Universität Leipzig
Institut für Informatik
Studiengang Informatik, B.Sc.

Das Novelupdates Korpus – Erzeugung und Evaluation eines Textkorpus mit Hilfe von maschinellem Lernen –

Bachelorarbeit

Henrik Bininda

Gutachter: Jun.-Prof. Dr. Martin Potthast
Mentor: Erik Körner

Datum der Abgabe: 7. September 2022

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Leipzig, 7. September 2022

.....
Henrik Bininda

Zusammenfassung

Anwendungen der Linguistischen Informatik, wie bspw. Autorenschaftsanalysen, benötigen große Datenmengen an Text – sogenannte Korpora – um entwickelt und erprobt zu werden. Idealerweise stammen diese Texte aus verschiedenen Domänen und decken unterschiedliche Thematiken ab, um eine breite Anwendbarkeit der Verfahren sicherzustellen.

Mit dieser Arbeit wird das „Novelupdates-Korpus“ präsentiert. Dabei handelt es sich um ein Korpus aus 232.398 Einzeltexten von ins Englische übersetzten asiatischen (Web-)Novels. Die Texte sind zum einen thematisch sehr vielschichtig, da die übersetzten Originalerzählungen von verschiedenen Autoren zeitgenössischer Literatur aus u. a. China, Japan und Südkorea stammen. Zum anderen stehen verschiedene Texte miteinander in inhaltlichem Zusammenhang, sei es, weil sie verschiedene Kapitel derselben Novel sind oder es sich sogar um dasselbe Kapitel, jedoch doppelt und unabhängig voneinander übersetzt, handelt.

Neben dem Hauptteil, welcher sich vor allem auf die Erzeugung des Korpus vom Crawl bis zum Säubern der Texte fokussiert, betrachtet diese Arbeit das fertige Korpus unter dem Aspekt der weiteren Verwendungsmöglichkeiten für Autorenschaftsanalysen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung und Idee

In den letzten Jahren haben Anwendungen des maschinellen Lernens bzw. „Machine Learning“ (ML) immer mehr an Bedeutung gewonnen. Ein Aspekt des maschinellen Lernens ist das sogenannte „Supervised Learning“, also „überwachtes Lernen“. Dabei werden Algorithmen anhand einer bereits klassifizierten Datensammlung so trainiert, dass neue, bisher unbekannte Objekte entsprechend des bisher gesammelten Wissens korrekt klassifiziert werden können.

Ein Anwendungsbereich von Supervised-Learning-Verfahren sind sogenannte Autorenschaftsanalysen. Dabei werden in verschiedenen Formen Texte automatisch ihrem jeweiligen – bis dahin unbekannten – Autor zugeordnet. Jedes System zur Autorenschaftsanalyse muss auf verschiedenen Textgattungen zu verschiedensten Thematiken robuste Ergebnisse liefern, um als zuverlässig zu gelten [1]. Grundlage für die Forschung in diesem Bereich sind also entsprechende Textsammlungen, auch Textkorpora genannt. Als Wissensbasis für weitere Forschung und als Trainingsdatensatz für Machine-Learning-Verfahren, wie sie für die Autorenschaftsanalyse eingesetzt werden, müssen die Korpora in geordneter Form mit den notwendigen Metadaten – insbesondere dem jeweiligen Autor – vorliegen. Oft stehen hierfür nur künstliche Kompositionen von Texten zur Verfügung, wobei Texte verschiedener Autoren und Domänen (bspw. Nachrichtenmeldungen zu unterschiedlichen Themen) zusammengefügt worden sind. Diese mögen grundlegende Autorenschaftsanalysen ermöglichen, bergen jedoch das Risiko möglicher Verzerrungen aufgrund der stark abweichenden Inhalte. Zudem würden Kombinationen dieser Texte, wie sie z. B. für Style Change Detection notwendig sind, oft für einen Leser wenig Sinn ergeben. Zielführender wäre es daher, Texte verschiedener Autoren in einem inhaltlichen Zusammenhang bzw. Texte eines Autors über verschiedene Themenbereiche hinweg zusammenzutragen. Damit würde man eine hinreichende Herausforderung zur Überprüfung der Robustheit verschiedener Verfahren zur Autorenschaftsanalyse schaffen.

