy des Klassifikators. Die schwarze Linie markiert die Accuracy vor dem AB.

verschiebt sich das Verhältnis der korrekt zugewiesenen Fälle im Vergleich zu den falsch klassifizierten Fällen nur zum Schlechteren oder bleibt gleich. Aufgrund dessen wird nur ein Überblick über die Ergebnisse gegeben. Bei PAN14 sinkt die Accuracy nach fünfzig Iterationen von 0.535 auf 0.525. Im Fall KW14 führt Boosting zu einer Verschlechterung von 0.730 auf 0.562, bei PAN15 von 0.570 auf 0.560, wobei die Zahl der ausgetauschten Samples mit 19 True Positives und 28 False Postives besonders hoch ist.

In den Experimenten konnte nur auf dem JB16-Korpus nach einer geringen Anzahl von Boost-Iterationen eine Verbesserung der Klassifikation auf dem Test-Korpus erreicht werden. Für eine hohe Anzahl von Boosting-Iterationen wird eine Verschlechterung beobachtet. Da sich die Klassen *gleicher Autor* und *unterschiedliche Autoren* andernfalls sehr ähnlich verhalten und die Verbesserungen auf Zufallseffekte zurückgeführt werden können, wird von der Verwendung von Authorship-Boosting als Präprozess abgeraten.

4.3 Authorship-Boosting gegen Obfuskierung

Die in Tabelle 4.4 gezeigte Evaluation des Klassifikators SIM-C auf den obfuskierten Korpora verdeutlicht, dass sich alle der betrachteten Korpora durch AO so stark beeinflussen lassen, dass der Klassifikator in vielen Fällen getäuscht werden kann. Die Ergebnisse des SIM-C Klassifikators auf dem obfuskierten JB16 Korpus sinken auf eine Accuracy von 0.44 und einen AUC von 0.47, was in einem Score von 0.21 resultiert. Wird mehr obfuskiert ($\epsilon_{0.7}$), fällt die Accuracy weiter auf 0.36, der AUC auf 0.38 was einen Score von 0.14 ergibt. Die Performanz auf PAN14 und PAN15 nimmt ähnlich stark ab.

Tabelle 4.4: Baseline Evaluation auf obfuskierten Korpora (Obfuskierer: Bevendorff [2018]). Beispielhaft mit Klassifikator SIM-C.

Korpus	C@1	AUC	C@1 · AUC	Schwellwert
PAN14	0.50	0.59	0.30	$\epsilon_{0.5}$
PAN15	0.58	0.71	0.41	$\epsilon_{0.5}$
JB16	0.44	0.47	0.21	$\epsilon_{0.5}$
JB16	0.36	0.38	0.14	$\epsilon_{0.7}$

Zum Zeitpunkt des Verfassens dieser Arbeit ist kein Verfahren bekannt, dass Texte mit computergestützten Verfahren derart obfuskieren kann so dass alle in Abschnitt 3.5 definierten Qualitätsmetriken erfüllt werden [Potthast et al., 2018b]. Forensiker können bei stark obfuskierten Dokumenten erkennen, dass obfuskiert wurde und das Dokument daraufhin gegen mögliche bekannte Texte der verdächtige Personen vergleichen. Der folgende Ansatz geht dieser Intuition nach, indem mit bekannten Texten des Autors versucht wird, die Repräsentation vor der Obfuskierung wieder herzustellen. Gelingt dies vergleichsweise schnell, kann damit die Hypothese der Gleichheit der Autoren der Texte gestärkt werden.

Im folgenden Versuch wird die Repräsentation des unbekannten Textes mit der des bekannten Textes bei m Features obfuskiert und anschließend mit dem gleichen Text die gleiche Anzahl Features geboostet. Wenn nicht anders definiert, werden $m = 100$ Features obfuskiert. Dazu wurde AO nach Algorithmus 1 implementiert. Zusätzlich wurden zwei verschieden stark obfuskierte Versionen des JB16-Korpus evaluiert, die mit dem darauf aufbauenden Obfuskierer aus Bevendorff [2018] obfuskiert wurden. Texte unterschiedlicher Autoren müssen nicht berücksichtigt werden, da keine Notwendigkeit der Obfuskierung besteht. Das Verfahren kann durch

$$\text{boosting}(\langle \widetilde{d_u}, D_a \rangle) = (\langle \widehat{d_u}, D_a \rangle) \tag{4.2}$$

beschrieben werden, wobei $\widehat{d_u}$ eine Approximation der ursprünglichen Textrepräsenation d_u ist. Ziel des AB ist demnach, die obfuskierte Repräsentation $\widetilde{d_u}$ der originalen Repräsentation d_u möglichst anzunähern. Wie im Sharpening Experiment wird dazu die Distanz, bzw. der Wert der Entscheidungsfunktion der SVM vor der Obfuskierung, nach der Obfuskierung und beispielhaft nach 40 Boost-Iterationen gezeigt (siehe Abbildung 4.4). Diesbezüglich kann eine Annäherung angenommen werden, wenn sich die Punkte eines

Tabelle 4.5: Boosting mit Obfuskierungs-Text (1): Distanz zur Hyperebene für Textpaare gleicher Autoren auf dem JB16-Korpus. In Abbildung 4.4a ist eine graphische Darstellung nach 40 Boost-Iterationen gezeigt. 100 Obfuskierungs-Iterationen nach Algorithmus 1. Die Spalten zeigen jeweils die Werte nach nach i Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Positives.

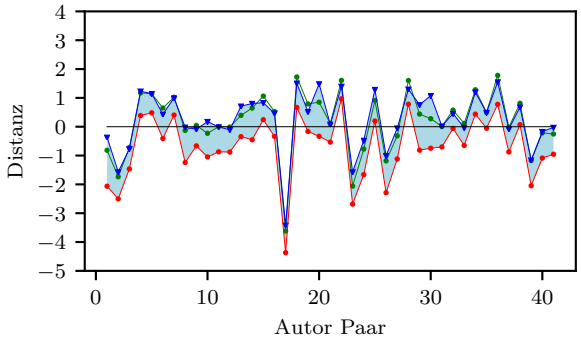
ID	Original	Obfuskiert	10	20	30	40	50	60	70	80	90	100
GB004	-1.172	-2.108	-0.951	-0.647	-0.349	-0.122	0.092	0.220	0.350	0.450	0.538	0.653
GB006	-1.812	-2.443	-2.232	-2.022	-1.728	-1.623	-1.482	-1.373	-1.255	-1.167	-1.110	-1.038
GB007	-0.814	-1.424	-1.090	-0.931	-0.824	-0.729	-0.638	-0.568	-0.467	-0.390	-0.323	-0.243
GB008	1.032	0.379	0.897	1.044	1.137	1.241	1.319	1.412	1.490	1.609	1.660	1.733
GB009	1.288	0.684	0.886	1.100	1.200	1.323	1.387	1.476	1.536	1.597	1.648	1.700
GB010	0.569	-0.259	0.151	0.313	0.525	0.685	0.797	0.906	1.031	1.090	1.186	1.254
GB011	0.514	-0.009	0.220	0.447	0.558	0.652	0.734	0.801	0.861	0.962	1.017	1.068
GB012	-0.090	-1.089	-0.687	-0.395	-0.177	0.061	0.225	0.367	0.492	0.587	0.658	0.710
GB014	0.190	-0.381	-0.189	-0.070	0.030	0.166	0.268	0.367	0.460	0.520	0.589	0.678
GB016	-0.038	-0.713	-0.190	0.100	0.292	0.447	0.545	0.668	0.778	0.850	0.917	1.008
GB019	-0.016	-0.750	-0.237	-0.110	0.058	0.203	0.331	0.422	0.510	0.596	0.712	0.793
GB022	0.245	-0.515	-0.250	-0.041	0.145	0.286	0.407	0.485	0.563	0.658	0.748	0.810
GB023	0.021	-0.625	-0.020	0.199	0.330	0.426	0.501	0.587	0.654	0.743	0.797	0.854
GB024	0.573	-0.331	0.188	0.415	0.575	0.712	0.876	1.048	1.181	1.319	1.407	1.521
GB025	0.736	0.037	0.272	0.389	0.518	0.616	0.669	0.742	0.808	0.868	0.949	1.053
GB027	0.230	-0.466	-0.190	-0.022	0.115	0.264	0.363	0.460	0.529	0.628	0.678	0.744
GB031	-3.555	-4.161	-3.839	-3.641	-3.381	-3.187	-3.025	-2.870	-2.773	-2.646	-2.497	-2.417
GB032	1.614	0.782	1.152	1.326	1.510	1.626	1.738	1.836	1.899	2.009	2.077	2.166
GB036	0.565	-0.277	0.015	0.159	0.310	0.450	0.570	0.684	0.774	0.865	0.952	1.029
GB039	0.898	-0.100	0.514	0.928	1.367	1.613	1.773	1.906	1.986	2.086	2.167	2.267
GB043	0.242	-0.333	0.002	0.121	0.284	0.377	0.505	0.577	0.649	0.694	0.742	0.814
GB045	1.590	1.019	1.096	1.186	1.260	1.402	1.461	1.546	1.589	1.646	1.709	1.765
GB047	-2.124	-2.615	-2.270	-1.882	-1.657	-1.444	-1.214	-1.081	-0.995	-0.916	-0.784	-0.720
GB050	-1.155	-1.872	-1.340	-1.031	-0.835	-0.673	-0.538	-0.401	-0.310	-0.224	-0.106	-0.012
GB051	0.473	-0.184	0.134	0.503	0.810	1.005	1.157	1.252	1.327	1.401	1.494	1.544
GB053	-1.321	-2.254	-1.772	-1.432	-1.057	-0.847	-0.690	-0.561	-0.423	-0.353	-0.222	-0.151
GB057	-0.501	-1.159	-0.628	-0.400	-0.203	-0.084	0.051	0.145	0.231	0.357	0.429	0.500
GB058	1.664	0.774	1.152	1.250	1.335	1.405	1.466	1.556	1.610	1.675	1.742	1.801
GB060	0.247	-0.775	-0.294	0.134	0.482	0.859	1.076	1.285	1.440	1.575	1.708	1.800
GB062	0.231	-0.547	0.111	0.534	0.815	1.028	1.210	1.370	1.474	1.597	1.673	1.746
GB064	0.266	-0.378	-0.171	-0.001	0.137	0.252	0.324	0.421	0.539	0.624	0.703	0.764
GB065	0.408	-0.116	0.083	0.184	0.270	0.375	0.436	0.525	0.575	0.638	0.718	0.768
GB066	0.293	-0.345	-0.109	0.033	0.157	0.265	0.344	0.403	0.483	0.548	0.595	0.642
GB068	1.359	0.645	0.943	1.065	1.164	1.319	1.415	1.511	1.599	1.669	1.733	1.814
GB069	0.734	0.243	0.382	0.522	0.637	0.729	0.804	0.900	0.973	1.017	1.086	1.176
GB070	1.712	0.851	1.086	1.280	1.477	1.598	1.704	1.772	1.857	1.954	2.012	2.089
GB072	-0.294	-1.012	-0.792	-0.603	-0.462	-0.321	-0.143	-0.020	0.152	0.234	0.316	0.426
GB074	0.832	0.209	0.428	0.574	0.675	0.824	0.922	1.000	1.102	1.169	1.258	1.385
GB075	-0.982	-1.752	-1.475	-1.308	-1.093	-0.996	-0.863	-0.750	-0.641	-0.574	-0.510	-0.418
GB081	-0.028	-0.830	-0.532	-0.362	-0.212	-0.002	0.137	0.320	0.480	0.585	0.684	0.834
GB082	-0.637	-1.302	-0.847	-0.610	-0.395	-0.253	-0.133	-0.052	0.014	0.106	0.180	0.257
True Positive	26	10	19	23	28	29	32	32	34	34	34	34

Tabelle 4.6: Boosting mit Obfuskierungs-Text (2): Distanz zur Hyperebene für Textpaare gleicher Autoren auf dem JB16-Korpus. In Abbildung 4.4a ist eine graphische Darstellung nach 40 Boost-Iterationen gezeigt. Obfuskiererung nach Bevendorff [2018], bis das 50-Perzentil ($\epsilon_{0.5}$) der Jensen-Shannon Distanzen respektive Textlänge der Klasse *unterschiedliche Autoren* überschritten ist. Die Spalten zeigen jeweils die Werte nach nach i Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Positives.

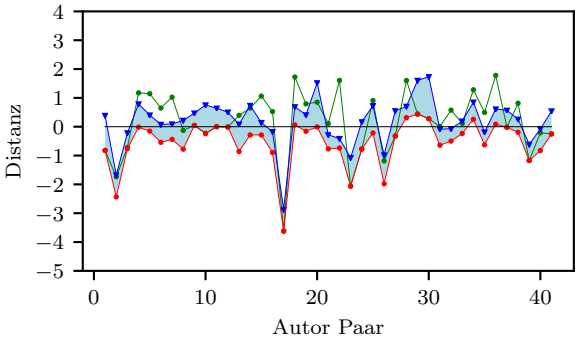
ID	Original	Obfuskiert	10	20	30	40	50	60	70	80	90	100
GB004	-0.818	-0.825	-0.114	0.152	0.390	0.527	0.660	0.793	0.874	0.967	1.053	1.108
GB006	-1.737	-2.433	-2.195	-1.883	-1.675	-1.515	-1.391	-1.233	-1.113	-1.045	-0.953	-0.881
GB007	-0.721	-0.761	-0.478	-0.332	-0.214	-0.079	0.004	0.097	0.161	0.247	0.340	0.393
GB008	1.174	-0.016	0.453	0.642	0.788	0.906	0.999	1.084	1.222	1.284	1.365	1.439
GB009	1.146	-0.154	0.081	0.285	0.404	0.513	0.574	0.674	0.741	0.850	0.899	0.956
GB010	0.648	-0.542	-0.290	-0.065	0.079	0.205	0.320	0.439	0.533	0.609	0.711	0.798
GB011	1.024	-0.438	-0.206	-0.020	0.095	0.207	0.298	0.385	0.460	0.546	0.603	0.678
GB012	-0.129	-0.778	-0.337	0.012	0.220	0.381	0.497	0.593	0.720	0.783	0.873	0.930
GB014	0.043	0.044	0.216	0.334	0.472	0.572	0.669	0.744	0.821	0.901	0.948	1.003
GB016	-0.229	-0.240	0.378	0.609	0.754	0.897	0.993	1.103	1.177	1.260	1.320	1.376
GB019	0.003	0.001	0.323	0.532	0.642	0.755	0.862	0.972	1.088	1.172	1.224	1.280
GB022	-0.013	-0.013	0.205	0.395	0.505	0.601	0.699	0.759	0.842	0.905	0.977	1.032
GB023	0.394	-0.860	-0.251	-0.040	0.088	0.174	0.269	0.331	0.424	0.522	0.591	0.672
GB024	0.646	-0.284	0.350	0.528	0.736	0.935	1.115	1.237	1.377	1.459	1.534	1.612
GB025	1.061	-0.286	-0.084	0.033	0.142	0.227	0.311	0.378	0.473	0.588	0.652	0.722
GB027	0.526	-0.895	-0.583	-0.353	-0.172	-0.046	0.045	0.146	0.229	0.333	0.406	0.506
GB031	-3.618	-3.621	-3.287	-3.079	-2.872	-2.710	-2.543	-2.417	-2.327	-2.214	-2.107	-1.990
GB032	1.725	0.063	0.305	0.513	0.692	0.844	0.998	1.105	1.227	1.347	1.451	1.551
GB036	0.792	-0.158	0.132	0.269	0.411	0.530	0.620	0.715	0.838	0.943	1.023	1.108
GB039	0.850	-0.008	0.815	1.301	1.522	1.647	1.750	1.844	1.960	2.035	2.106	2.169
GB043	0.110	-0.769	-0.557	-0.394	-0.281	-0.169	-0.093	-0.025	0.030	0.100	0.163	0.210
GB045	1.607	-0.740	-0.605	-0.504	-0.412	-0.300	-0.216	-0.118	-0.011	0.128	0.195	0.280
GB047	-2.060	-2.066	-1.556	-1.314	-1.074	-0.953	-0.809	-0.698	-0.616	-0.512	-0.450	-0.373
GB050	-0.775	-0.774	-0.194	0.034	0.171	0.311	0.388	0.465	0.562	0.630	0.703	0.783
GB051	0.909	-0.216	0.189	0.505	0.729	0.850	0.971	1.069	1.156	1.256	1.333	1.421
GB053	-1.185	-1.980	-1.516	-1.184	-0.980	-0.825	-0.710	-0.595	-0.466	-0.402	-0.328	-0.256
GB057	-0.317	-0.327	0.190	0.398	0.551	0.670	0.786	0.862	0.931	1.038	1.098	1.166
GB058	1.602	0.312	0.525	0.625	0.705	0.783	0.852	0.933	0.998	1.072	1.136	1.189
GB060	0.438	0.439	0.990	1.402	1.607	1.742	1.857	1.972	2.055	2.154	2.206	2.271
GB062	0.279	0.279	1.085	1.477	1.734	1.872	1.975	2.058	2.161	2.227	2.300	2.389
GB064	0.008	-0.642	-0.391	-0.207	-0.082	0.037	0.161	0.256	0.333	0.398	0.493	0.551
GB065	0.575	-0.497	-0.284	-0.187	-0.070	0.006	0.084	0.147	0.225	0.302	0.362	0.432
GB066	0.113	-0.236	-0.032	0.117	0.194	0.280	0.343	0.420	0.498	0.585	0.635	0.698
GB068	1.281	0.257	0.557	0.702	0.851	1.004	1.113	1.242	1.362	1.438	1.529	1.661
GB069	0.496	-0.629	-0.509	-0.354	-0.187	-0.074	0.031	0.135	0.194	0.278	0.351	0.405
GB070	1.780	0.086	0.346	0.495	0.613	0.740	0.831	0.927	1.064	1.135	1.203	1.308
GB072	-0.019	-0.020	0.236	0.419	0.569	0.666	0.850	0.935	1.076	1.153	1.227	1.333
GB074	0.813	-0.192	-0.032	0.125	0.260	0.413	0.519	0.645	0.735	0.865	0.929	0.994
GB075	-1.173	-1.173	-0.876	-0.728	-0.613	-0.524	-0.423	-0.335	-0.252	-0.197	-0.113	-0.015
GB081	-0.212	-0.825	-0.549	-0.241	-0.079	0.111	0.312	0.402	0.563	0.648	0.724	0.808
GB082	-0.252	-0.249	0.184	0.426	0.546	0.644	0.736	0.807	0.870	0.963	1.064	1.116
True Positive	26	8	19	25	28	31	34	34	35	36	36	36

Tabelle 4.7: Boosting mit Obfuskierungs-Text (3): Distanz zur Hyperebene für Textpaare gleicher Autoren auf dem JB16-Korpus. In Abbildung 4.4a ist eine graphische Darstellung nach 40 Boost-Iterationen gezeigt. Obfuskiererung nach Bevendorff [2018], bis das 70-Perzentil ($\epsilon_{0.7}$) der Jensen-Shannon Distanzen respektive Textlänge der Klasse *unterschiedliche Autoren* überschritten ist. Die Spalten zeigen jeweils die Werte nach nach i Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Positives.