Mit dieser Arbeit wird ein Textkorpus aus dem Bereich der (Fan-)Übersetzung von asiatischen (Web-)Novels in die englische Sprache vorgelegt. Unter einer Novel ist dabei eine Art Kurzgeschichte oder längere fortlaufende Erzählung in Einzelkapiteln zu verstehen. Das Korpus umfasst 232.398 Einzeltexte von mehr als 2.500 Übersetzerinnen und Übersetzern. An mehr als 2.100 der enthaltenen Novels haben zudem mindestens zwei verschiedene Übersetzungsgruppen partizipiert.[1] Damit weist das Korpus eine natürlichsprachliche Qualität der Texte verschiedener Übersetzerinnen und Übersetzer innerhalb eines fortlaufenden, inhaltlichen Zusammenhangs auf (eine Geschichte, Kapitel ggf. von unterschiedlichen Gruppen bearbeitet). Zudem haben Übersetzungsgruppen auch an unterschiedlichen Novels verschiedener Gattungen gearbeitet.

Ausgangspunkt der Erzeugung des Korpus waren dabei ca. 320.000 HTML-Dateien, welche durch Crawling der Website novelupdates.com [2] gewonnen wurden. Hieraus galt es im Rahmen dieser Arbeit die einzelnen Texte für das Korpus zu extrahieren und zu säubern. Das Verfahren zur Erzeugung des Korpus sollte möglichst einfach mithilfe des im Rahmen dieser Arbeit zur Verfügung gestellten Materials (Erläuterungen, Git-Repositorium) auf andere Datenmengen übertragbar und schnell replizierbar sein.

Auf dem Korpus wurden zudem ausgewählte Ansätze zur Autorenschaftsanalyse getestet, um einen Anwendungsfall zu demonstrieren.

Im Ergebnis präsentiert diese Arbeit ein umfangreiches Textkorpus als Material für weitere Forschung, insbesondere im Bereich der Autorenschaftsanalysen. Beispielsweise könnten die Texte als Datensatz in einer der PAN-Konferenzen [3] der Webis-Gruppe [4] verwendet werden. Zudem dürfte das zur Erzeugung verwendete Verfahren mit relativ wenig Aufwand replizierbar sein, um weitere Korpora nach eigenem Bedarf zu erzeugen.

[1] Tatsächlich haben an allen ausgewählten Novels mehrere Übersetzungsgruppen mitgewirkt, jedoch spiegelt sich dies nach verschiedenen Säuberungsschritten nicht mehr im finalen Korpus wieder.

2. Theoretischer Hintergrund

2.1 Korpora

„Ein Textkorpus [...] ist eine Sammlung von schriftlichen Texten oder textlich aufgezeichneten mündlichen Äußerungen einer bestimmten Sprache oder Textgattung“ [5]. Der Begriff leitet sich vom lateinischen Wort *corpus* „Körper“ ab. Die Mehrzahl von „Korpus“ bezeichnet man als „Korpora“.

Korpora sind die wichtigste Wissensbasis für die Fachgebiete der Computerlinguistik und der natürlichen Sprachverarbeitung bzw. „Natural Language Processing“ (NLP). Jedoch finden sie auch in zahlreichen anderen Fachgebieten Anwendung, wie bspw. themenspezifische Korpora aus der Medizin oder den Rechtswissenschaften. Diese Korpora liegen heutzutage überwiegend in elektronischer Form vor, da eine Verarbeitung und Analyse dieser gewaltigen Datensammlungen ohne Zuhilfenahme informationstechnischer Verfahren wohl kaum realisierbar wäre.

Mit der voranschreitenden Digitalisierung rücken zudem immer weitere Anwendungen aus dem Bereich der künstlichen Intelligenz und dabei insbesondere des maschinellen Lernens in den Mittelpunkt der öffentlichen Wahrnehmung. Hierfür bilden Korpora quasi einen Goldstandard, sind sie doch Wissensbasis sowie Trainings- und Testmengen gleichermaßen für diese Verfahren.

Nachdem die Relevanz von Korpora aufgezeigt wurde, stellt sich die Frage, wie man die jeweiligen Korpora für einen bestimmten Zweck gewinnen kann. Zumeist werden sich bereits vorhandene Sammlungen finden lassen. So existieren bereits zahlreiche frei verfügbare Korpora wie z. B. das Brown-Korpus [6] oder das Gutenberg-Korpus [7] welche wiederum über verschiedene NLP-Bibliotheken wie bspw. NLTK [8] abrufbar sind.

Werden Korpora für spezifischere Zwecke benötigt, besteht die Möglichkeit, ein eigenes Korpus zusammenzustellen. Hierfür bietet sich das Crawling

verschiedener im Web verfügbarer Dokumente/Seiten an. Mit einer Weiterverarbeitung und Extraktion von Informationen aus den hierbei erhaltenen HTML-Dateien können textuelle Informationen gewonnen werden, welche in Ihrer Gesamtheit dann das Korpus bilden.

2.2 Novelupdates und die asiatische Webnovel-Szene

Novelupdates [2] ist ein community-betriebenes Verzeichnis von englischsprachigen Übersetzungen asiatischer Novels. Dabei kann es sich sowohl um Fan-Übersetzungen als auch offizielle Übersetzungen der Novels handeln.

Wie bereits eingangs erwähnt, handelt es sich bei diesen Novels um Kurzgeschichten oder fortlaufende längere Erzählungen. Dabei werden u. a. folgende Typen von Novels unterschieden, wobei die Übergänge teilweise fließend sind [9]:

Webnovel Wie der Name sagt, handelt es sich hierbei um Novels die lediglich Online veröffentlicht werden. Dies kann professionell erfolgen, oft aber auch als bloßes Hobby „nebenbei“.

Lightnovel Diese Novels können auch Online vorliegen, haben für gewöhnlich jedoch auch physische Kopien. Oft handelt es sich hierbei um etwas professionellere Arbeiten, insofern Publisher dahinterstehen und ein Lektorat oder Ähnliches durchlaufen wurde. Inhaltlich bestehen keine Unterschiede zur Webnovel.

Published-Novel Hierbei handelt es sich inhaltlich auch um Web- bzw. Lightnovels. Diese Novels wurden jedoch durch einen Publisher herausgegeben.

Die Novels stammen überwiegend aus den Ländern China, Japan und Südkorea.

Novelupdates enthält Metadaten zu den Novels sowie Listen mit Links zu den einzelnen Kapiteln einer Novel auf externen Seiten (bspw. eigene Websites, Blogs, etc.), wo der eigentliche übersetzte Inhalt zu finden ist.

Die Novels sind in der Regel in einzelne Kapitel aufgeteilt und werden kapitelweise übersetzt. Dabei können einzelne Kapitel wiederum in Teil 1 bis

Teil n aufgeteilt werden, wenn sie zu lang sind. Die Übersetzungen werden durch Einzelpersonen oder Gruppen durchgeführt. Die Gruppen wiederum bestehen regelmäßig aus Übersetzern, Editoren und Korrekturlesern.

Synonyme Verwendung der Begriffe
Im Rahmen dieser Arbeit werden die Begriffe „Übersetzer“, „Gruppen“ und „Autoren“ synonym verwendet. Daraus folgt, dass wenn vom „Autor“ gesprochen wird, nicht der tatsächliche Autor des asiatischen Originaltextes der Novel gemeint ist, sondern der Übersetzer / die Gruppe entsprechend Novelupdates. Dies bedeutet weiterhin, dass es sich bei „dem Autor“ sowohl um eine Einzelperson, als auch eine Personengruppe handeln kann.

Es existieren Gruppen, welche eine große Anzahl an Übersetzern beschäftigen und sich damit eine gewisse Vormachtstellung für bestimmte Genres oder genreübergreifend erarbeiten konnten. Zwischen den Gruppen herrscht ein ausgesprochener Wettbewerb bzgl. Qualität und Geschwindigkeit der Übersetzungen und damit einhergehend der Gewinnung von neuer Leserschaft (Traffic auf deren Websites). Gleichwohl werden im Rahmen dieses Konkurrenzkampfes für gewöhnlich ungeschriebene Regeln beachtet, etwa dass keine andere Gruppe ein bereits laufendes Übersetzungsprojekt einer anderen Gruppe aufgreift und diesbezüglich in Wettbewerb tritt.