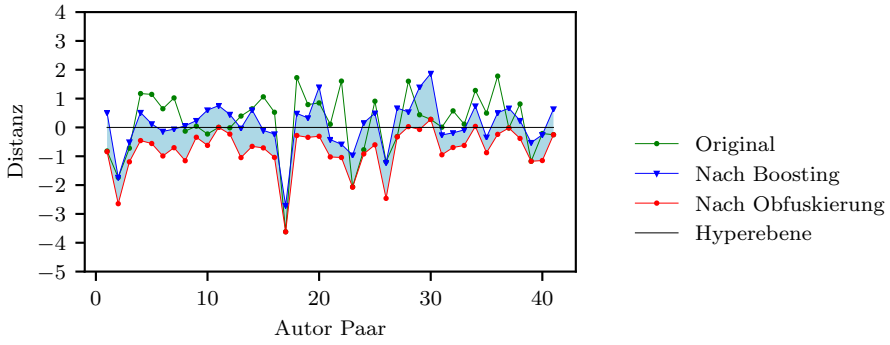
ID	Original	Obfuskiert	10	20	30	40	50	60	70	80	90	100
GB004	-1.172	-2.027	-0.847	-0.527	-0.289	-0.031	0.100	0.206	0.294	0.399	0.455	0.539
GB006	-1.812	-3.494	-3.238	-3.056	-2.726	-2.632	-2.487	-2.355	-2.258	-2.166	-2.108	-2.030
GB007	-0.814	-1.979	-1.598	-1.443	-1.321	-1.224	-1.140	-1.022	-0.966	-0.863	-0.793	-0.730
GB008	1.032	-1.061	-0.499	-0.278	-0.116	0.012	0.112	0.195	0.299	0.381	0.488	0.562
GB009	1.288	-0.992	-0.777	-0.558	-0.426	-0.312	-0.205	-0.116	-0.047	0.054	0.110	0.188
GB010	0.569	-1.564	-1.125	-0.860	-0.642	-0.507	-0.363	-0.241	-0.104	-0.017	0.072	0.129
GB011	0.514	-1.795	-1.567	-1.372	-1.174	-1.057	-0.940	-0.841	-0.734	-0.670	-0.600	-0.503
GB012	-0.090	-1.752	-1.302	-1.046	-0.762	-0.522	-0.392	-0.248	-0.156	-0.101	0.004	0.147
GB014	0.190	-0.864	-0.682	-0.549	-0.434	-0.292	-0.211	-0.126	-0.030	0.028	0.106	0.186
GB016	-0.038	-1.127	-0.533	-0.241	-0.046	0.101	0.230	0.332	0.448	0.518	0.604	0.674
GB019	-0.016	-0.768	-0.340	-0.113	0.026	0.121	0.239	0.348	0.443	0.573	0.636	0.702
GB022	0.245	-0.720	-0.452	-0.236	-0.052	0.058	0.183	0.259	0.358	0.419	0.493	0.536
GB023	0.021	-2.005	-1.374	-1.148	-0.999	-0.893	-0.805	-0.730	-0.636	-0.566	-0.485	-0.408
GB024	0.573	-1.767	-1.004	-0.696	-0.535	-0.389	-0.196	-0.010	0.134	0.264	0.392	0.505
GB025	0.736	-1.528	-1.270	-1.145	-1.014	-0.922	-0.852	-0.767	-0.708	-0.648	-0.581	-0.489
GB027	0.230	-1.998	-1.818	-1.572	-1.395	-1.255	-1.108	-0.992	-0.911	-0.828	-0.762	-0.682
GB031	-3.555	-4.570	-4.169	-3.902	-3.713	-3.525	-3.383	-3.221	-3.114	-3.029	-2.920	-2.841
GB032	1.614	-0.872	-0.379	-0.166	0.048	0.168	0.292	0.409	0.508	0.583	0.696	0.783
GB036	0.565	-1.490	-1.251	-1.042	-0.876	-0.700	-0.562	-0.439	-0.318	-0.211	-0.113	-0.030
GB039	0.898	-1.048	-0.367	0.146	0.544	0.720	0.889	1.017	1.134	1.224	1.329	1.407
GB043	0.242	-1.574	-1.216	-1.125	-1.016	-0.810	-0.690	-0.600	-0.531	-0.449	-0.401	-0.347
GB045	1.590	-1.360	-1.208	-1.126	-1.027	-0.933	-0.827	-0.735	-0.657	-0.513	-0.432	-0.356
GB047	-2.124	-3.020	-2.529	-2.129	-1.892	-1.657	-1.494	-1.357	-1.273	-1.181	-1.122	-1.011
GB050	-1.155	-2.085	-1.448	-1.175	-0.960	-0.830	-0.718	-0.606	-0.523	-0.433	-0.367	-0.274
GB051	0.473	-1.799	-1.515	-1.139	-0.781	-0.514	-0.323	-0.224	-0.129	-0.035	0.061	0.133
GB053	-1.321	-3.226	-2.701	-2.337	-1.974	-1.771	-1.613	-1.473	-1.342	-1.225	-1.111	-1.021
GB057	-0.501	-1.293	-0.712	-0.462	-0.276	-0.146	-0.077	0.057	0.135	0.220	0.278	0.372
GB058	1.664	-0.490	-0.088	0.016	0.121	0.186	0.256	0.320	0.404	0.468	0.531	0.576
GB060	0.247	-0.932	-0.405	0.135	0.531	0.782	0.976	1.122	1.224	1.337	1.446	1.548
GB062	0.231	-0.489	0.248	0.717	0.938	1.122	1.256	1.363	1.450	1.554	1.633	1.697
GB064	0.266	-1.255	-1.020	-0.871	-0.698	-0.599	-0.517	-0.364	-0.295	-0.210	-0.138	-0.074
GB065	0.408	-1.350	-1.132	-1.023	-0.926	-0.831	-0.765	-0.658	-0.609	-0.552	-0.478	-0.397
GB066	0.293	-1.160	-0.891	-0.734	-0.591	-0.504	-0.445	-0.351	-0.309	-0.241	-0.170	-0.071
GB068	1.359	-0.419	-0.115	0.016	0.101	0.271	0.371	0.477	0.572	0.660	0.750	0.823
GB069	0.734	-1.097	-0.962	-0.838	-0.697	-0.592	-0.510	-0.408	-0.345	-0.293	-0.238	-0.152
GB070	1.712	-0.778	-0.527	-0.327	-0.111	0.024	0.144	0.232	0.353	0.426	0.509	0.621
GB072	-0.294	-1.006	-0.734	-0.545	-0.389	-0.211	-0.113	0.008	0.140	0.218	0.346	0.440
GB074	0.832	-0.872	-0.634	-0.473	-0.363	-0.257	-0.115	-0.005	0.086	0.191	0.262	0.384
GB075	-0.982	-1.811	-1.568	-1.412	-1.256	-1.165	-1.083	-0.984	-0.892	-0.814	-0.757	-0.681
GB081	-0.028	-1.647	-1.287	-1.153	-0.937	-0.773	-0.611	-0.452	-0.308	-0.212	-0.051	0.021
GB082	-0.637	-1.385	-0.904	-0.616	-0.433	-0.333	-0.232	-0.161	-0.092	-0.026	0.051	0.151
True Positive	26	0	1	5	7	11	12	14	16	18	22	23



(a) Obfuskierung nach Algorithmus 1.



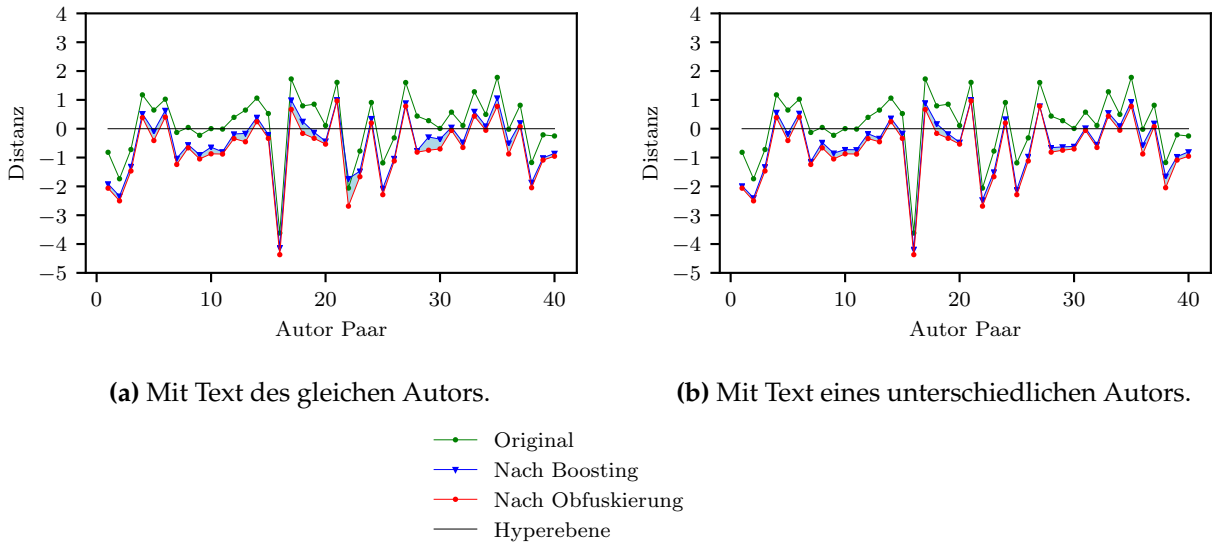
(b) Obfuskierer von Bevendorff [2018]: $\epsilon_{0.5}$.



(c) Obfuskierer von Bevendorff [2018]: $\epsilon_{0.7}$.

Abbildung 4.4: Distanzen der Samples **x** der Klasse *gleicher Autor* zur Hyperebene H_0 ohne Boosting (Original), nach Obfuskierung und nach 40 Boost-Iterationen auf dem JB16-Korpus. **(a)** 100 Obfuskierungs-Iterationen nach Algorithmus 1. Eine vollständige Auflistung ist in Tabelle 4.7 gezeigt. **(b)** Mit Obfuskierer von Bevendorff [2018], bis das 50-Perzentil ($\epsilon_{0.5}$) der Jensen-Shannon Distanzen respektive Textlänge der Klasse *unterschiedliche Autoren* überschritten ist. **(b)** Mit Obfuskierer von Bevendorff [2018], bis das 70-Perzentil ($\epsilon_{0.7}$) der Jensen-Shannon Distanzen respektive Textlänge der Klasse *unterschiedliche Autoren* überschritten ist.

Samples vor der Obfuskierung (Original) und nach dem Boosting überschneiden. Zu diesem Zeitpunkt weist der Klassifikator die Samples mit der gleichen Sicherheit wie vor der Obfuskierung zu (siehe Tabelle 4.7). Mit fortlaufenden Iterationen erhöhen sich die Werte. Die Mehrzahl der Samples befindet sich nach der Obfuskierung leicht unterhalb der Hyperebene, was eine schnelle Annäherung durch das AB erleichtert. Auffällig ist, dass AB im gezeigten Versuch keine Tendenz dazu hat, komplementär zur Obfuskierung zu sein, wie zunächst angenommen, sondern eine stärkere Boosting-Wirkung erzielt. Um zu überprüfen, ob sich AB auch gegen komplexere Obfuskierer einsetzen lässt, wurde der Versuch auf dem mit heuristischer Suche obfuskierten JB16-Korpus durchgeführt. Auch hier ist ein deutlicher Anstieg der Distanzen, bzw. Wechsel auf die andere Seite der Hyperebene zu erkennen (siehe Abbildungen 4.4b und 4.4c). Aufgrund der abweichenden Methode,



(a) Mit Text des gleichen Autors.

(b) Mit Text eines unterschiedlichen Autors.

Abbildung 4.5: Distanzen der Samples **x** der Klasse *gleicher Autor* zur Hyperebene H_0 ohne Boosting (Original), nach Obfuskierung und nach 40 Boost-Iterationen auf dem JB16-Korpus. Nach 100 Obfuskierungs-Iterationen nach Algorithmus 1 wird mit (**a**) mit einem Text des gleichen Autors geboostet, vollständige Auflistung in Tabelle 4.8. (**b**) mit einem Text eines unterschiedlichen Autors geboostet, vollständige Auflistung in Tabelle 4.9.

wird die ursprüngliche Repräsentation weniger genau angenähert. Paare, die nah an der Hyperebene liegen, können durch AB leicht nach oben verschoben werden. Bei vielen Paaren ist eine Annäherung an den ursprünglichen Wert zu erkennen. Der Versuch hat gezeigt, dass durch Authorship-Boosting die ursprüngliche Repräsentation der Texte, gemessen an der Distanz zur Hyperebene, nach wenigen Durchgängen angenähert werden kann, wenn Algorithmus 1 als Obfuskierer und Algorithmus 2 als AB verwendet wird. Das Verfahren lässt sich auch gegen komplexere Obfuskierungs-Methoden wie den Obfuskierer von Bevendorff [2018] einsetzen. Die Anzahl der AB-Iterationen, die benötigt wird, um die ursprüngliche Distanz zur Hyperebene wiederherzustellen, ist geringer als die verwendete Anzahl der Obfuskierungs-Iterationen. Das kann damit begründet werden, dass neben den entfernten Features, die während der AO entfernt wurden, weitere eingefügt worden, wie bereits in Abschnitt 4.2 beschrieben.

Weiter wird überprüft, ob AB als Rückführung zur ursprünglich Repräsentation eingesetzt werden kann, wenn zum AB statt des gleichen Textes ein anderer Text des Autors verwendet wird. Der Aufbau des Experiments entspricht dem des vorherigen, mit der Ausnahme, dass mit einem anderen Text des Autors geboostet wird. Dieser dritte Text fehlt für den JB16-Korpus. Aufgrund dessen wurde der Test-Korpus in Fällen des gleichen

Tabelle 4.8: Boosting mit einem weiterem Text des gleichen Autors auf JB16-Korpus. In Abbildung 4.5a ist eine graphische Darstellung nach 40 Boost-Iterationen gezeigt. 100 Obfuskierungs-Iterationen nach Algorithmus 1. Die Spalten zeigen jeweils die Werte nach i Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Positives.

ID	Original	Obfuskiert	20	40	60	80	100	120	140	160	180	200
GB004	-1.172	-2.063	-2.011	-1.914	-1.787	-1.662	-1.620	-1.505	-1.413	-1.383	-1.331	-1.316
GB006	-1.812	-2.502	-2.396	-2.339	-2.283	-2.260	-2.190	-2.093	-2.057	-1.972	-1.945	-1.922
GB007	-0.814	-1.465	-1.399	-1.310	-1.253	-1.207	-1.191	-1.079	-1.026	-1.001	-1.000	-0.918
GB008	1.032	0.386	0.440	0.521	0.597	0.637	0.701	0.742	0.761	0.834	0.905	0.978
GB010	0.569	-0.410	-0.157	-0.085	-0.016	0.090	0.162	0.203	0.263	0.298	0.388	0.478
GB011	0.514	0.404	0.535	0.634	0.696	0.825	0.920	1.021	1.079	1.095	1.175	1.262
GB012	-0.090	-1.240	-1.161	-1.029	-0.948	-0.898	-0.797	-0.751	-0.740	-0.664	-0.605	-0.524
GB014	0.190	-0.663	-0.619	-0.557	-0.493	-0.422	-0.364	-0.331	-0.243	-0.152	-0.106	-0.084
GB016	-0.038	-1.048	-0.943	-0.901	-0.824	-0.798	-0.772	-0.695	-0.611	-0.604	-0.457	-0.430
GB019	-0.016	-0.871	-0.818	-0.641	-0.633	-0.519	-0.423	-0.320	-0.314	-0.202	-0.171	-0.143
GB022	0.245	-0.879	-0.856	-0.801	-0.738	-0.641	-0.627	-0.558	-0.535	-0.528	-0.467	-0.404
GB023	0.021	-0.343	-0.262	-0.183	-0.130	-0.023	0.014	0.117	0.140	0.177	0.218	0.283
GB024	0.573	-0.453	-0.278	-0.162	-0.097	-0.036	0.065	0.083	0.144	0.225	0.256	0.373
GB025	0.736	0.244	0.336	0.394	0.451	0.539	0.585	0.705	0.758	0.777	0.845	0.949
GB027	0.230	-0.336	-0.265	-0.204	-0.126	-0.071	-0.021	0.069	0.101	0.150	0.198	0.240
GB031	-3.555	-4.368	-4.245	-4.127	-4.043	-3.905	-3.838	-3.772	-3.690	-3.665	-3.638	-3.536
GB032	1.614	0.669	0.798	0.994	1.025	1.052	1.081	1.146	1.211	1.253	1.266	1.334
GB036	0.565	-0.163	0.139	0.255	0.313	0.372	0.465	0.566	0.633	0.664	0.739	0.794
GB039	0.898	-0.333	-0.184	-0.126	-0.085	-0.020	0.040	0.075	0.104	0.149	0.225	0.282
GB043	0.242	-0.534	-0.520	-0.431	-0.366	-0.332	-0.257	-0.234	-0.173	-0.104	-0.066	-0.045
GB045	1.590	0.967	0.981	1.005	1.035	1.061	1.098	1.135	1.184	1.196	1.243	1.267
GB047	-2.124	-2.685	-1.995	-1.731	-1.582	-1.403	-1.285	-1.222	-1.130	-1.073	-0.999	-0.927
GB050	-1.155	-1.664	-1.546	-1.472	-1.443	-1.365	-1.333	-1.301	-1.225	-1.203	-1.157	-1.095
GB051	0.473	0.198	0.249	0.351	0.448	0.542	0.572	0.642	0.685	0.699	0.771	0.837
GB053	-1.321	-2.288	-2.237	-2.071	-2.004	-1.865	-1.736	-1.717	-1.673	-1.601	-1.573	-1.540
GB057	-0.501	-1.118	-1.080	-1.033	-0.956	-0.840	-0.831	-0.764	-0.705	-0.654	-0.605	-0.523
GB058	1.664	0.780	0.844	0.893	0.952	1.016	1.034	1.103	1.157	1.193	1.202	1.231
GB060	0.247	-0.813	-0.786	-0.762	-0.618	-0.551	-0.542	-0.446	-0.406	-0.215	-0.185	-0.157
GB062	0.231	-0.746	-0.437	-0.288	-0.151	-0.037	0.006	0.023	0.139	0.219	0.257	0.297
GB064	0.266	-0.702	-0.462	-0.361	-0.242	-0.170	-0.115	-0.012	0.080	0.144	0.173	0.202
GB065	0.408	-0.059	-0.022	0.049	0.110	0.177	0.267	0.308	0.318	0.386	0.423	0.443
GB066	0.293	-0.650	-0.572	-0.470	-0.425	-0.359	-0.340	-0.257	-0.218	-0.172	-0.164	-0.102
GB068	1.359	0.436	0.555	0.599	0.690	0.749	0.792	0.859	0.893	0.906	0.916	1.002
GB069	0.734	-0.054	0.024	0.092	0.147	0.205	0.237	0.280	0.337	0.383	0.413	0.421
GB070	1.712	0.777	0.922	1.064	1.114	1.199	1.251	1.310	1.354	1.402	1.429	1.513
GB072	-0.294	-0.876	-0.673	-0.502	-0.261	-0.143	-0.036	-0.009	0.110	0.152	0.194	0.307
GB074	0.832	0.071	0.131	0.205	0.271	0.345	0.370	0.432	0.537	0.591	0.603	0.684
GB075	-0.982	-2.047	-1.891	-1.863	-1.818	-1.693	-1.651	-1.540	-1.494	-1.435	-1.374	-1.317
GB081	-0.028	-1.088	-1.064	-1.010	-0.961	-0.906	-0.891	-0.840	-0.815	-0.762	-0.758	-0.692
GB082	-0.637	-0.954	-0.921	-0.856	-0.801	-0.783	-0.746	-0.640	-0.594	-0.547	-0.512	-0.433
True Positve	25	10	12	13	13	14	18	19	21	21	21	21

Tabelle 4.9: Boosting mit einem Text eines anderen Autors: Distanz zur Hyperebene für Textpaare gleicher Autoren auf dem JB16-Korpus. In Abbildung 4.5 ist eine graphische Darstellung nach 40 Boost-Iterationen gezeigt. 100 Obfuskierungs-Iterationen nach Algorithmus 1. Die Spalten zeigen jeweils die Werte nach i Iterationen. Negative Werte entsprechen der Zuordnung der Klasse *unterschiedliche Autoren*. Die letzte Zeile enthält die Anzahl der True Positives.

ID	Original	Obfuskiert	2	4	6	8	10	12	14	16	18	20
GB004	-1.172	-2.063	-2.056	-1.984	-1.931	-1.872	-1.847	-1.793	-1.778	-1.722	-1.665	-1.639
GB006	-1.812	-2.502	-2.460	-2.398	-2.366	-2.280	-2.226	-2.177	-2.113	-2.026	-1.977	-1.957
GB007	-0.814	-1.465	-1.411	-1.320	-1.298	-1.226	-1.172	-1.140	-1.082	-1.055	-1.034	-0.983
GB008	1.032	0.386	0.475	0.564	0.613	0.667	0.711	0.743	0.817	0.860	0.874	0.880
GB010	0.569	-0.410	-0.288	-0.174	-0.112	-0.033	-0.017	0.038	0.076	0.092	0.158	0.202
GB011	0.514	0.404	0.494	0.532	0.605	0.659	0.688	0.775	0.792	0.835	0.879	0.895
GB012	-0.090	-1.240	-1.200	-1.138	-1.037	-0.998	-0.939	-0.916	-0.862	-0.766	-0.730	-0.678
GB014	0.190	-0.663	-0.536	-0.477	-0.384	-0.309	-0.275	-0.156	-0.148	-0.075	0.002	0.060
GB016	-0.038	-1.048	-0.962	-0.847	-0.797	-0.770	-0.695	-0.661	-0.624	-0.595	-0.512	-0.459
GB019	-0.016	-0.871	-0.789	-0.725	-0.663	-0.640	-0.586	-0.565	-0.548	-0.496	-0.473	-0.452
GB022	0.245	-0.879	-0.826	-0.730	-0.682	-0.586	-0.530	-0.520	-0.438	-0.395	-0.357	-0.333
GB023	0.021	-0.343	-0.232	-0.173	-0.049	-0.005	0.043	0.091	0.141	0.166	0.199	0.245
GB024	0.573	-0.453	-0.403	-0.336	-0.240	-0.120	-0.038	0.024	0.025	0.054	0.122	0.139
GB025	0.736	0.244	0.307	0.365	0.412	0.464	0.496	0.524	0.589	0.624	0.734	0.782
GB027	0.230	-0.336	-0.263	-0.162	-0.123	-0.063	-0.018	0.048	0.089	0.094	0.136	0.163
GB031	-3.555	-4.368	-4.259	-4.184	-4.051	-3.940	-3.840	-3.790	-3.659	-3.603	-3.553	-3.479
GB032	1.614	0.669	0.830	0.897	0.916	0.930	0.945	1.014	1.057	1.070	1.146	1.219
GB036	0.565	-0.163	0.058	0.170	0.236	0.302	0.313	0.375	0.452	0.552	0.609	0.613
GB039	0.898	-0.333	-0.256	-0.179	-0.121	-0.085	-0.002	0.022	0.047	0.085	0.159	0.200
GB043	0.242	-0.534	-0.508	-0.471	-0.420	-0.386	-0.339	-0.330	-0.270	-0.236	-0.219	-0.204
GB045	1.590	0.967	0.986	1.006	1.031	1.047	1.097	1.112	1.147	1.163	1.211	1.232
GB047	-2.124	-2.685	-2.553	-2.470	-2.424	-2.387	-2.358	-2.302	-2.259	-2.223	-2.212	-2.165
GB050	-1.155	-1.664	-1.564	-1.501	-1.373	-1.275	-1.233	-1.189	-1.125	-1.068	-1.002	-0.967
GB051	0.473	0.198	0.262	0.337	0.386	0.468	0.523	0.550	0.591	0.662	0.694	0.723
GB053	-1.321	-2.288	-2.204	-2.116	-2.073	-2.064	-2.051	-2.032	-1.957	-1.931	-1.912	-1.867
GB057	-0.501	-1.118	-1.083	-0.963	-0.901	-0.838	-0.777	-0.760	-0.694	-0.647	-0.606	-0.575
GB058	1.664	0.780	0.784	0.786	0.813	0.857	0.870	0.911	0.948	0.990	0.995	1.037
GB060	0.247	-0.813	-0.758	-0.657	-0.583	-0.497	-0.359	-0.276	-0.255	-0.096	0.006	0.034
GB062	0.231	-0.746	-0.694	-0.630	-0.587	-0.463	-0.422	-0.370	-0.333	-0.287	-0.225	-0.208
GB064	0.266	-0.702	-0.640	-0.614	-0.462	-0.394	-0.323	-0.228	-0.178	-0.093	-0.087	0.013
GB065	0.408	-0.059	-0.030	0.024	0.056	0.106	0.122	0.213	0.277	0.312	0.325	0.333
GB066	0.293	-0.650	-0.602	-0.547	-0.504	-0.452	-0.440	-0.379	-0.363	-0.300	-0.253	-0.229
GB068	1.359	0.436	0.487	0.553	0.591	0.682	0.713	0.735	0.758	0.821	0.852	0.858
GB069	0.734	-0.054	0.035	0.098	0.148	0.171	0.198	0.203	0.237	0.259	0.313	0.332
GB070	1.712	0.777	0.816	0.938	1.013	1.037	1.068	1.125	1.158	1.166	1.202	1.237
GB072	-0.294	-0.876	-0.654	-0.568	-0.471	-0.325	-0.251	-0.107	-0.067	0.045	0.097	0.126
GB074	0.832	0.071	0.128	0.188	0.244	0.294	0.369	0.402	0.500	0.576	0.600	0.615
GB075	-0.982	-2.047	-1.771	-1.649	-1.589	-1.527	-1.480	-1.429	-1.391	-1.318	-1.274	-1.219
GB081	-0.028	-1.088	-1.070	-0.968	-0.910	-0.829	-0.776	-0.744	-0.713	-0.637	-0.537	-0.487
GB082	-0.637	-0.954	-0.908	-0.803	-0.764	-0.709	-0.625	-0.611	-0.543	-0.512	-0.489	-0.480
True Positive	25	10	12	13	13	13	14	18	18	19	21	22

Autors jeweils erweitert[5]. Eine vollständige Übersicht über den Korpus ist in Tabelle B.1 gezeigt. Die Annäherung zur ursprünglichen Repräsentation, in Abbildung 4.5a gezeigt, gelingt weniger gut. Die Verteilungen von Boost- und Obfuskier-Text sind hier bereits zu unterschiedlich. Nach 200 Iterationen können elf weitere Texte korrekt klassifiziert werden (siehe Tabelle 4.8).

Schließlich wird das Verhalten untersucht, wenn bei AB fälschlicherweise ein Text eines anderen Autors eingesetzt wird. Die Annäherung sollte hier noch langsamer geschehen, da sich die Verteilungen unterschiedlicher Autoren stärker unterschieden (vgl. [Koppel und Schler, 2004]). Im zuvor erstellten Korpus wurde dazu der Boost-Text durch einen zufälligen Text eines anderen Autors ersetzt. Eine vollständige Übersicht des Korpus ist in Tabelle B.2 gezeigt. Die Annäherung zur ursprünglichen Repräsentation erfolgt in ähnlicher Weise, wie wenn ein Text des gleichen Autors verwendet wird (siehe Abbildung 4.5b). Nach 200 Iterationen werden zwölf weitere Texte als gleich klassifiziert, die Zunahme erfolgt etwas langsamer als wenn ein Text des gleichen Autors verwendet wird (siehe Tabelle 4.9). Die Versuche haben gezeigt, dass wenn der Obfuskierungs-Text nicht vorhanden ist, sich die Repräsentationen weniger leicht rückführen lassen. Dabei bestehen geringe Unterschiede bei Verwendung von Texten des gleichen und eines verschiedenen Autors. Im Mittel unterscheiden sich die Wertsteigerungen der Entscheidungsfunktion wenn der Obfuskierungs-Text verwendet wird signifikant zu den Wertsteigerungen der Entscheidungsfunktion wenn ein anderer Text Verwendet wird (p-Value = 1.34×10^{-19} bei unterschiedlichem Autor, p-Value = 3.93×10^{-19} bei gleichem Autor). Demnach kann der Obfuskierungs-Text als Schlüssel interpretiert werden, der für die Rückführung der Textrepräsentation nach einer Obfuskierung eingesetzt werden kann.

Die Abstände der Samples zur Hyperplane bieten eine gute Orientierung, ob die ursprüngliche Repräsentation angenähert werden konnte oder nicht. Eine übermäßige Annäherung durch AB bewegt alle Samples in den positiven Bereich. Um zu überprüfen, wie stark die Menge der in AB und AO verwendeten Features sich überschneidet, wurde für jedes Paar der Anteil der Features berechnet, die sowohl für die Obfuskierung als auch für das Boostings verwendet wurden. Wird ein Text des gleichen Autors verwendet, liegt die Übereinstimmung bei durchschnittlich 31.58%, bei einem Text eines anderen Autors liegt die Übereinstimmung bei durchschnittlich 26.88%. Damit liegt die Übereinstimmung niedriger als bei Verwendung des gleichen Textes (durchschnittlich 66.21%). Die Übereinstimmungenraten der Features bei Verwendung des gleichen Textes sind über alle Paare hinweg, verglichen mit den ausgewählten Features bei Verwendung eines Textes eines

[5]Korpora, die dieser Form entsprechen sind der PAN14 Essay Korpus [Stamatatos et al., 2014] und der PAN13 Korpus [Juola und Stamatatos, 2013].

unterschiedlichen Autors sowie bei Verwendung eines anderen Textes des gleichen Autors, signifikant unterschiedlich (p-Value = 4.3×10^{-35} bei unterschiedlichem Autor, p-Value = 2.0×10^{-21} bei gleichem Autor).

4.4 Zwischenfazit

Die vorgestellten Versuche haben gezeigt, dass die Textrepräsentationen durch AB so verändert werden können, dass der verwendete Klassifikator SIM-C mehr dazu tendiert, die Klasse *gleicher Autor* zuzuweisen. AB eignet sich nicht als Präprozess, da das beobachtete Verhalten der Klassen *gleicher Autor* und *unterschiedliche Autoren* gleich, und damit keine Steigerung der erkannten Fälle zu erwarten ist. Beim Einsatz nach AO unterscheidet sich die Annäherung bei Verwendung des gleichen Textes von der wenn andere Texte zum AB verwendet werden. Der Obfuskierungs-Text kann als Schlüssel interpretiert werden, da eine Annäherung der ursprünglichen Distanz zur Hyperebene und die Wiederherstellung der Features in vielen Fällen möglich ist. Es ist demnach möglich, mit Boosting den Text zu finden, der zur Obfuskierung verwendet wurde. Der Autor des Obfuskierungs-Textes und der Autor des Original-Textes stimmen für die untersuchten Obfuskierer auf dem JB16-Korpus überein. Aufgrund dessen kann das Verfahren hier dazu verwendet werden, um über den Autor des Obfuskierungs-Textes auf den Autor des Original-Textes zu schließen. Das Verfahren lässt sich auf dem JB16-Korpus nicht auf andere Texte des gleichen Autors erweitern, da sich diese in den untersuchten Fällen weniger annähern und sich das beobachtete Verhalten nicht von dem unterscheidet, wenn ein Text eines anderen Autors verwendet wird. Das Fehlschlagen der Experimente auf den Korpora KW14, PAN14 und PAN15 deutet darauf hin, dass die Ergebnisse korpusbedingt unterschiedlich sein können. Die Textmenge der Korpora KW14 und PAN15 ist zu gering, sodass sich die Effekte des AB zu stark auswirken. Die in Abschnitt 3.7 beschriebenen Besonderheiten des Korpus PAN14 führen dazu, dass das AB keine Wirkung zeigt. Demnach ist die Effektivität des AB abhängig davon, ob die Texte des Trainings- und Test-Korpus lang genug und bezüglich ihrer Länge vergleichbar sind und sich thematisch nicht stark überschneiden. Die Textlänge im untersuchten JB16 Korpus beträgt durchschnittlich 3 879 Wörter und bietet eine gute Orientierung für vergleichbare Experimente.

4.5 Authorship-Boosting Erweiterungen

Die von Bevendorff [2016, 2018] entwickelten AO-Ansätze sowie der AB-Ansatz nehmen in den Texten wenige effektive Änderungen vor, um den Qualitätsmetriken aus Potthast et al. [2016] zu entsprechen. Es kann argumentiert werden, dass dies für eine Vorverarbeitung bzw. für eine Anwendung nach AO vernachlässigt werden kann, da der Einsatz von AB nicht zur Täuschung von AV-Verfahren eingesetzt wird und keine Texte erzeugt werden. Alternative AB-Ansätze können deshalb in Anzahl, Auswahl und Modifizierung der Features vom ursprünglichen Algorithmus abweichen. Im Folgenden werden Anmerkungen und Vorschläge zu zukünftigen AB-Varianten gegeben. Algorithmus 4 zeigt eine Variante mit möglichen Änderungen im AB-Verfahren. Die Varianten wurden beispielhaft mit dem Klassifikator SIM-C auf dem Korpus JB16 untersucht. Die Anzahl der Modifizierungen in AO soll gering gehalten werden, um die Metriken Soundness und Sensibleness zu erfüllen. Wie in Gleichung (3.17) gezeigt, führt eine Erhöhung des Features in der Verteilung Q zu einer Steigerung der KLD, wenn die Häufigkeit in der Verteilung P höher als die Häufigkeit in Q ist. Werden alle Features zum Boosting einsetzt, läuft die KLD gegen Null und der Klassifikator weist die Klasse *gleicher Autor* zu. Die starke Annäherung von P und Q lässt demnach keine Schlüsse über den Obfuskierungs-Text zu (siehe oben). Ein stichprobenartiger Versuch auf dem JB16-Korpus zeigt, dass bereits nach dem ersten Aufrufen der Boosting-Funktion im in Abschnitt 4.3 besprochenen Experiment, keine Unterscheidung der Klassen *gleicher Autor* und *unterschiedliche Autoren* mehr möglich ist. Deshalb werden die Features ausgewählt, die in P häufiger vorkommen als in Q. Aus dieser Menge werden n zufällige Features ausgewählt, deren Häufigkeit in Q erhöht wird. Im Experiment wurden $0,5\%$ der Features ausgewählt. Weiter wird vorgeschlagen, dem Wert des Features in Q den Mittelwert des Features in P und Q zuzuweisen:

$$q = \frac{p+q}{2}. \tag{4.3}$$

Aufgrund dessen nähern sich alle Features in Q dem Wert der Features in P an. Schließlich kann eine relative Annäherung in Bezug auf die Differenz $p - q$ vorgenommen werden (vgl. Kapitel 5). Da die Verteilungen auch hier Gefahr laufen, sich zu stark anzunähern, wird vorschlagen, die Features zufällig zu gewichten oder wie oben, eine zufällige Auswahl von n Features zu treffen.

Im Sharpening Experiment kann in den vorgestellten Variationen keine Verbesserung im Vergleich zum klassischen AB erzielt werden. Beim Einsatz gegen AO erfolgt die Annäherungen schneller und sind bei Verwendung des gleichen Textes mit den Ergebnissen aus Abschnitt 4.3 vergleichbar.

Algorithm 4 Authorship-Boosting-Variante. Seien p und q die Vektoren mit den Häufigkeiten der Features in den Dokumenten d_k und d_u, n die Anzahl der Features, die modifiziert werden und method die Variante des AB. Der Vektor feature_preselection enthält die Features, für die p{feature} $>$ q{feature} gilt. Die Zufallszahlen werden einer diskreten Gleichverteilung entnommen.

Input: vec p, vec q, int n, string method
Output: vec q

if method='default' **then**
  vec feature_preselection := p > q
  vec feature_selection := random_choice(feature_preselection, n)
  q{feature_selection} := q{feature_selection} +1
else if method='mean' **then**
  q := $\frac{\text{p+q}}{2}$
else if method='difference' **then**
  $\Delta_{feature}$:= (p – q) · random_probability(||p||)
  $i := 0$
  for each feature ∈ features **do**
    q{feature} := q{feature}+ $\Delta^i_{feature}$
    $i := i + 1$
  end for
end if

Kapitel 5

Boosting als Unmasking Strategie

Im folgenden Abschnitt wird eine neue Strategie zur Annäherung von Features während des Unmasking-Verfahrens vorgestellt. Das in Abschnitt 3.6 vorgestellte Verfahren zur Annäherung der Verteilungen durch Entnahme von Features wird als (Iterative) Feature Removal (FR) bezeichnet, um das neu entwickelte Verfahren *(Iterative) Feature Leveling* (FL) davon abzugrenzen. Durch die iterative Entnahme der Features, die maximal zu einer Trennung der Chunk-Mengen beitragen, nähern sich die Verteilungen in FR einander an. In Abschnitt 4.2 wurde gezeigt, dass eine Annäherung einer Dokumentenrepräsentation d_u an d_k mit AB erzielt werden kann. Es wird überprüft, ob sich ein mit FR vergleichbarer Effekt zeigt, wenn AB als Strategie zur Annäherung verwendet wird. Weiter werden die Strategien FR und (Iterative) Feature Leveling (FL) verglichen, um festzustellen, welche Fälle sich bzgl. der Klassifikation unterscheiden und ob eine Kombination der Strategien zu besseren Ergebnissen führt. Im folgenden Abschnitt wird das Verfahren erläutert und die Entwicklung anhand von empirischen Experimenten auf dem KS04-Korpus dokumentiert.

Die Evaluation auf dem KS04-Korpus ermöglicht einen guten Vergleich mit den Ergebnissen und Kurvenverläufen von UM (vgl. graphische Darstellung in Abbildung 3.5) der FR-Strategie. Die Ergebnisse des Vergleichs sind in Tabelle 5.1 gezeigt. Auf dem KS04-Korpus erreicht FR eine Accuracy von 0.978 und einen F1-Score von 0.929. Die neu entwickelte Strategie erreicht mit einer Accuracy von 0.989 und einem F1-Score von 0.960 eine ähnlich hohe Performanz. Anschließend wird FL auf den Korpora JB16 und PAN14 evaluiert und gegen FR verglichen. Auf dem JB16-Korpus erreicht UM mit der FR Strategie eine Accuracy von 0.738 und einen F1-Score von 0.727, die Ergebnisse von FL liegen mit einer Accuracy von 0.700 und einem F1-Score von 0.714 leicht darunter. Auf dem PAN14-Korpus wird mit FL eine Accuracy von 0.580 und ein F1-Score von 0.669 erreicht. Die Ergebnisse liegen niedriger, wenn FR verwendet wird (Accuracy von 0.545,

Tabelle 5.1: Vergleich von FL und FR als Unmasking-Strategie. Es wird jeweils das beste Modell gezeigt.

Korpus	FR		FL	
	Accuracy	F1	Accuracy	F1
PAN14	0.545	0.671	**0.580**	**0.669**
JB16	**0.738**	**0.712**	0.700	0.714
KS04	0.978	0.920	**0.989**	**0.960**

F1-Score von 0.675). Die Klassifikations-Güte ist demnach erwartungsgemäß bei längeren Texten höher als bei kürzeren. Auf Korpora mit kürzeren Texten liegt sie vergleichbar mit der des Baseline-Klassifikators SIM-C (vlg. Tabelle 4.1). Die Experimente zeigen, dass eine Unmasking-Strategie, die auf AB basiert, auf ausgewählten Korpora kompetitive Ergebnisse erbringen kann.

5.1 (Iterative) Feature Leveling

Das neue Verfahren ist eine Weiterentwicklung von AB und bedient sich weiterer Eigenschaften der Unmasking-Repräsentationen. Es wird als (Iterative) Feature Leveling bezeichnet. Der Begriff ist angelehnt an Linguistic Leveling (auch: Dialect Leveling [Mougeon et al., 1985]), welches den Prozess der Assimilation und Durchmischung von bestimmten Dialekten, oft hergeführt durch Sprachnormalisierung beschreibt.

In der Repräsentation der Baseline-Klassifikatoren wird für jedes Dokument ein Featurevektor verwendet. Damit unterscheidet sich die Repräsentation von der in Unmasking verwendeten Darstellung. Unmasking teilt die Dokumente in Chunks auf und erstellt jeweils einen Featurevektor mit den $N = 250$ häufigsten Wörtern aus dem Dokumentenpaar. Aus dieser Unterschiedlichkeit ergeben sich mehrere Stellen, an denen AB in abgewandelter Weise eingesetzt werden kann. Ein naiver Ansatz wählt zufällig jeweils einen Chunk aus den Chunk-Modellen aus und boostet nach Algorithmus 2. Da die Chunks nur einen Teil des Gesamttextes und damit des Stils des Autors beschreiben, ist die Berücksichtigung aller charakteristischen Features bezüglich des Gesamttextes nicht garantiert. Die Aufsummierung der Häufigkeit der Features in den n Chunks ermöglicht eine Zusammenfassung zur Feature-Häufigkeit im Dokument $\mathrm{d}\{feature\}$:

$$\mathrm{d}\{feature\} = \sum_{i=1}^{n} \mathrm{d}_i\{feature\}. \tag{5.1}$$

Damit ergibt sich der Dokumentenvektor d aus der Vereinigung der Häufigkeiten der verwendeten Features in den Chunks:

$$\mathrm{d} := \bigcup_{feature \in features} \mathrm{d}\{feature\} \tag{5.2}$$

Alternativ wurde vorgeschlagen, den Mittelwert der Feature-Häufigkeit in den Chunks, welcher einen Schätzer $\widehat{\mathrm{d}\{feature\}}$ für die Anzahl der Features in den Chunks d_i gibt, als Werte des Dokumentenvektors zu verwenden:

$$\widehat{\mathrm{d}\{feature\}} = \frac{\mathrm{d}\{feature\}}{n}. \tag{5.3}$$

Demnach werden Features, die sich in wenigen Chunks häufen, niedriger gewichtet. Gegen die Intuition hat sich dies in initialen Versuchen als nicht effektiv herausgestellt. Eine Erklärung hierfür ist, dass der sich der genannte Effekt auch negativ auswirken kann. Tritt ein Feature gehäuft in wenigen Chunks auf, ist der Wert des resultierenden Schätzers $\widehat{\mathrm{d}\{feature\}}$ sehr gering und demnach wird das Feature weniger oft ausgewählt. Dieser Effekt wurde während der Evaluierung auf dem KS04-Korpus beobachtet, andere Korpora können von diesem Verhalten abweichen. Die Verwendung des Dokumentenvektors d mit Features nach Gleichung (5.1) erlaubt zudem eine einfachere Berechnung von Inkrementen, die relativ zu den Dokumentenvektoren sind (siehe unten).

Die in AB eingesetzte Sortierung der Features wählt mehrheitlich Features aus, deren Summand am größten ist und sich damit am stärksten auf die Summe der KLD auswirkt. Da mehrere Iterationen der FL Strategie durchlaufen werden, besteht keine Notwendigkeit einer fortlaufenden Sortierung innerhalb der *While*-Schleife (vlg. Algorithmus 2, Zeile 3). Aufgrund dessen werden die Sortierung und die Auswahl der Features zu Beginn der Methode durchgeführt. Die beschriebene Auswahl der zu verändernden Features generiert die Features, um eine Annäherung von $\mathrm{D}_q = \mathrm{D}_u$ und $\mathrm{D}_p = \mathrm{D}_k$ zu erzielen[1]. Es fehlt eine Auswahl der Chunk-Modelle, die verändert werden sollen. Dazu wird zunächst eine naive Strategie, die Auswahl zufälliger Chunks aus dem Set der Chunks D_q, gewählt[2]. In FR werden die Features in allen Chunk Modellen entfernt. Die Anzahl der Chunks, die pro Iteration modifiziert werden müssen, um einen zu FR ähnlichen Effekt zu erzielen, muss empirisch ermittelt werden.

[1] Aus Gründen der Konsistenz, vgl. Algorithmus 1 und Algorithmus 2, wird im Folgenden p, q für die Dokumentenvektoren $\mathrm{d_k}, \mathrm{d_u}$ verwendet.

[2] Es wird eine diskrete Gleichverteilung verwendet, d. h. jeder gezogene Chunk hat die gleiche Wahrscheinlichkeit ausgewählt zu werden.

Algorithm 5 (Iterative) Feature Leveling als Unmasking Strategie. Sei X die Matrix mit den Chunk-Modellen der Dokumente d_k, d_u, iterations die Anzahl der Features, die pro Iteration verändert werden und c die Anzahl der Chunks in denen Veränderungen vorgenommen werden und $\Delta_{feature}$ die Stärke der Veränderung.

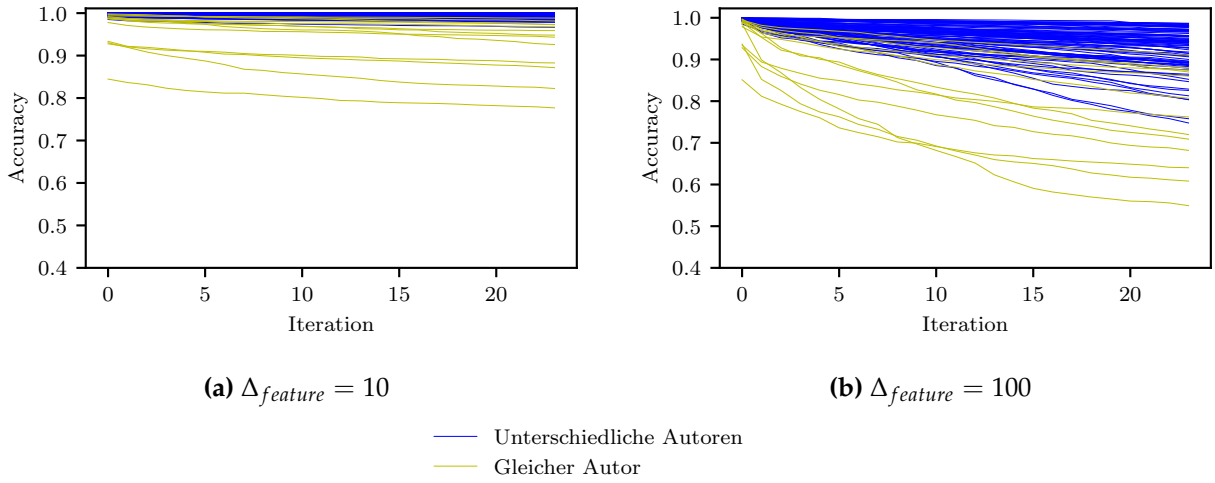
Input: mat X, int iterations, int c, int $\Delta_{feature}$
Output: mat X

```
1: p, q := document_representation(X)
2: vec features := sorted_desc(q, key := func(n): p[n]/q[n])
3: for each feature ∈ features do
4:    chunks := choose(p, random, c)
5:    for each chunk ∈ chunks do
6:       X[chunk, feature] := X[chunk, feature] + Δ_feature
7:    end for
8: end for
```

Das Verfahren ist vereinfacht in Algorithmus 5 gezeigt. Zunächst werden die Dokumentenvektoren p, q aus der Eingabematrix X mit den Chunk-Modellen des Textpaars $< d_k, d_u >$ erzeugt. Dann werden die Features nach Größe ihres Einfluss auf die Summe der KLD sortiert. Für eine vorgegebene Anzahl an Iterationen werden zufällig Chunks aus D_q in der Matrix X ausgewählt und für ein Feature um einen konstanten Faktor $\Delta_{feature}$ erhöht.

Die FL Strategie unterscheidet sich von FR in Einfluss und Auswahl, mit der die Repräsentationen verändert werden. Es werden die Features ausgewählt, deren Anzahl in Dokumentvektor q gering und in Dokumentvektor p hoch ist. Da dies in FR nicht notwendigerweise gegeben ist, werden andere Features ausgewählt. In einer Unmasking-Iteration wurden stichprobenartig Features verglichen, die nach Gewichtung des Einfluss $\mathbf{w}_i$ im Modell der SVM (vlg. FR) und durch die KLD-basierte Selektion entnommen wurden. Der Anteil der von beiden Methoden gleich ausgewählten Features betrug dabei durchschnittlich 12.5%.

Der Grad der Annäherung kann durch die Höhe des Hyperparameters $\Delta_{feature}$ eingestellt werden. In Abbildung 5.1 sind Kurvenverläufe aus Experimenten mit den Inkrementen $\Delta_{feature} = 10$ und $\Delta_{feature} = 100$ auf dem KS04-Korpus als Mittel mit jeweils 500 und 700 Wörtern pro Chunk über 5 Unmasking-Iterationen gezeigt. In jeder Iteration werden $m = 10$ Features ausgewählt. In einem zufälligen Chunk-Modell werden dann die Features erhöht. Aufgrund der Textlänge kommt es zu einer betragsmäßig



(a) $\Delta_{feature} = 10$ **(b)** $\Delta_{feature} = 100$

Abbildung 5.1: FL mit absoluten Häufigkeiten und Konstanten Änderungen $\Delta_{feature}$ auf dem KS04-Korpus. Mittel über fünf Iterationen mit je 500 und 700 Wörter pro Chunk, $m = 10$ Features je Iteration, monotonisiert. **(a)** $\Delta_{feature} = 10$. Accuracy 0.910, F1-Score 0.470. **(b)** $\Delta_{feature} = 100$. Accuracy 0.940, F1-Score 0.700.

höheren Steigung der durch Unmasking erzeugten Kurven, wenn eine höhere Konstante verwendet wird (siehe Abbildung 5.1b). Dies gilt sowohl für Kurven der Klasse *gleicher Autor* als auch der Klasse *unterschiedliche Autoren*, wobei bei Kurven der Klasse *gleicher Autor* ein steilerer Abfall zu beobachten ist. Auffällig ist, dass einige der Kurven der Klasse *gleicher Autor* bereits einen kleineren Anfangswert haben, als Kurven der Klasse *unterschiedliche Autoren*. Die stärkere Reaktion auf höhere Werte von $\Delta_{feature}$ ist auf die Textlänge des Korpus KS04 zurückzuführen: Die Häufigkeit der ausgewählten Features liegt in der Verteilung p zwischen 100 und 1 000. Die Addition der Konstante $\Delta_{feature} = 100$ bewirkt, dass die Verteilungen p, q hinsichtlich des ausgewählten Features gleich bzw. sehr ähnlich werden. Der Wert von $\Delta_{feature}$ ist je nach Korpus unterschiedlich und muss relativ zur Textlänge bzw. zur Varianz der Häufigkeits-Verteilungen angepasst werden. Die Evaluation des Meta-Klassifikators zeigt mit einer Accuracy von 0.940 und einem F1-Score 0.700, dass die Strategie bereits erfolgreich eingesetzt werden kann.

In den vorangegangenen Experimenten wurde wie im originalen Unmasking-Algorithmus jeweils das komplette Dokument verwendet. Wie in Abschnitt 3.6 beschrieben, ergibt sich dadurch eine Einschränkungen bzgl. der Textlänge, da eine Mindestanzahl von 10 000 Wörtern gewährleistet sein muss und keine Mehrfachverwendung von Textabschnitten vorgesehen ist. Durch die hohe Anzahl an Chunks steigt die absolute Anzahl an Features. In Folge dessen muss eine höhere Konstante $\Delta_{feature}$ verwendet werden. Um die Anzahl der Chunks und damit die absoluten Häufigkeiten der Features zu

verringern, kann wie in Abschnitt 3.6 beschrieben, eine zufällige Untermenge der Wörter gezogen werden (siehe [Bevendorff, 2018], auch [Stamatatos, 2008]). Da das Verfahren eine Mehrfachverwendung von Wörtern erlaubt, wird eine Anwendung auf Korpora mit kürzeren Texten wie JB16 und PAN14 möglich. Durch die kleine Anzahl der Features in den Chunks wirken sich kleine Veränderungen bzgl. der Ähnlichkeit stärker aus. Die durch Unmasking erzeugten Kurven werden über mehrere Durchgänge als Mittel dargestellt, um Zufallseinwirkungen zu minimieren.

Bei FR werden Veränderungen in allen Chunk-Modellen vorgenommen. Aufgrund dessen wird untersucht, ob sich eine Erhöhung der Anzahl der Chunks auf das Ergebnis des Meta-Klassifikators auswirkt. Das Experiment wurde mit der oben beschriebenen Methode zur Generierung einer Untermenge der Wörter und der Konstante $\Delta_{feature} = 10$ wiederholt. Die Konstante wird für jedes ausgewählte Feature in drei zufälligen Chunk-Modellen addiert. Damit kann die Accuracy auf 0.95 und der F1-Score auf 0.76 gesteigert werden (vgl. graphische Darstellung in Abbildung A.1).

Wie beschrieben, muss der Hyperparameter $\Delta_{feature}$ je nach Korpus empirisch, z. B. durch eine Grid-Search, ermittelt werden. Um dies zu vermeiden, wird vorgeschlagen die Veränderungen $\Delta_{feature}$ relativ zu einem Zielwert, beispielsweise zum Mittelwert des ausgewählten Features in den Dokumentenvektoren p und q oder dem Wert des ausgewählten Features in Dokumentvektor p, zu berechnen. Ist die Differenz der Häufigkeiten der Features in den Vektoren p und q hoch, führt dies zu hohen, bei geringen Differenzen zu niedrigen Faktoren. Im Folgenden wird der Mittelwert des ausgewählten Features in den Dokumentenvektoren p und q als Zielwert definiert (vgl. Abschnitt 4.4) und dessen Abstand zur boostenden Verteilung q als Änderung $\Delta_{feature}$ verwendet:

$$\begin{aligned} \Delta_{feature} &= \frac{\mathrm{p}\{feature\} + \mathrm{q}\{feature\}}{2} - \mathrm{q}\{feature\} \\ &= \frac{\mathrm{p}\{feature\} - \mathrm{q}\{feature\}}{2} \end{aligned} \tag{5.4}$$

Die Sortierung der Features nach Einflusses auf die Summe der KLD garantiert, dass mehrheitlich Features ausgewählt werden, für die $\mathrm{p}\{feature\} > \mathrm{q}\{feature\}$ gilt und die folglich positive Werte für $\Delta_{feature}$ generieren. Nähert sich die Anzahl der pro Iteration verändernden Features m der Anzahl der verwendeten Features N im Chunk-Modell an, kann es vorkommen, dass die Häufigkeit des Features in Repräsentation p höher als die Repräsentation in q ist und damit $\Delta_{feature}$ negativ wird. Bei $\mathrm{p}\{feature\} = \mathrm{q}\{feature\}$ sind die Verteilungen bzgl. des ausgewählten Features vollständig angenähert und $\Delta_{feature} = 0$. Die

Höhe des Werts von $\Delta_{feature}$ kann durch die Berechnung nach Gleichung (5.4) adaptiv je nach Textlänge bzw. Repräsenation, siehe unten, ermittelt werden. Für den Korpus KS04 wurden in den Experiementen Werte im Bereich $[5, 208]$ berechnet, für den Korpus JB16 lagen die Werte im Bereich $[2, 5]$. Bei Experimenten auf dem Korpus PAN14 lagen die Werte unter 0.1. Da sich die nach Gleichung (5.4) berechnete Differenz auf die Dokumentenvektoren bezieht, wird sie auf c Chunk-Modelle verteilt

$$\Delta^i_{feature} = \frac{\Delta_{feature}}{c}. \tag{5.5}$$

In Bevendorff [2018] wurde das Verhalten von Unmasking auf relative Wort-Häufigkeiten untersucht. Eine ähnliche Repräsentation wird in der Darstellung der Features des Klassifikators SIM-C verwendet. Die erzielten Ergebnisse liegen dabei entgegen der Erwartung unterhalb der Ergebnisse mit absoluten Häufigkeiten. Aufgrund der abweichenden Modifizierungen wird diese Methode erneut untersucht (siehe unten).

Die Auswahl der Chunks erfolgt in den bisher dokumentierten Experimenten zufällig aus dem Set der Chunks D_q, die aus dem Dokument d_u generiert wurden. Da die Chunk-Menge D_u zum Zeitpunkt der Ausführung des Algorithmus bekannt ist, können Informationen über die Häufigkeiten der Features in den einzelnen Chunks verwendet werden, um eine Auswahl der Chunks zu treffen. Liegt die Häufigkeit des Features im ausgewählten Chunk ähnlich hoch wie in Dokumentenvektor p, wirken sich die Modizierungen weniger auf die Ähnlichkeit, bzw. die Trennung der Dokumentenvektoren aus. Um Chunks auszuwählen, in denen der Effekt der Modifizierung maximal ist, können die Chunks ausgewählt werden, in denen die Häufigkeit des ausgewählten Features am niedrigsten ist. Seien $D^i_q, 1 \leq i \leq N$ die Chunk-Modelle der zur verändernden Repräsentation und c die Anzahl der Chunks, in denen Veränderungen vorgenommen werden. Dann enthält der Vektor $(\alpha_1, \ldots, \alpha_N)$ die Indizes der Chunk-Modelle, in denen Veränderungen vorgenommen werden:

$$\begin{aligned} (\alpha_1, \ldots, \alpha_N) := \arg\min_{\alpha_i, \forall i} \quad & \sum_{i=1}^{N} \left(\alpha_i \cdot D^i_q\{feature\} \right) \\ \text{s. t.} \quad & \sum_{i}^{N} \alpha_i = c \\ & \alpha_i \in [0, 1] \end{aligned} \tag{5.6}$$

Das Optimierungsproblem (5.6) kann durch Hinzunahme des Zielwerts erweitert werden, sodass die Chunks ausgewählt werden, deren Häufigkeit für das Feature am weitesten

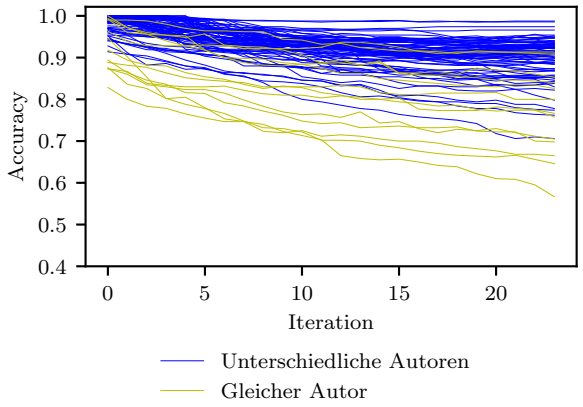


Abbildung 5.2: FL relative Häufigkeiten auf dem KS04-Korpus. Mittel über fünf Iterationen mit 500 Wörtern je Chunk, $m = 10$ Features je Iteration, Änderung $\Delta_{feature}$ nach Gleichung (5.5) und Verteilung auf je drei Chunks. Chunk Auswahl nach Gleichung 5.7. Accuracy 0.914, F1-Score 0.556.

unter dem Zielwert liegen:

$$\begin{aligned} (\alpha_1, \ldots, \alpha_N) := \arg\max_{\alpha_i, \forall i} \quad & \sum_{i=1}^{N} \alpha_i \cdot \Big(target\{feature\} - \mathrm{D}_q^i\{feature\} \Big) \\ \text{s. t.} \quad & \sum_{i}^{N} \alpha_i = c \\ & \alpha_i \in [0, 1] \end{aligned} \tag{5.7}$$

Als Zielwert wird der Mittelwert nach Gleichung (5.1) verwendet. Die Ergebnisse liegen mit einer Accuracy von 0.914 und einem F1-Score von 0.556 unter den bisherigen, aber im gleichen Bereich, siehe Abbildungen 5.2.

Wie in Abschnitt 3.4 beschrieben, ist die als Grundlage für die Sortierung dienende KLD asymmetrisch, d. h. $\mathrm{KLD}(\mathrm{p}||\mathrm{q}) \neq \mathrm{KLD}(\mathrm{q}||\mathrm{p})$. Infolgedessen bestimmt die Zusammensetzung der Dokumentenvektoren p und q, welche Features ausgewählt werden[3]. Wie im FR besteht keine Restriktion, dass nur eine Verteilung verändert werden darf. Aufgrund dessen werden die Dokumentenvektoren p und q in jeder Iteration ausgetauscht, sodass sich die Chunk-Mengen abwechselnd einander annähern. In Abbildung 5.3 sind die Kurvenverläufe von zwei Experimenten gezeigt, in denen p und q in jeder Iteration gewechselt werden. Sowohl die Kurven der Klasse *gleicher Autor* als auch die Kurven der Klasse *unterschiedliche Autoren* zeigen eine betragsmäßig höhere Steigung (siehe Abbildung 5.3a). Der Meta-Klassifikator kann die Kurven jedoch mit

[3] Da $\max \frac{\mathrm{p}\{feature\}}{\mathrm{q}\{feature\}} = \min \frac{\mathrm{q}\{feature\}}{\mathrm{p}\{feature\}}$.

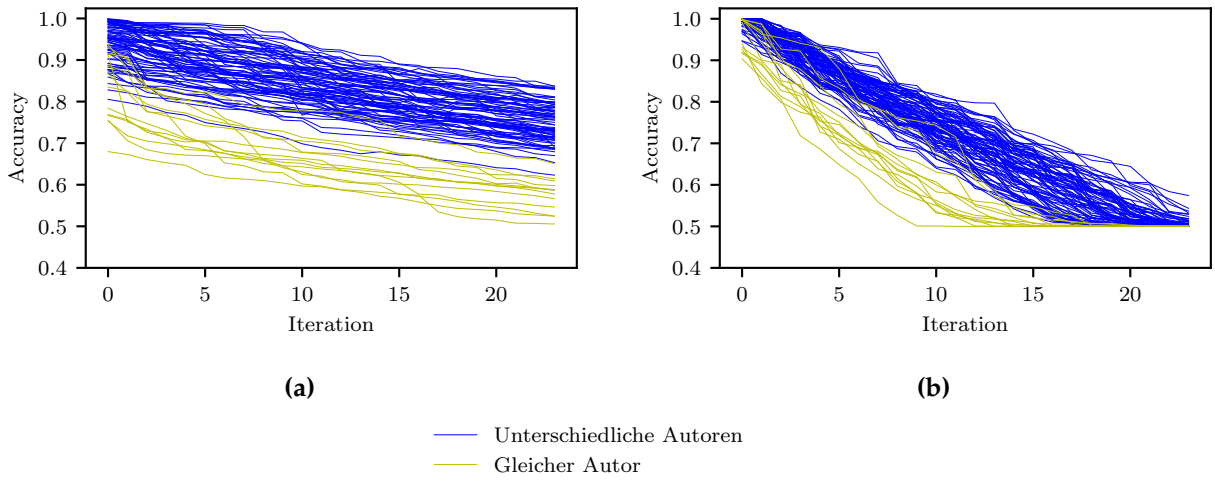


Abbildung 5.3: FL relative Häufigkeiten auf KS04-Korpus. Mittel über fünf Iterationen mit 500 und 700 Wörtern je Chunk, **(a)** $m = 10$ Features je Iteration, Änderung $\Delta_{feature}$ nach Gleichung (5.5) und Auswahl von drei Chunks nach Gleichung (5.6). Der Text der Dokumente wird in Chunks geteilt. Accuracy 0.989, F1-Score 0.960. **(b)** $m = 30$ Features je Iteration, Verteilung auf je drei Chunks, monotonisiert. Die Wörter werden zufällig mit Zurücklegen aus Text gezogen und auf 25 Chunks verteilt. Accuracy 0.980, F1-Score 0.917.

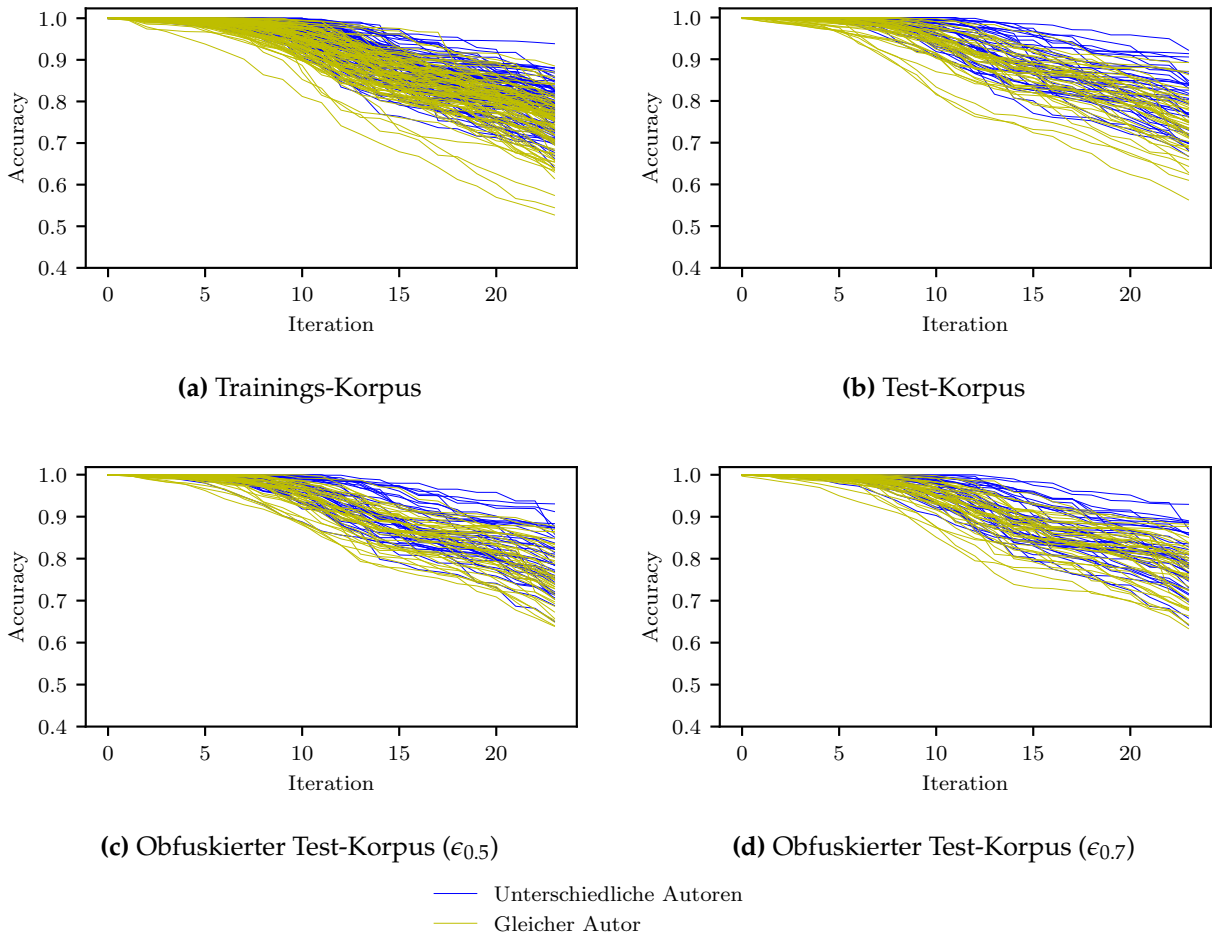
einer Accuracy von 0.989 und einem F1-Score von 0.960 sehr gut auseinander halten. Eine ähnlich hohe Performanz wird bei Verwendung einer zufälligen Untermenge der Wörter, in Abbildung 5.3b dargestellt, erreicht. Der Vergleich zeigt, dass beide Methoden ähnlich gut auf den langen Texten des KS04-Korpus funktionieren. Die Kurven der Klasse *unterschiedliche Autoren* liegen mehrheitlich über den Kurven der Klasse *gleicher Autor*. Letztere fallen besonders während der ersten Iterationen am stärksten. Der Abfall der Kurven ist wie in den vorangegangenen Experimenten geringer als im klassischen FR. Aufgrund dessen kann argumentiert werden, dass der Einfluss der FL Methode nur zum Teil für die Ergebnisse verantwortlich ist und die Ergebnisse korpusbedingt sind. Um dies zu überprüfen werden weitere Korpora untersucht.

Der Korpus KS04 eignet sich zur Entwicklung der FL-Strategie, da sich die Ergebnisse mit denen der FR-Strategie vergleichen lassen. Längere Texte lassen sich aufgrund der größeren Datenlage leichter verifizieren [Stamatatos, 2016]. Im Folgenden wird FL auf Korpora mit kürzeren Texten getestet und damit der Schwierigkeitsgrad erhöht. Zunächst wird der JB16-Korpus untersucht. Eine graphische Darstellung der während des Unmasking generierten Kurven ist in Abbildung 5.4 gezeigt. Der Meta-Klassifikator wird auf den Kurvenverläufen des Train-Sets trainiert und auf den Kurvenverläufen des Test-Sets evaluiert. Die Ergebnisse liegen mit einer Accuracy von 0.700 und einem F1-Score von

0.714 unterhalb denen des zuvor untersuchten KS04-Korpus. Die Ergebnisse sind sowohl mit denen der Baseline-Klassifikatoren (SIM-C auf tf Repräsentation ein Accuracy von 0.700, auf tf idf Repräsentation eine Accuracy von 0.680) als auch mit den Ergebnissen von Unmasking mit der FR-Strategie (Accuracy: 0.713, F1 Score: 0.722) vergleichbar. In den ersten Unmasking Iterationen liegt die Accuracy für alle betrachteten Fälle bei 1.0, d. h. die SVM kann die Chunk-Mengen D_p und D_q zu Beginn gut trennen. Durch die gleichen Anfangsbedingungen sind die Features des Meta-Klassifikators stärker von der Art, bzw. der Steigung des Falls abhängig als in den zuvor gezeigten Versuchen auf dem KS04 Korpus. Einige der Kurven der Klasse *gleicher Autor* fallen schneller, die Mehrheit der Kurven ist jedoch visuell weniger stark von denen der Klasse *unterschiedliche Autoren* zu unterscheiden.

Der in Bevendorff [2018] implementierte Obfuskierer bietet eine Möglichkeit, FL bezüglich seiner Robustheit gegen AO zu untersuchen. In Abbildungen 5.4c und 5.4d sind die Kurven gezeigt, die mit FL auf dem obfuskierten JB16 Test-Korpus erzeugt wurden. Die Accuracy fällt durch die Obfuskierung von 0.700 auf 0.625 bzw. 0.535. Abbildungen A.3c und A.3d illustrieren das Kurven-Verhalten, wenn FR verwendet wird. Hier fällt die Accuracy durch die Obfuskierung von 0.738 auf 0.650. Auffällig ist, dass sich die stärkere Obfuskierung nicht weiter auf die Metriken des Meta-Klassifikators auswirken (siehe Abbildung A.3d). Demnach wirkt sich die stärkere Obfuskierung ($\epsilon_{0.7}$) stärker auf FL als auf FR aus. Dies kann damit begründet werden, dass eine Abhängigkeit zwischen FL den unterliegenden Obfuskierungs-Methoden in Bevendorff [2018] besteht. Die Auswahl der Features bei FR hängt weniger stark mit der Obfuskierungs-Methode zusammen und zeigt auf Grund dessen weniger Anfälligkeiten.

Weiter wird das Verhalten von FL auf dem PAN14-Korpus untersucht. Der PAN14-Korpus eignet sich ebenfalls zu einem Vergleich mit anderen AV-Verfahren, da er Bestandteil mehrerer Shared Tasks war. Die während Unmasking generierten Kurven sind für Trainings- und Test-Korpus in Abbildung 5.5 gezeigt. Bezüglich der Accuracy ist das Ergebnis von FL im Mittelfeld der eingereichten Klassifikatoren einzuordnen (siehe Stamatatos et al. [2014]). Bei der Evaluation wird eine Accuracy von 0.580 und eine F1-Score von 0.669 erreicht (siehe Tabelle 5.1). Die Ergebnisse der FR-Strategie liegen mit einer Accuracy von 0.530 und einem F1-Score von 0.671, in Abbildung A.2 gezeigt, deutlich darunter. Die Ergebnisse der Baseline-Klassifikatoren liegen mit einer Accuracy von 0.520 (SIM-C) bzw. 0.560 (AD-C) ebenfalls leicht darunter. Beim Vergleich der während des Trainings mit Unmasking erzeugten Kurven fällt auf, dass die Kurven-Verläufe, ein zu FR ähnliches Verhalten zeigen (vgl. Abbildung 5.5a und Abbildung A.2a). Die Kurven der Klassen *gleicher Autor* und *unterschiedliche Autoren* können in beiden Verfahren gut getrennt werden, was zu



(a) Trainings-Korpus

(b) Test-Korpus

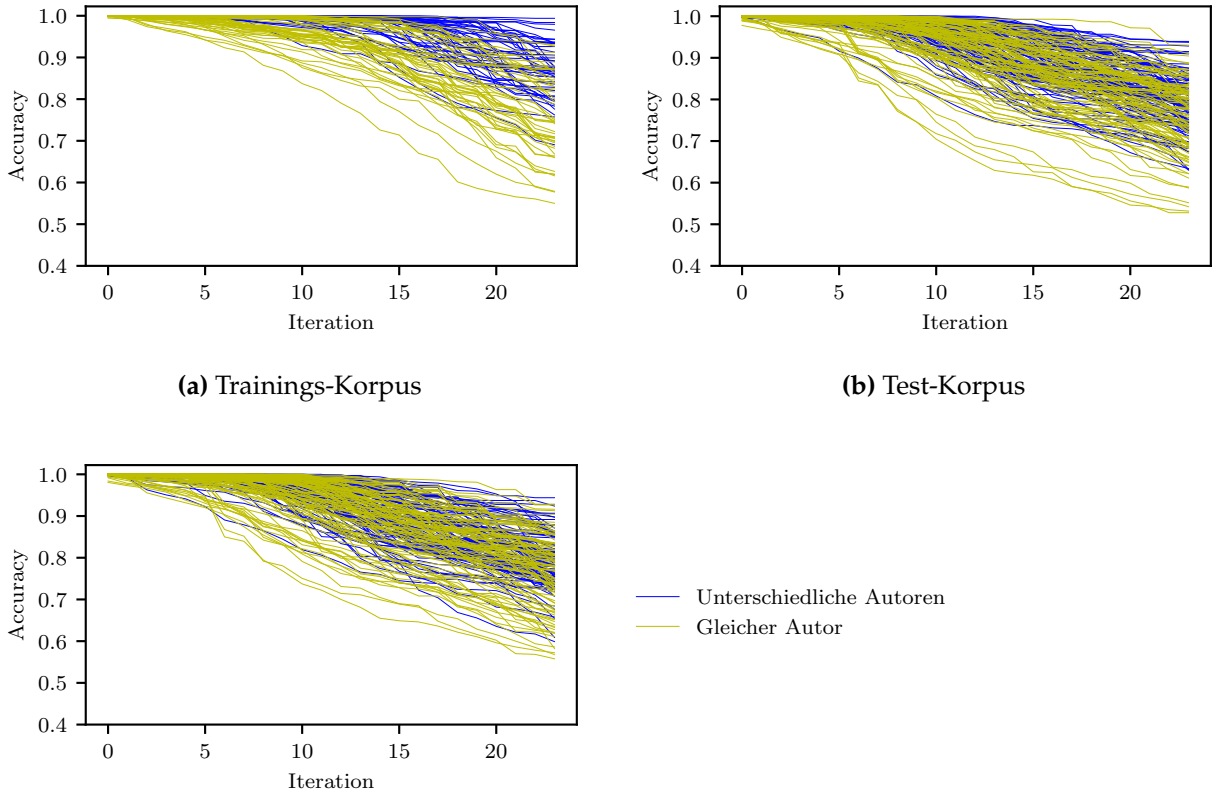
(c) Obfuskierter Test-Korpus ($\epsilon_{0.5}$)

(d) Obfuskierter Test-Korpus ($\epsilon_{0.7}$)

Abbildung 5.4: FL absolute Häufigkeiten auf dem JB16-Korpus. Mittel über zehn Iterationen mit je 400 und 500 Wörtern pro Chunk, $m = 10$ Feature je Iteration, Änderung $\Delta_{feature}$ nach Gleichung (5.5) und Auswahl von drei Chunks nach Gleichung (5.6), monotonisiert. **(a)** Train-Korpus. Crossvalidierungs-Accuracy: 0.649 **(b)** Test-Korpus Accuracy 0.700, F1 0.714 **(c)** Obfuskierter Test-Korpus ($\epsilon_{0.5}$): Accuracy 0.625, F1 0.605. (**d**) Obfuskierter Test-Korpus ($\epsilon_{0.7}$): Accuracy 0.535, F1 0.507.

einer Crossvalidierungs-Accuracy von 0.780 bei FL und 0.760 bei FR führt. Eine ähnlich stark abweichendes Verhalten zwischen Trainings- und Test-Korpora konnte bereits bei der Evaluation des Klassifikators SIM-C beobachtet werden und kann mit der unterschiedlichen Zusammensetzung der Texte begründet werden.

Auch hier wurde die Robustheit gegen das AO-Verfahren aus Bevendorff [2018] untersucht. Die Kurven des obfuskierten Korpus sind in Abbildung 5.5c für FL und in Abbildung A.2c gezeigt. Die Accuracy fällt bei FL von 0.580 auf 0.530 und bei FR von 0.545 auf 0.530.



(a) Trainings-Korpus

(b) Test-Korpus

(c) Obfuskierter Test-Korpus ($\epsilon_{0.5}$)

Abbildung 5.5: FL absolute Häufigkeiten auf dem PAN14-Korpus. Mittel über fünf Iterationen mit je 300, 400 und 600 Wörtern pro Chunk, $m = 10$ Features je Iteration, Änderung $\Delta_{feature}$ nach Gleichung (5.5) und Auswahl von fünf Chunks nach Gleichung (5.6), monotonisiert. **(a)** Trainings-Korpus. Crossvalidierungs-Accuracy 0.780 **(b)** Test-Korpus. Accuracy 0.580, F1 0.669 **(c)** Obfuskierter Test-Korpus ($\epsilon_{0.5}$): Accuracy 0.535, F1 0.635.

Zwischenfazit

FL zeigt auf den untersuchten Korpora eine ähnlich hohe Performanz wie FR. Wie beim klassischen FR Verfahren sind die Ergebnisse auf dem Korpus KS04 mit längeren Texten und dem Korpus JB16, dessen Texte einheitlichen Längen haben und ohne Mehrfachverwendung der Texte auskommen, im Vergleich zu PAN14 hoch. Die neue Strategie lässt sich demnach alternativ zu FR einsetzen. Die Werte der verwendeten Hyperparameter müssen je nach Korpus bestimmt werden. Verglichen mit FR besitzt FL deutlich mehr Komponenten, die angepasst werden können, um das Ergebnis zu verbessern. Daraus folgt auch, dass die Bestimmung von geeigneten Parametern mit FL aufwendiger ist. Die FL-Strategie ist weniger robust gegenüber AO als die FR-Strategie, was damit begründet wird dass eine Abhängigkeit zwischen Obfuskierer und Verifizierungs-Verfahren besteht.

Tabelle 5.2: Vergleich der unterschiedlichen Klassifikationen und skalierten absoluten Werte der Entscheidungsfunktion bei Verwendung der FR- und FL-Strategie auf dem KS04-Korpus.

Author A	Author B	Feature Leveling		Feature Removal		Korrekt
		Klasse	Distanz	Klasse	Distanz	
H. Melville	H. Melville	gleich	0.125	unterschiedlich	0.102	FL
B. Shaw	B. Shaw	unterschiedlich	0.128	gleich	0.091	FR
B. Shaw	B. Shaw	unterschiedlich	0.140	gleich	0.249	FR
A. Bronte	C. Bronte	unterschiedlich	0.211	gleich	0.000	FL

5.2 Gegenüberstellung der Strategien

Im folgenden Abschnitt werden die einzelnen Klassifikations-Ergebnisse von FL und FR verglichen. In Tabelle 5.1 sind Accuracy und F1-Score des jeweils besten Modells gezeigt. Das beste Modell wurde mittels einer Suche über die in Abschnitt 5.1 vorgestellten Parameter (Grid-Search) ermittelt. Die Ergebnisse von FL und FR liegen jeweils ähnlich hoch. Es wird untersucht, in welchen Fällen sich die Klassifikationen unterscheiden und ob durch die Kombination der Verfahren eine Verbesserung zu erwarten ist. Die Klassifikationen der Probleme werden mittels ihrer zugewiesenen Klasse und dem Wert der Entscheidungsfunktion der SVM verglichen. Für jedes Problem im betrachteten Korpus wird dazu die vorhergesagte Klasse und der skalierte absolute Wert ihrer Entscheidungsfunktion ermittelt[4]. Dieser Wert wird als Sicherheit der Klassifikation interpretiert.

Zunächst werden die unterschiedlichen Klassifikationen auf dem KS04 untersucht und mit den in Koppel und Schler [2004] beschriebenen Klassifikatoren verglichen. Eine Auflistung der unterschiedlichen Fälle ist in Tabelle 5.2 gezeigt. Die in Abschnittt 3.6 erläuterten Abweichungen der Implementierung bei FR führen im Vergleich zu Koppel und Schler [2004] zu Differenzen in Anzahl der Samples und Auswahl der Klassen. Die hohe Accuracy bei FL und FR führt dazu, dass 95/99 der Probleme gleich klassifiziert werden. Bei FL werden zwei Text-Paare des Autors Bernard Shaw fälschlicherweise der Klasse *unterschiedliche Autoren* mit niedrigen Sicherheiten von 0.012 und 0.014 zugewiesen. In Koppel und Schler [2004] werden die Bücher der Schwestern Charlotte und Anne Brontë der Klasse *gleicher Autor* zugeordnet. Dies wird damit begründet, dass sie aufgrund ihres ähnlichen sozialen Umfelds einen ähnlichen Schreib-Stil entwickelt haben. Das Text-Paar wird mit der FL-Strategie mit einer Sicherheit von 0.210 korrekt der Klasse *unterschiedliche Autoren* zugewiesen, die FR-Strategie weist die Klasse *gleicher Autor* mit einer

[4]Die Werte werden mit der Min-Max-Skalierung $y' = \frac{y - \min(y)}{\max(y) - \min(y)}$ auf Intervall $[0, 1]$ skaliert.

Tabelle 5.3: Vergleich der unterschiedlichen Klassifikationen und skalierten absoluten Werte der Entscheidungsfunktion bei Verwendung der FR- und FL-Strategie auf dem JB16-Korpus.

Fall	Feature Leveling		Feature Removal		Korrekt
	Klasse	Distanz	Klasse	Distanz	
GB022	gleich	0.002	unterschiedlich	0.149	FL
GB002	gleich	0.011	unterschiedlich	0.345	FR
GB033	gleich	0.029	unterschiedlich	0.213	FR
GB054	unterschiedlich	0.057	gleich	0.254	FL
GB053	gleich	0.070	unterschiedlich	0.078	FL
GB055	unterschiedlich	0.080	gleich	0.103	FL
GB021	gleich	0.085	unterschiedlich	0.151	FR
GB078	gleich	0.089	unterschiedlich	0.166	FR
GB034	gleich	0.120	unterschiedlich	0.185	FR
GB015	gleich	0.142	unterschiedlich	0.092	FR
GB065	unterschiedlich	0.174	gleich	0.180	FR
GB047	gleich	0.199	unterschiedlich	0.552	FL
GB003	gleich	0.214	unterschiedlich	0.082	FR
GB006	gleich	0.275	unterschiedlich	0.172	FL
GB050	unterschiedlich	0.298	gleich	0.057	FR

Sicherheit von 0 zu. Eine Zuweisung mit Sicherheit 0 ist möglich, da nach Gleichung (3.5) $\mathbf{w}^T\mathbf{u} + b \geq 0$ then $+$ gilt. Ein Text-Paar von Herman Melville wird von der FR-Strategie mit einer Sicherheit von 0.102 als Element der Klasse *unterschiedliche Autoren* zugewiesen, die FL-Strategie weist korrekt die Klasse *gleicher Autor* mit Sicherheit 0.140 zu.

Weiter werden die Klassifikations-Unterschiede auf dem JB16 Test-Korpus untersucht. In Tabelle 5.3 ist eine tabellarische Ansicht der 15 unterschiedlichen Klassifikationen gezeigt. Die FL-Strategie weist davon in 9 Fällen die falsche Klasse und 6 Fällen die richtige Klasse zu. Von den korrekt klassifizierten Fälle werden 4 Fälle mit Sicherheiten im Intervall $[0.002, 0.275]$ der Klasse *gleicher Autor* und drei Fälle mit Sicherheiten $0.057, 0.199$ und 0.080 der Klasse *unterschiedliche Autoren* zugewiesen. Für die FR gilt jeweils das komplementäre Ergebnis der Klassifikation. Die Werte der Sicherheiten liegen bei den korrekten Zuweisungen der Klasse *unterschiedliche Autoren* im Intervall $[0.082, 0.345]$ und damit deutlich höher. Bei den falschen Zuweisungen der FL-Strategie werden 2 Fälle mit Sicherheiten von 0.174 und 0.298 der Klasse *unterschiedliche Autoren* zugewiesen und 7 Fälle der Klasse *gleicher Autor* mit Sicherheiten im Intervall $[0.011, 0.275]$ zugewiesen. Auf dem JB16 Test-Korpus besteht bei Verwendung der FL Strategie die Tendenz die Klasse *gleicher Autor* zuzuweisen.

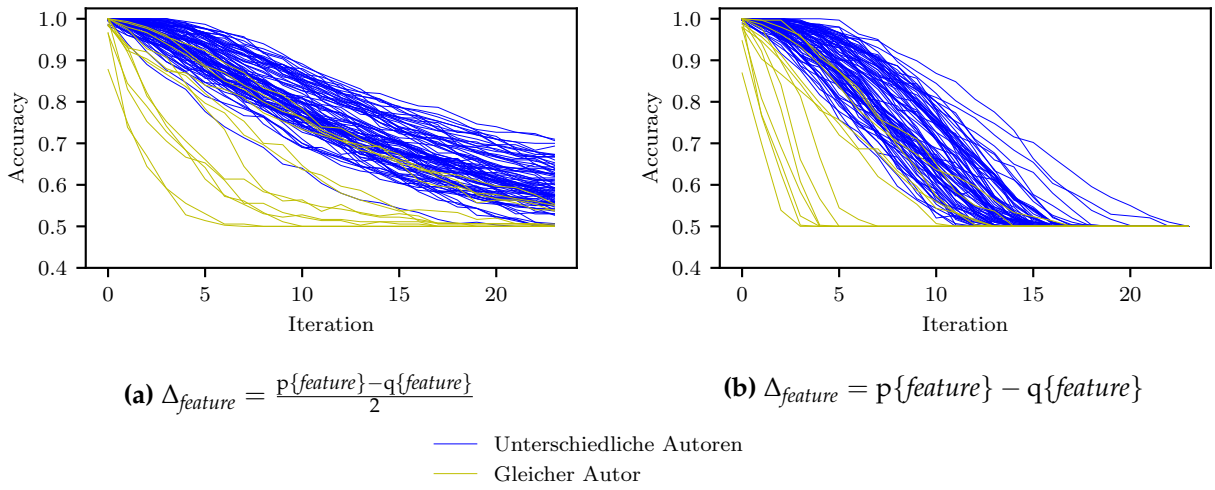
Bei den Klassifikationen, die auf dem Test-Korpus von PAN14 gemacht werden, liegen in 25/200 Fällen Unterschiede vor (vgl. Tabelle 5.4). Auffällig ist, dass FL bis auf einen Fall, der korrekt zugewiesen wird, die Klasse *unterschiedliche Autoren* zuweist. Bei den 9 falsch zugewiesenen Textpaaren liegen die Sicherheiten im Bereich von 0.001 und 0.152. Die 15

Tabelle 5.4: Vergleich der unterschiedlichen Klassifikationen und skalierten absoluten Werte der Entscheidungsfunktion bei Verwendung der FR- und FL-Strategie auf dem PAN14-Korpus.

Fall	Feature Leveling		Feature Removal		Korrekt
	Klasse	Distanz	Klasse	Distanz	
EN134	unterschiedlich	0.001	gleich	0.328	FR
EN150	unterschiedlich	0.004	gleich	0.149	FL
EN193	unterschiedlich	0.004	gleich	0.156	FL
EN257	unterschiedlich	0.005	gleich	0.350	FL
EN277	unterschiedlich	0.005	gleich	0.212	FL
EN125	unterschiedlich	0.008	gleich	0.032	FR
EN179	unterschiedlich	0.012	gleich	0.113	FL
EN214	gleich	0.017	unterschiedlich	0.082	FR
EN182	unterschiedlich	0.018	gleich	0.402	FR
EN168	unterschiedlich	0.029	gleich	0.420	FL
EN104	unterschiedlich	0.043	gleich	0.294	FL
EN212	unterschiedlich	0.046	gleich	0.285	FR
EN123	unterschiedlich	0.050	gleich	0.065	FL
EN122	unterschiedlich	0.059	gleich	0.242	FL
EN274	unterschiedlich	0.062	gleich	0.008	FL
EN185	unterschiedlich	0.068	gleich	0.019	FL
EN216	unterschiedlich	0.074	gleich	0.262	FR
EN177	unterschiedlich	0.083	gleich	0.002	FL
EN249	unterschiedlich	0.103	gleich	0.117	FR
EN158	unterschiedlich	0.108	gleich	0.212	FL
EN290	unterschiedlich	0.109	gleich	0.320	FL
EN152	unterschiedlich	0.110	gleich	0.370	FL
EN245	unterschiedlich	0.117	gleich	0.289	FR
EN200	unterschiedlich	0.138	gleich	0.350	FL
EN242	unterschiedlich	0.152	gleich	0.051	FR

korrekt zugewiesenen Textpaare der Klasse *unterschiedliche Autoren* werden mit Sicherheiten im Intervall $[0.004, 0.138]$ klassifiziert. Auf dem PAN14 Test-Korpus besteht bei Verwendung der FL Strategie eine starke Tendenz die Klasse *unterschiedliche Autoren* zuzuweisen.

Die Werte liegen, mit Ausnahme des KS04-Korpus, bei Verwendung von FL deutlich niedriger als bei Verwendung von FR. Es ist kein direkter Zusammenhang zwischen den Werten der beiden Strategien sichtbar. Vergleicht man die Textlängen in den Test-Korpora, fällt auf, dass FL auf den längeren Texten in PAN14 und KS04 eher die Klasse *unterschiedliche Autoren* zuweist und auf kürzeren Texten die Klasse *gleicher Autor*. Die einseitige Klassifizierung bei PAN14 kann auch damit begründet werden, dass sich Trainings- und Test-Korpus zu stark unterscheiden (siehe Abschnitt 3.7) und FL empfindlicher gegenüber Schwankungen der Textlänge ist. Wenn die Zahl der False Positives gering gehalten werden soll, sollte für längere Texte FL verwendet werden und für kürzere Text FR verwendet werden.



(a) $\Delta_{feature} = \frac{\mathrm{p}\{feature\} - \mathrm{q}\{feature\}}{2}$ **(b)** $\Delta_{feature} = \mathrm{p}\{feature\} - \mathrm{q}\{feature\}$

Abbildung 5.6: FL mit Feature Gewichtung vlg. FR auf dem KS04-Korpus auf absoluten Häufigkeiten. Mittel über fünf Iterationen mit 500 Wörtern pro Chunk, Auswahl von jeweils drei Chunks, $m = 6$ Features je Iteration **(a)** Änderung $\Delta_{feature} = \frac{\mathrm{p}\{feature\} - \mathrm{q}\{feature\}}{2}$. Accuracy 0.950, F1 0.78 **(b)** Änderung $\Delta_{feature} = \mathrm{p}\{feature\} - \mathrm{q}\{feature\}$. Accuracy 0.94, F1-Score 0.700.

5.3 Alternative Strategien

Die Klassifikations-Güte von FL und FR ist auf den untersuchten Korpora vergleichbar, es liegen jedoch z. T. Unterschiede in der Klassifizierung vor. Deshalb wird untersucht, ob eine Kombination der Verfahren ein ähnliches oder besseres Ergebnis erzielen kann. Dies kann durch einen Austausch der Selektions- und Modifizierungsmethoden im FL-Algorithmus oder durch einen *Ensemble-Klassifikator*, der die Ergebnisse von FL und FR kombiniert (vlg. Juola und Stamatatos [2013], Stamatatos et al. [2014, 2015]) erreicht werden. Beide Herangehensweisen werden im folgenden Abschnitt beschrieben und evaluiert.

Es wird untersucht, ob sich die in FR vorgenommene Auswahl der Features gemäß der Höhe des Betrags der Gewichtung $|\mathbf{w}_i|$ der SVM, siehe Gleichung 3.18, mit den in Abschnitt 5.1 vorgestellten Methoden kombinieren lässt. Die Auswahl der Features gibt keine Informationen darüber, in welchem Chunk-Set Veränderungen vorgenommen werden müssen, damit es zu einer Annäherung der Chunk-Mengen D_p und D_q kommt. Es wird das Chunk-Set ausgewählt, in dem die Häufigkeit des ausgewählten Features niedriger ist. In ersten Experimenten zeigt sich, dass die Modifizierung eines Features i nicht mit einer Reduzierung der Gewichtung von $\mathbf{w}_i$ einhergeht, die ausreicht, um die Sortierung von $\mathbf{w}$ zu verändern. Um zu vermeiden, dass nur eine geringe Menge an Features ausgewählt wird, wird jedes Feature nur einmal verwendet. Abbildung 5.6 zeigt zwei Experimente, in denen die Features auf diese Weise ausgewählt wurden. Einige der Kurven der Klasse *gleicher*

Autor fallen bereits in den ersten Iterationen sehr steil ab. Wird statt der Differenz zum Mittelwert der Features (siehe Gleichung (5.4)) die Differenz $p\{feature\} - q\{feature\}$ auf die gesetzte Anzahl Chunks verteilt, verstärkt sich der Effekt. Die Klassifikations-Güte liegt bei der Evaluation der abweichenden Modelle unterhalb der in Abschnitt 5.1 vorgestellten Ergebnisse.

Stamatatos et al. erstellen aus den eingereichten Klassifikatoren der AV Shared Tasks einen Meta-Klassifikator. Der Entscheidungswert eines Samples wird dabei als Mittelwert über die Klassifikatoren berechnet. Der Meta-Klassifikator ist bei der Evaluation immer unter den besten Ergebnissen [Juola und Stamatatos, 2013, Stamatatos et al., 2014, 2015]. In Abschnitt 5.2 wurde gezeigt, dass die Entscheidungen unterschiedlich sein können. Ein Meta-Klassifikator lässt sich auch für die zuvor vorgestellten Unmasking-Strategien entwerfen. Da für die Klassifikatoren Unabhängigkeit gelten soll, wird nur das beste FL-Modell nach den in Abschnitt 5.1 beschriebenen Methoden und FR dafür eingesetzt. Im Rahmen dieser Arbeit wird überprüft, ob durch eine Gewichtung der Verfahren eine Steigerung der Performanz zu erwarten ist. Auf dem Korpus KS04 konnte durch eine stärkere Gewichtung von FL gegenüber FR ein weiteres Textpaar des Autors Bernard Shaw zugewiesen werden (0.64/0.36). Auf dem Korpus PAN14 führt eine sehr hohe Gewichtung von FL ebenfalls zu einer weiteren korrekten Klassifikation (0.99/0.01). Auf dem Korpus JB16 kommt es zu keiner Verbesserung durch die Kombination der Verfahren, die besser als FR ist. Die hohe einseitige Gewichtung der Verfahren zeigt, dass sich die verwendeten Unmasking-Strategien FL und FR nicht systematisch ergänzen. Daher ist durch die Verwendung eines Meta-Klassifikatores keine Verbesserungen zu erwarten.

Kapitel 6

Fazit und Ausblick

Im folgenden Abschnitt werden die Ergebnisse der Arbeit zusammengefasst, sowie ein Fazit und ein Ausblick gegeben.

6.1 Fazit

Im ersten Teil der Auswertungen dieser Arbeit (Kapitel 4) wurde das Verhalten des in Bevendorff [2016] vorgestellten Authorship-Boosting-Verfahren zur Annäherung von Textrepräsenationen als Vorverarbeitungsschritt für Authorship-Verification-Verfahren auf dem in Bevendorff [2016] erstellten JB16 Korpus untersucht. Der Vergleich von Textpaaren unterschiedlicher Autoren und gleicher Autoren zeigte keine stärkere Annäherung bei Textpaaren gleicher Autoren. Infolgedessen führte AB zu keiner Verbesserung des Klassifikationsergebnisses. AB kann aufgrund dessen nicht als Vorverarbeitungsschritt eingesetzt werden.

Weiter wurde untersucht, ob mit AB eine Rückführung zur ursprünglichen Textrepräsentation erzielt werden kann, wenn zuvor eine Obfuskierung nach Bevendorff [2016] bzw. Bevendorff [2018] vorgenommen wurde. Dies ist möglich, wenn der gleiche Text sowohl zur Obfuskierung als auch zum AB verwendet wird. Eine große Übereinstimmung in der Wahl der Features während beider Phasen des Experiments zeigt, dass dieser Text auf dem JB16-Korpus als *Schlüssel* für das verwendete Obfuskierungsverfahren interpretiert werden kann. Dadurch wird das Verfahren angreifbar. Da für die untersuchten Obfuskierer auf dem JB16-Korpus angenommen werden kann, dass als Obfuskierungs-Text ein Text des gleichen Autors verwendet wird, kann auf dieser Weise der Autor des Textes ermittelt werden. Um zu untersuchen, ob dies auch für andere Texte des gleichen Autors gilt, wurde der JB16 Korpus um einen dritten Text des gleichen Autors erweitert. Die Experimente zeigen, dass diesbezüglich keine Generalisierung angenommen werden kann, da sich die

Textrepräsentationen nur sehr langsam annähern und ein ähnliches Verhalten auftritt, wenn ein Text eines anderen Autors zum AB verwendet wird. Die vorgestellten Experimente schlugen fehl, wenn die Textmenge zu gering waren (PAN15 [Stamatatos et al., 2015] und KW14 [Koppel und Winter, 2014]) oder sich Trainings- und Test-Korpus zu sehr unterschieden (PAN14 Novel Korpus [Stamatatos et al., 2014]).

Im zweiten Teil der Arbeit (Kapitel 5) wurde eine alternative Strategie des Unmasking (UM)-Algorithmus [Koppel und Schler, 2004] zur Auswahl und Modifizierung von Features entwickelt. Das Verfahren basiert auf den Grundlagen des AB-Algorithmus und wird in dieser Arbeit (Iterative) Feature Leveling genannt. Koppel und Schler [2004] stellen die These auf, dass sich Unterschiede in Texten des gleichen Autors in einer geringen Menge an Features zeigen. Daraus folgern sie, dass durch Entnahme dieser Features aus Texten des gleichen Autors eine signifikant schnellere Annäherung des Textpaars bewirkt werden kann, als wenn die Texte von einem unterschiedlichen Autor stammen. Mit dem Entwurf des neuen Verfahrens wurde überprüft, ob sich die These auf alternative Auswahl-Methoden der Features übertragen lässt.

Die entwickelte Strategie zeigt auf den Korpora KS04 [Koppel und Schler, 2004], PAN14 und JB16 eine zum klassischen (Iterative) Feature Removal (FR) Verfahren kompetitive Performanz. In beiden Varianten des Unmasking ist die Effektivität auf Korpora mit längeren Texten deutlich höher als bei kurzen Texten. In Abschnitt 5.3 wurden weitere Varianten bzw. Kombinationen der Unmasking-Strategien vorgeschlagen. Diesbezüglich wurde beispielhaft die Auswahl der zu modifizierenden Features nach FR mit einer Annäherung nach FL kombiniert. Die Ergebnisse des abweichenden Verfahrens sind vergleichbar mit den Ergebnissen von FL. Trotz Unterschieden in den einzelnen Klassifikationen lassen sich die Verfahren nicht zu einem *Meta-Klassifikator* kombinieren, sodass es zu keiner signifikanten Steigerung der Performanz kommt. Die Experimente mit FL haben gezeigt, dass sich Strategien entwickeln lassen, die alternativ zu FR eingesetzt werden können und eine Verallgemeinerung auf alternative Verfahren zulässig ist.

6.2 Ausblick

Im folgenden Abschnitt wird ein Ausblick auf kommende Forschungsfragen und Verbesserungsvorschläge gegeben. Es sollte überprüft werden, ob sich die Annahme bewahrheitet, dass über den Obfuskierungs-Text auf den Autor des obfuskierten Text geschlossen werden kann. Dazu muss, ähnlich zu den in Abschnitt 4.3 vorgestellten Experimenten, das Verhalten

bei Verwendung des Obfuskierungs-Text gegen das anderer Texte verglichen werden.

Der in Kapitel 4 eingesetzte AB-Algorithmus verwendet ein einfaches Verfahren zur Annäherung von zwei Verteilungen. In Abschnitt 4.4 wurde vorgeschlagen, das Verfahren bei Einsatz gegen AO derart zu abzuändern, dass die Auswahl der Features in Bezug auf ihren Einfluss auf die Summe der KLD entfällt. Weiter wird vorgeschlagen, das Verfahren um das Element einer heuristischen Suche, zu erweitern (vgl. Bevendorff [2018]). Anstelle einer Erhöhung der Jensen-Shannon Distanz JS_Δ bezüglich des zu obfuskierenden Dokuments wird eine Reduzierung der JS_Δ bezüglich des Dokuments, dessen Stil angenähert werden soll angestrebt. Das Verfahren kann neben dem Einsatz gegen AO auch mit direkten Ersetzungen im Text erfolgen und folglich als alternatives AO-Verfahren eingesetzt werden. Eine offene Forschungsfrage bleibt in Folge dessen, in wie weit mit AB Texte derart modifiziert werden können, sodass sowohl computergestützte AA-Ansätze, als auch Experten statt des tatsächlichen Autors einen anderen Autor zuweisen.

Ähnliche Verfahren können auch auf den in dieser Arbeit entwickelten FL-Algorithmus übertragen werden. Mit dem Auslassen einer Auswahl von Features stützt sich das Verfahren nicht mehr auf die von Koppel und Schler [2004] formulierte Annahme, dass die Unterschiede von Texten gleicher Autoren bei wenigen Features hoch, bei den restlichen Features aber sehr gering sind. Eine Variante mit einer Auswahl ohne Gewichtung der Features, welche eine Annäherung bezüglich des Mittelwerts vornimmt, wurde bereits beispielhaft untersucht. Die Ergebnisse des Meta-Klassifikators sind mit einer Accuracy von 0.940 und einem F1-Score von 0.700 mit den Verfahren FL und des klassischen FR vergleichbar. Weiter wird vorgeschlagen zu untersuchen, ob eine entgegengesetzte Operation, d. h. ein *Unähnlicher-Machen* ein vergleichbares Verhalten hervorbringen kann. Die Abnahme der Fähigkeit, die Chunk-Mengen der Klassen *gleicher Autor* und *unterschiedliche Autoren* zu trennen, sollte mit einer abweichenden Methode wie z. B. der Kosinusähnlichkeit berechnet werden, siehe Loose [2011], Stein et al. [2008], da die Anfangswerte der Kurven zumeist bei ≈ 1 liegen (vgl. Abbildung 3.5). Da eine Wertsteigerung vorgenommen wird, sind keine charakteristischen Kurvenvenverläufe zu erwarten. Eine offene Forschungsfrage bleibt, ob sich alternative Unmasking-Strategien entwickeln lassen, die ähnlich effektiv sind und von der oben formulierten Annahme abweichen. Des Weiteren bleibt offen, ob durch die Kombination von verschiedenen Unmasking-Strategien, die unterschiedliche Features einsetzen, innerhalb eines Ensemble-Klassifikators Verbesserungen erzielt werden können.

Literaturverzeichnis

Ahmed Abbasi und Hsinchun Chen. Writeprints: A Stylometric Approach to Identity-Level Identification and Similarity Detection in Cyberspace. *ACM Transactions on Information Systems*, 26(2):1–29, 2008. doi: 1344411.1344413.

Ahmed Abbasi und Hsinchun Chen. Applying Authorship Analysis to Extremist-Group Web Forum Messages. *IEEE Intelligent Systems*, 20(5):67–75, sep 2005. doi: 10.1109/MIS.2005.81.

Douglas Bagnall. Author Identification Using Multi-headed Recurrent Neural Networks. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*, 2015. URL `http://ceur-ws.org/Vol-1391/150-CR.pdf`.

Douglas Bagnall. Authorship Clustering using Multi-headed Recurrent Neural Networks. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016.*, pages 791–804, 2016. URL `http://ceur-ws.org/Vol-1609/16090791.pdf`.

Oleg Bakhteev und Andrey Khazov. Author Masking using Sequence-to-Sequence Models. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017.*, 2017. URL `http://ceur-ws.org/Vol-1866/paper_68.pdf`.

Janek Bevendorff. Authorship Verification and Obfuscation Using Distributional Features. Bachelor's thesis, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media, September 2016. URL `http://www.uni-weimar.de/medien/webis/teaching/theses/bevendorff_2016.pdf`.

Janek Bevendorff. Authorship Obfuscation Using Heuristic Search. Master's thesis, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media, June 2018. URL `https://www.uni-weimar.de/medien/webis/teaching/theses/bevendorff_2018.pdf`.

Bernhard E. Boser, Isabelle M. Guyon, und Vladimir N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the fifth annual workshop on Computational*

learning theory - COLT '92, pages 144–152, New York, New York, USA, 1992. ACM Press. doi: 10.1145/130385.130401.

Edward Gaylord Bourne. The Authorship of the Federalist. *The American Historical Review*, 2 (3):443–460, 1897. doi: 10.2307/1833399.

Daniel Castro-Castro, Reynier Ortega Bueno, und Rafael Muñoz. Author Masking by Sentence Transformation. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017.*, 2017. URL `http://ceur-ws.org/Vol-1866/paper_170.pdf`.

Chih-chung Chang und Chih-jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, 2011. doi: 10.1145/1961189.1961199.

Corinna Cortes und Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20: 273–297, 1995. doi: 10.1023/A:1022627411411.

Manuel Fernández Delgado, Eva Cernadas, Senén Barro, und Dinani Gomes Amorim. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems? *Journal of Machine Learning Research*, 15(1):3133–3181, 2014. URL `http://dl.acm.org/citation.cfm?id=2697065`.

Joachim Diederich, Jörg Kindermann, Edda Leopold, und Gerhard Paass. Authorship Attribution with Support Vector Machines. *Applied Intelligence*, 19:109–123, 2003. doi: 10.1023/A:102382490.

Hugo Jair Escalante, Manuel Montes, und Luis Villaseñor. Particle Swarm Model Selection for Authorship Verification. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5856 LNCS:563–570, 2009. doi: 10.1007/978-3-642-10268-4_66.

Rong-En Fan, Kai-Wai Chang, Cho-Jui Hsieh, Xiang-Rui Wang, und Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008. doi: 10.1145/1390681.1442794.

W. Nelson Francis und Henry Kučera. A Standard Corpus of Present-Day Edited American English, for use with Digital Computers (Brown). Technical report, Department of Linguistics, Brown University, Providence, Rhode Island, US, 1979.

Ingo Frommholz, Haider M. Al-Khateeb, Martin Potthast, Zinnar Ghasem, Mitul Shukla, und Emma Short. On Textual Analysis and Machine Learning for Cyberstalking Detection. *Datenbank-Spektrum*, 16(2):127–135, 2016. doi: 10.1007/s13222-016-0221-x.

Bent Fuglede und Flemming Topsoe. Jensen-Shannon divergence and Hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, page 31. IEEE, 2004. doi: 10.1109/ISIT.2004.1365067.

Tim Gollub, Benno Stein, Steven Burrows, und Dennis Hoppe. TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments. In *DEXA'12*, Wien, Österreich, 2012. IEEE. doi: 10.1109/DEXA.2012.55.

Matthias Hagen, Martin Potthast, und Benno Stein. Overview of the Author Obfuscation Task at PAN 2017: Safety Evaluation Revisited. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017.*, 2017. URL `http://ceur-ws.org/Vol-1866/invited_paper_4.pdf`.

Patrick Juola. Rowling and Galbraith: An Authorial Analysis. `http://languagelog.ldc.upenn.edu/nll/?p=5315`, 2013. Abgerufen am 9. Mai 2019.

Patrick Juola. The Rowling Case: A Proposed Standard Analytic Protocol for Authorship Questions. *Digital Scholarship in the Humanities*, 30:i100–i113, 2015. doi: 10.1093/llc/fqv040.

Patrick Juola und Efstathios Stamatatos. Overview of the Author Identification Task at PAN 2013. In *Working Notes for CLEF 2013 Conference , Valencia, Spain, September 23-26, 2013.*, 2013. URL `http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-JuolaEt2013.pdf`.

Gary Kacmarcik und Michael Gamon. Obfuscating Document Stylometry to Preserve Author Anonymity. *Proceedings of the COLING/ACL on Main conference poster sessions -*, pages 444–451, July 2006. doi: 10.3115/1273073.1273131.

Mike Kestemont, Kim Luyckx, Walter Daelemans, und Thomas Crombez. Cross-Genre Authorship Verification Using Unmasking. *English Studies*, 93(3):340–356, 2012. doi: 10.1080/0013838X.2012.668793.

Mike Kestemont, Justin Stover, Moshe Koppel, Folgert Karsdorp, und Walter Daelemans. Authenticating the writings of Julius Caesar. *Expert Systems with Applications*, 63:86–96, 2016. doi: 10.1016/j.eswa.2016.06.029.

Mahmoud Khonji und Youssef Iraqi. A Slightly-modified GI-based Author-verifier with Lots of Features (ASGALF). In *Working Notes for CLEF 2014 Conference, Sheffield,*

UK, September 15-18, 2014., pages 977–983, 2014. URL http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-KonijEt2014.pdf.

Foaad Khosmood und Robert Levinson. Toward Automated Stylistic Transformation of Natural Language Text. *Machine Translation*, pages 177–181, 2009.

Foaad Khosmood und Robert Levinson. Automatic Synonym and Phrase Replacement Show Promise for Style Transformation. *Proceedings - 9th International Conference on Machine Learning and Applications, ICMLA 2010*, pages 958–961, 2010. doi: 10.1109/ICMLA.2010.153.

Moshe Koppel und Jonathan Schler. Authorship verification as a one-class classification problem. In Carla E. Brodley, editor, *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4–8, 2004*, volume 69 of *ACM International Conference Proceeding Series*, page 62. ACM, 2004. doi: 10.1145/1015330.1015448.

Moshe Koppel und Jonathan Schler. Computational Methods in Authorship Attribution. *In vivo (Athens, Greece)*, 60(1):155–157, 2009. doi: 10.1002/asi.20961.

Moshe Koppel und Yaron Winter. Determining if Two Documents Are Written by the Same Author. *Journal of the Association for Information Science and Technology*, 65(1):178–187, jan 2014. doi: 10.1002/asi.22954.

Moshe Koppel, Jonathan Schler, und Shlomo Argamon. Authorship Attribution in the Wild. *Language Resources and Evaluation*, 45(1):83–94, 2011. doi: 10.1007/s10579-009-9111-2.

Solomon Kullback und Richard Arthur Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, mar 1951. doi: 10.1214/aoms/1177729694.

Maarten Lambers und Cor J. Veenman. Forensic Authorship Attribution Using Compression Distances to Prototypes. In Zeno J. M. H. Geradts, Katrin Y. Franke, und Cor J. Veenman, editors, *Computational Forensics*, pages 13–24. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-03521-0.

David D. Lewis. Reuters-21578 Text Categorization Test Collection, 2004. URL http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html. Abgerufen am 9. Mai 2019.

Fabian Loose. Paarweise Autorenschaftsverifikation von kurzen Texten. Master's thesis, Bauhaus-Universität Weimar, Fakultät Medien, Computer Science and Media, May 2011. URL http://www.uni-weimar.de/medien/webis/teaching/theses/loose_2011.pdf.

Tsvetomila Mihaylova, Georgi Karadzhov, Preslav Nakov, Yasen Kiprov, Georgi Georgiev, und Ivan Koychev. SU@ PAN'2016: Author Obfuscation—Notebook for PAN at CLEF 2016. In *Conference and Labs of the Evaluation Forum*, 2016.

Frederick Mosteller und David Wallace. *Inference and Disputed Authorship: The Federalist*. Addison-Wesley, 1964.

Raymond Mougeon, Edouard Beniak, und Daniel Valois. A sociolinguistic study of language contact, shift, and change. *Linguistics*, 23(3):455–488, 1985. doi: 10.1515/ling.1985.23.3.455.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, und Dawn Song. On the Feasibility of Internet-Scale Author Identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314. IEEE, may 2012. doi: 10.1109/SP.2012.46.

Andrew P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. doi: 10.1016/S0031-3203(96)00142-2.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, und Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.*, 12:2825–2830, November 2011. ISSN 1532-4435.

Anselmo Peñas und Alvaro Rodrigo. A Simple Measure to Assess Non-response. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1415–1424, Portland, Oregon, USA, 2011. Association for Computational Linguistics.

Nektaria Potha und Efstathios Stamatatos. A Profile-Based Method for Authorship Verification. In *Artificial Intelligence: Methods and Applications - 8th Hellenic Conference on AI, SETN 2014, Ioannina, Greece, May 15-17, 2014. Proceedings*, pages 313–326, 2014. doi: 10.1007/978-3-319-07064-3_25.

Nektaria Potha und Efstathios Stamatatos. Intrinsic Author Verification Using Topic Modeling. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence - SETN '18*, pages 1–7, New York, New York, USA, 2018. ACM Press. doi: 10.1145/3200947.3201013.

Martin Potthast, Matthias Hagen, und Benno Stein. Author Obfuscation: Attacking the State of the Art in Authorship Verification. In *Working Notes of CLEF 2016 - Conference and Labs*

of the Evaluation forum, Évora, Portugal, 5-8 September, 2016., pages 716–749, 2016. URL http://ceur-ws.org/Vol-1609/16090716.pdf.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, und Benno Stein. A Stylometric Inquiry into Hyperpartisan and Fake News. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 231–240, 2018a. URL https://aclanthology.info/papers/P18-1022/p18-1022.

Martin Potthast, Felix Schremmer, Matthias Hagen, und Benno Stein. Overview of the Author Obfuscation Task at PAN 2018: A New Approach to Measuring Safety. In *Working Notes of CLEF 2018 - Conference and Labs of the Evaluation Forum, Avignon, France, September 10-14, 2018.*, 2018b. URL http://ceur-ws.org/Vol-2125/invited_paper_16.pdf.

Josyula R. Rao und Pankaj Rohatgi. Can Pseudonymity Really Guarantee Privacy? In *9th USENIX Security Symposium, Denver, Colorado, USA, August 14-17, 2000*, 2000. URL https://www.usenix.org/conference/9th-usenix-security-symposium/can-pseudonymity-really-guarantee-privacy.

Anderson Rocha, Walter J. Scheirer, Christopher W. Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne R.B. Carvalho, und Efstathios Stamatatos. Authorship Attribution for Social Media Forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33, 2017. doi: 10.1109/TIFS.2016.2603960.

Conrad Sanderson und Simon Guenter. Short text Authorship Attribution via Sequence Kernels, Markov Chains and Author Unmasking. In *Conference on Empirical Methods in Natural Language Processing*, pages 482–491, 2006. doi: 10.3115/1610075.1610142.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, und James Pennebaker. Effects of Age and Gender on Blogging. *AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*, 2006.

Shachar Seidman. Authorship Verification Using the Impostors Method. *Working Notes for CLEF 2013 Conference*, pages 1–4, 2013. URL http://ceur-ws.org/Vol-1179/CLEF2013wn-PAN-Seidman2013.pdf.

Efstathios Stamatatos. Author identification: Using Text Sampling to Handle the Class Imbalance Problem. *Information Processing & Management*, 44(2):790–799, mar 2008. doi: 10.1016/j.ipm.2007.05.012.

Efstathios Stamatatos. Authorship Verification : A Review of Recent Advances. *Research in Computing Science*, 123:9–25, 2016.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Benno Stein, Martin Potthast, Patrick Juola, Miguel A. Sánchez-Pérez, und Alberto Barrón-Cedeño. Overview of the Author Identification Task at PAN 2014. In *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15-18, 2014.*, pages 877–897, 2014. URL `http://ceur-ws.org/Vol-1180/CLEF2014wn-Pan-StamatosEt2014.pdf`.

Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López-López, Martin Potthast, und Benno Stein. Overview of the Author Identification Task at PAN 2015. In *Working Notes of CLEF 2015 - Conference and Labs of the Evaluation forum, Toulouse, France, September 8-11, 2015.*, 2015. URL `http://ceur-ws.org/Vol-1391/inv-pap3-CR.pdf`.

Benno Stein, Nedim Lipka, und Sven Meyer Zu Eissen. Meta Analysis within Authorship Verification. *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, pages 34–39, 2008. doi: 10.1109/DEXA.2008.20.

Vladimir Naumovich Vapnik und Alexey Yakovlevich Chervonenkis. Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control*, pages 774–780, 1963.

Ying Zhao und Justin Zobel. Searching With Style: Authorship Attribution in Classic Literature. In *Australasian conference on Computer science*, volume 62, pages 59–68, 2007. ISBN 1-920-68243-0.

Ying Zhao, Justin Zobel, und Phil Vines. Using Relative Entropy for Authorship Attribution. In *Information Retrieval Technology, Third Asia Information Retrieval Symposium, AIRS 2006, Singapore, October 16-18, 2006, Proceedings*, pages 92–105, 2006. doi: 10.1007/11880592_8.

Anhang A

Graphiken

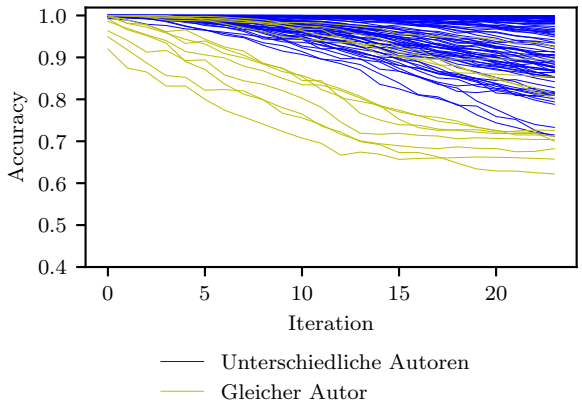
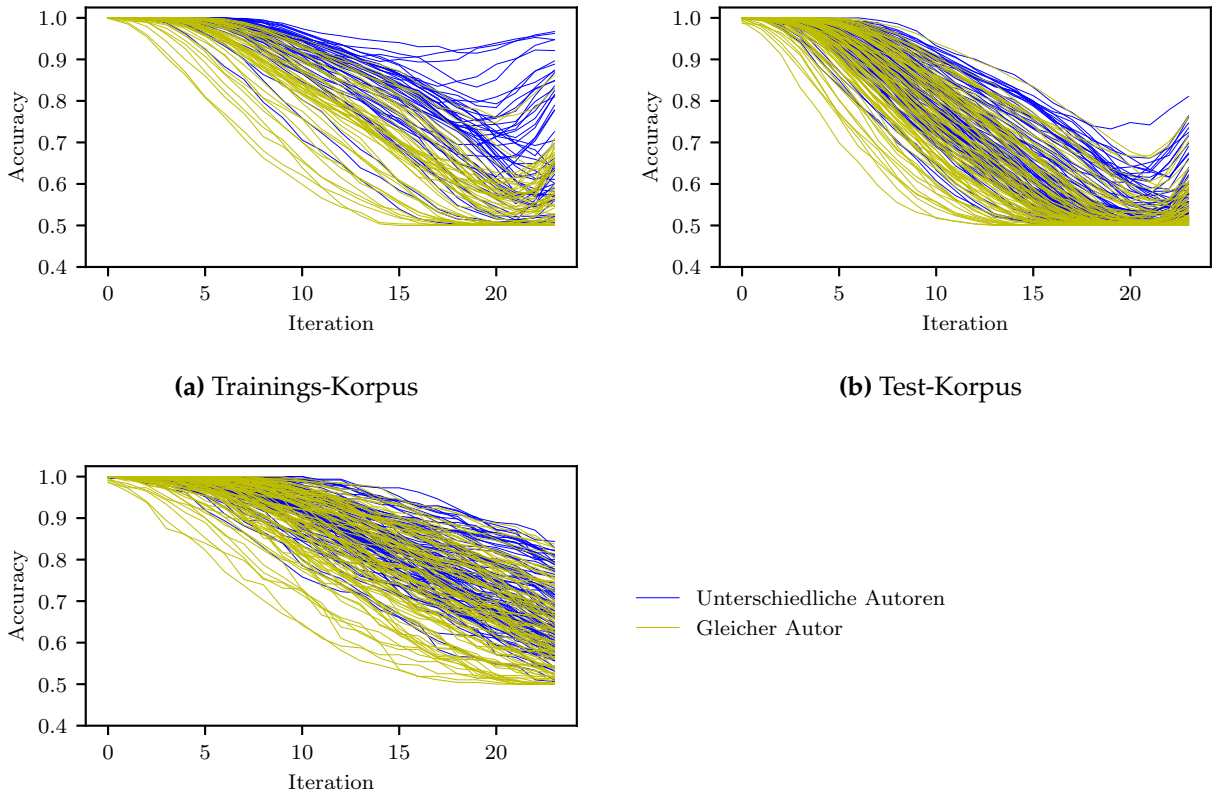


Abbildung A.1: FL absolute Frequenzen mit Konstanten Änderungen $\Delta_{feature}$ und Random Sampler auf dem KS04-Korpus. Mittel über fünf Iterationen mit je 500 und 700 Wörtern pro Chunk, $m = 10$ Features je Iteration, monotonisiert, Mittel über 5 Iterationen. $\Delta_{feature} = 10$ Features je Iteration. Accuracy 0.950, F1 Score 0.760.

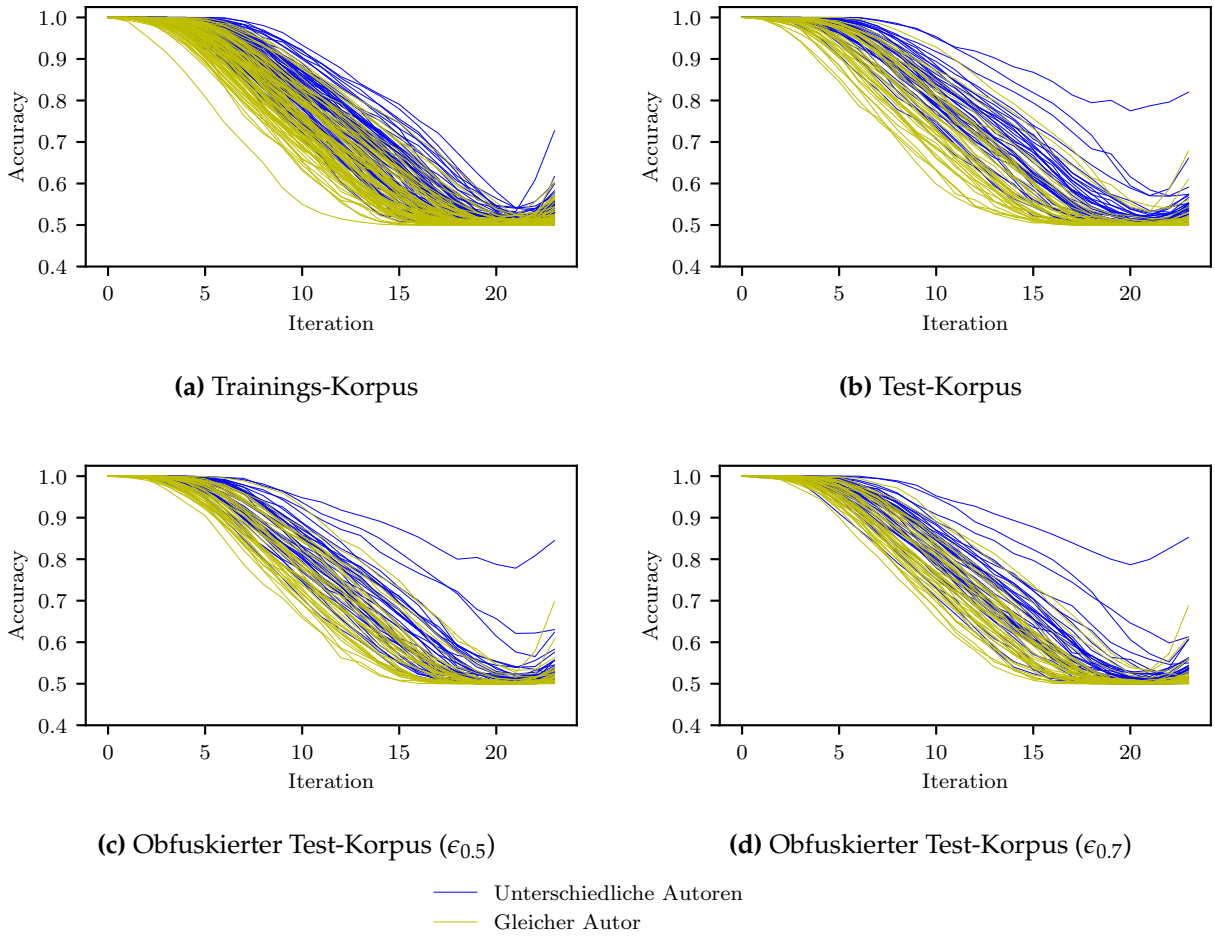


(a) Trainings-Korpus

(b) Test-Korpus

(c) Obfuskierter Test-Korpus ($\epsilon_{0.5}$)

Abbildung A.2: FR-UM Ergebnisse auf dem PAN14-Korpus. Mittel über zehn Iterationen mit je 400 und 600 Wörtern pro Chunk, Entnahme von zehn Features pro Durchgang.**(a)** Trainings-Korpus: Accuracy 0.760 **(b)** Test-Korpus: Accuracy 0.545, F1 0.671 **(c)** Obfuskierter Test-Korpus ($\epsilon_{0.5}$) Accuracy 0.530 F1 0.652.



(a) Trainings-Korpus

(b) Test-Korpus

(c) Obfuskierter Test-Korpus ($\epsilon_{0.5}$)

(d) Obfuskierter Test-Korpus ($\epsilon_{0.7}$)

Abbildung A.3: FR-UM Ergebnisse auf dem JB16-Korpus. Mittel über zehn Iterationen mit 500 und 600 Wörter pro Chunk, Entnahme von zehn Features pro Durchgang **(b)** Test-Set: Accuracy 0.738, F1 0.712. **(c)** Obfuskierter Test-Korpus: ($\epsilon_{0.5}$) Accuracy 0.650, F1 0.611., **(d)** Obfuskierter Test-Korpus ($\epsilon_{0.7}$): Accuracy 0.650, F1 0.632.

Anhang B

Tabellen

Tabelle B.1: JB16-Test Korpus mit Boost Texten des selben Autors. Aus jedem Buch wurde ein zufälliger Text mit 3800 Wörtern entnommen. Die Textlänge entspricht damit dem Durchschnitt der Textlänge des JB16-Korpus. Aus Platzgründen wurden ggf. Buchtitel gekürzt.

Autor	Bekannter Text	Unbekannter Text	Boost Text
L. Frank Belknap	The Mississippi Saucer	The Sky Trap	The Man from Time
R. Rick	Code Three	A Filbert Is a Nut	The Thirst Quenchers
L. Jack	A Son Of The Sun	Lost Face	Valley of the Moon
D. Lester	Police Your Planet	Victory	Let em Eat Space
F. George Manville	Cormorant Crag	Bunyip Land	A Boy at Sea
W. Stanley	The Wolf's Long Howl	The Story of Ab	A Man and a Woman
N. Kris	General Max Shorter	New Apples in the Garden	EARTH ALERT!
E. George Allan	Darkness and Dawn	The Air Trust	The Flying Legion
R. Francis	The Boy With the U.S. Miners	The Boy With the U. S. Fisheries	Weather Men
H. Edmond	The Stars, My Brothers	The Sargasso of Space	The City at World's End
L. Jack	The Iron Heel	The Scarlet Plague	Tales of the Fish Patrol
D. Arthur Conan	The Dealings of Captain Sharkey	The Adventures of Gerard	My Friend The Murderer
M. James	George Loves Gistla	Planet of Dreams	Confidence Game
M. Kirk	The Copper Princess:	Under the Great Bear	Cap and Caboose
T. Louis	The Captain of the Kansas	The Silent Barrier	Cynthia's Chauffeur
N. Alan Edward	Star Surgeon	My Friend Bobby	Martyr
S. Edward	The Rover Boys at School	The Boy Land Boomer	On the Trail of Pontiac
P. Randall	Prisoners of Chance	Keith of the Border	My Lady of the North
H. Bracebridge	Jack Harkaway's Boy	Jack Harkaway and His Son's Escape	The Slave of the Mine
M. Talbot	The Ivory Trail	The Eye of Zeitoon	Affair in Araby
B. Louis	Five-Head Creek	Martin Of Nitendi	Tom Gerrard
A. Roger D.	Control Group	To Remember Charlie By	Pet Farm
G. Tom	And Devious the Line of Duty	Cry from a Far Planet	Space Prison
G. George Chetwynd	A Honeymoon in Space	The Angel of the Revolution	The Missionary
D. Dave	Tree, Spare that Woodman	Waste Not, Want	Uniform of a Man
B. Algis	Citadel	The Barbarians	Wall of Crystal, Eye of Night
F. Edna	Roast Beef, Medium	Personality Plus	Fanny Herself
W. Robert Moore	Be It Ever Thus	Thompson's Cat	The Lost Warship
H. Henry Rider	Eric Brighteyes	Fair Margaret	Cleapatra
F. Homer Eon	The Emancipatrix	The Blind Spot	The Lord of Death and the Queen of Life
L. William	The Great White Queen	The Mystery of the Green Ray	Hushed Up
S. Matthew Phipps	The Purple Cloud	The Lord of the Sea	Prince Zaleski
S. Joseph	Divinity	Runaway	Beside Manner
M. George	The Adventures of Harry Richmond (V4)	The Adventures of Harry Richmond (V8)	Rhoda Fleming (V2)
N. Edith	Oswald Bastable and Others	The Wouldbegoods	The Enchanted Castle
S. Frank Richard	The Great Stone of Sardis	The Great War Syndicate	Rudder Grange
K. William Henry Giles	Afar in the Forest	Dick Cheveley	The Three Lieutenants
H. Henry	Walls of Acid	We're Friends, Now	The Beginning
B. Robert Michael	Black Ivory	Away in the Wilderness	Post Haste
H. George Alfred	In Freedom's Cause	A Jacobite Exile	The Reign Of Terror

Tabelle B.2: JB16-Test Korpus mit Boost Texten unterschiedlicher Autoren. Die Textlänge entspricht damit dem Durchschnitt der Textlänge des JB16-Korpus. Aus jedem Buch wurde ein zufälliger Text mit 3800 Wörtern entnommen. Aus Platzgründen wurden ggf. Buchtitel gekürzt.

Autor	Bekannter Text	Unbekannter Text	Autor	Boost Text
L. Frank Belknap	The Mississippi Saucer	The Sky Trap	B. Leigh Douglass	A World is Born
R. Rick	Code Three	A Filbert Is a Nut	J. Conrad	Typhoon
L. Jack	A Son Of The Sun	Lost Face	H. Thompson	Spawn of the Comet
D. Lester	Police Your Planet	Victory	L. Floyd	Second Landing
F. George Manville	Cormorant Crag	Bunyip Land	A. K. Lang	Blind Man's Lantern
W. Stanley	The Wolf's Long Howl	The Story of Ab	R. Jefferies	After London Or, Wild England
N. Kris	General Max Shorter	New Apples in the Garden	F. F. Moore	The Devil's Admiral
E. George Allan	Darkness and Dawn	The Air Trust	M. Corelli	The Secret Power
R. Francis	The Boy With the U.S. Miners	The Boy With the U. S. Fisheries	W. H. Hodgson	The House on the Borderland
H. Edmond	The Stars, My Brothers	The Sargasso of Space	G. P. Serviss	Edison's Conquest of Mars
L. Jack	The Iron Heel	The Scarlet Plague	S. Rohmer	The Return of Dr. Fu-Manchu
D. Arthur Conan	The Dealings of Captain Sharkey	The Adventures of Gerard	A. E. Dingle	Gold Out of Celebes
M. James	George Loves Gistla	Planet of Dreams	D. William	The Brassbounder
M. Kirk	The Copper Princess:	Under the Great Bear	R. F. Jones	The Great Gray Plague
T. Louis	The Captain of the Kansas	The Silent Barrier	J. Franklin	Pandemic
N. Alan Edward	Star Surgeon	My Friend Bobby	C. Goddard	The Perils of Pauline
S. Edward	The Rover Boys at School	The Boy Land Boomer	W. West	The End of Time
P. Randall	Prisoners of Chance	Keith of the Border	A. C. Doyle	Danger! and Other Stories
H. Bracebridge	Jack Harkaway's Boy (...)	Jack Harkaway and His Son's Escape	W. M. Miller	The Hoofer
M. Talbot	The Ivory Trail	The Eye of Zeitoon	J. Archibald	Operation Earthworm
B. Louis	Five-Head Creek	Martin Of Nitendi	A. E. Dingle	Gold Out of Celebes
A. Roger D.	Control Group	To Remember Charlie By	B. Mitford	The Ruby Sword
G. Tom	And Devious the Line of Duty	Cry from a Far Planet	E. B. Haverfield	The Best Made Plans
G. George Chetwynd	A Honeymoon in Space	The Angel of the Revolution	E. L. Haverfield	Queensland Cousins
D. Dave	Tree, Spare that Woodman	Waste Not, Want	R. Shirley	Disturbing Sun
B. Algis	Citadel	The Barbarians	H. Bindloss	Carmen's Messenger
F. Edna	Roast Beef, Medium	Personality Plus	R. Garret	The Judas Valley
W. Robert Moore	Be It Ever Thus	Thompson's Cat	P. Anderson	Security
H. Henry Rider	Eric Brighteyes	Fair Margaret	W. M. Graydon	Jolly Sally Pendleton
F. Homer Eon	The Emancipatrix	The Blind Spot	H. Bates	Under Arctic Ice
L. William	The Great White Queen	The Mystery of the Green Ray	J. Harmon	The Last Place on Earth
S. Matthew Phipps	The Purple Cloud	The Lord of the Sea	E. Orczy	El Dorado
S. Joseph	Divinity	Runaway	M. Moldeven	The Universe - or Nothing
M. George	The Adventures of Harry Richmond (V4)	The Adventures of Harry Richmond (V8)	H. King	Fred Fearnot's New Ranch
N. Edith	Oswald Bastable and Others	The Wouldbegoods	H. MacGrath	The Grey Cloak
S. Frank Richard	The Great Stone of Sardis	The Great War Syndicate	V. A. Endersby	Disowned
K. William Henry Giles	Afar in the Forest	Dick Cheveley	A. Stringer	The Man Who Couldn't Sleep
H. Henry	Walls of Acid	We're Friends, Now	A. Hope	The Prisoner of Zenda
B. Robert Michael	Black Ivory	Away in the Wilderness	H. S. Merriman	The Last Hope
H. George Alfred	In Freedom's Cause	A Jacobite Exile	M. Nicoll	The Blue Germ

Anhang C

Listings

Algorithm 6 Authorship-Obfuscation (Extension) nach Bevendorff [2016]. Seien p und q die Vektoren mit den Häufigkeiten der Features in den Dokumenten d_k und d_u und max_iterations die Anzahl der Features, die modifiziert werden sollen.

Input: vec p, vec q, int max_iterations
Output: vec q

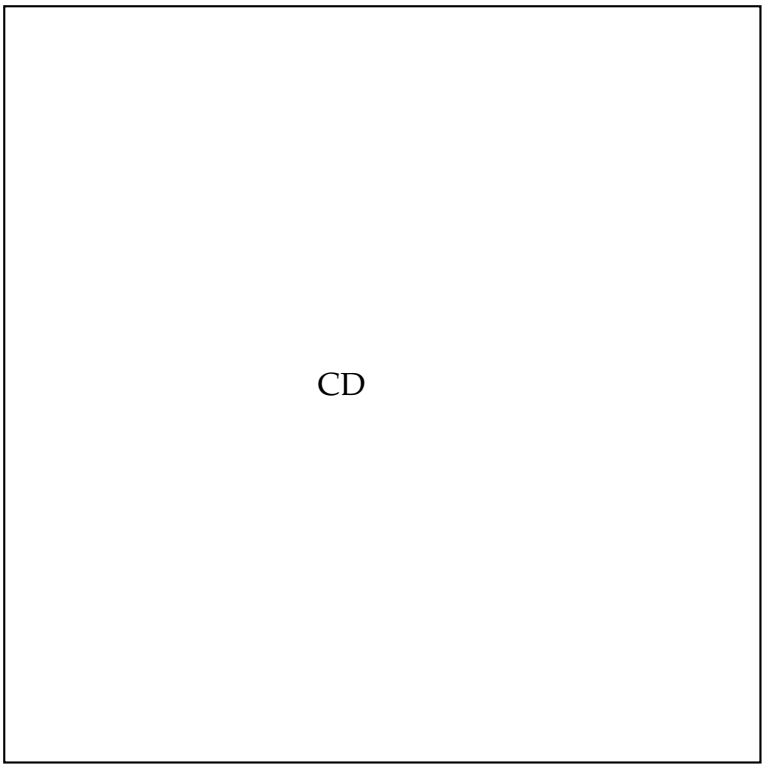
```
1:  counter := 0
2:  vec ps := sorted_desc(p, key := func(n): p[n])
3:  while counter < max_iterations do
4:      for each feature in ps do
5:          if q[feature] = 0 then
6:              q[feature]:= 1
7:              break
8:          end if
9:      end for
10:     counter++
11: end while
```

Anhang D

Inhalt der CD

Der Masterarbeit sind eine elektronische Fassung der Arbeit, der LaTeX-Quellcode sowie lauffähige Versionen der implementierten Algorithmen als CD beigefügt. Die Verzeichnis-Struktur gibt einen Überblick über den Inhalt der CD.

```
/
├── Arbeit ..... Enthält LaTeX Files und Abbildungen.
├── Code ..... Enthält den Quellcode sowie Bibliotheken.
├── Experimente ..... Enthält Daten der Auswertung.
├── Korpora ..... Enthält erstellte Korpora.
└── MA_2019_Wenzel,Tobias_Titel.pdf ..... Masterarbeit als PDF-Dokument.
```



```
/
├── Arbeit ..... Enthält LaTeX Files und Abbildungen.
├── Code ..... Enthält den Quellcode.
│   ├── authorship_unmasking_py
│   │   └── [...] ..... Enthält UM-Quellcode aus Bevendorff [2016] und FL Erweiterungen.
│   └── wstud-thesis-wenzel-master
│       └── [...] ..... Enthält Quellcode der AB-Experimente sowie Hilfsfunktionen.
├── Experimente ..... Enthält Daten der Auswertung.
│   ├── Sharpening ..... Enthält Auswertungen des Sharpening Experiments.
│   ├── Gleicher Text ..... Enthält AB-Auswertungen mit Obfuskierungs-Text.
│   ├── Gleicher Autor ..... Enthält AB-Auswertungen mit Text des gleichen Autors.
│   ├── Unterschiedlicher Autor ..... Enthält AB-Auswertungen mit anderem Autor.
│   └── Unmasking ..... Enthält UM-Kurvenverläufe der Experimente.
├── Korpora ..... Enthält erstellte Korpora..
│   ├── KW14 ..... Blogger-Korpus nach Koppel und Winter [2014].
│   ├── JB16 BO 1 ..... JB16 Boosting Korpus mit Texten des gleichen Autors.
│   └── JB16 BO 2 ..... JB16 Boosting Korpus mit Texten eines anderen Autors.
└── MA_2019_Wenzel,Tobias_Authorship Boosting: Ein Verfahren zur Bekämpfung der
    Autorschaftsverschleierung.pdf ..... Masterarbeit als PDF-Dokument.
```