Die Übersetzer finanzieren sich über Werbeanzeigen auf ihren Websites/Blogs, Spenden und kostenpflichtige Inhalte, wie Bonuskapitel oder frühere/ schnellere Veröffentlichungen gegen Bezahlung. Ebenfalls sind komplett kostenpflichtige Inhalte möglich, welche hinter sogenannten Paywalls verborgen werden.

Feedback erhalten die Übersetzer über die Kommentarsektionen auf ihren Websites oder Novelupdates selbst. Weiterhin kann der Erfolg einer Übersetzung an dem Traffic auf der jeweiligen Website gemessen werden sowie an den erhaltenen finanziellen Zuwendungen.

Aus rechtlicher Sicht handelt es sich bei diesen Übersetzungen oft um Grauzonen, schließlich stellen die inoffiziellen Übersetzungen – d. h. solche, die ohne Wissen und Wollen der ursprünglichen Autoren erfolgen – dem Grunde nach Urheberrechtsverletzungen dar, jedenfalls nach dem Verständnis des deutschen Urheberrechtsgesetzes (vgl. § 23 UrhG). Die konkrete Einstufung der Übersetzungen als Urheberrechtsverstoß scheint sich hier nach dem Recht der jeweiligen Herkunftsländer der (Original-) Autoren sowie der Strenge der Ahndung von Verstößen zu richten. Zum Beispiel kann man von chinesischen Novels oft zahlreiche Kopien durch eine einfache Google-Suche

des Originaltitels finden, während Südkorea gegen jegliche illegalen Kopien konsequent vorzugehen scheint, sodass hier nur die offiziellen Versionen online gefunden werden können. Gibt es einen offiziellen englischsprachigen Publisher, so werden auch oft DMCA-Anträge gegen die Hobby-Übersetzer gestellt bzw. werden diese aufgefordert, ihre eigenen Inhalte offline zu nehmen [10, 11].

2.3 Autorenschaftsanalysen

Vereinfacht gesagt, versteht man unter dem Begriff Autorenschaftsanalyse stets die Zuordnung eines Textes zu einem Autor. Dies ist eine extrem vereinfachte Definition und es empfiehlt sich, für ein grundlegenderes Verständnis der Thematik, die einzelnen Teilbereiche der Autorenschaftsanalyse zu betrachten:

Single Authorship Tasks Hierbei wird mit Texten gearbeitet, die jeweils von einem einzelnen Autor verfasst worden sind. D. h. die verschiedenen Texte können verschiedene Autoren haben, aber innerhalb eines Textes hat stets nur ein und derselbe Autor gewirkt. Man unterscheidet hier folgende Teilprobleme:

- **Verifikation (Authorship Verification)**
  Authorship Verification untersucht anhand des jeweiligen individuellen Stils, ob zwei Texte von ein und demselben Autor geschrieben worden sind [12–17]. Jedes der folgenden Probleme lässt sich im Kern auf ein Authorship Verification Problem reduzieren [18].
- **Attribution (Authorship Attribution)**
  Bei Authorship Attribution liegen für eine Menge an Autoren Texte vor. Dabei ist bekannt, welcher der Autoren welchen Text verfasst hat. Ziel ist nun, einen unbekannten Text, entweder einem der bekannten Autoren zuzuordnen (closed set) oder ggf. die Autorenschaft der bekannten Autoren auszuschließen (open set) [19–21].
- **Gruppierung (Authorship Clustering)**
  Beim Authorship Clustering werden einzelne Texte einer Textsammlung anhand ihrer stilistischen Features zu Gruppen/Klassen zusammengefasst. Typischerweise ist Kriterium der Gruppierung der Autor selbst [22–27]. Aber auch andere Merkmale wie bspw. Gruppenbildung nach Geschlecht, Alter, Herkunft etc. wären denkbar.

Multi Authorship Tasks Hierbei wird mit Texten gearbeitet, die jeweils von einem oder mehreren Autoren verfasst worden sein können. D. h. die verschiedenen Texte können verschiedene Autoren haben und innerhalb eines Textes haben ggf. mehrere Autoren gewirkt. Folgende Multi Authorship Tasks können unterschieden werden:

- **Erkennen mehrfacher Autorenschaft (Multi Author Detection)**
  Multi Author Detection beschreibt die grundlegende Aufgabe, automatisch zu erkennen, ob ein Text von mehreren Autoren verfasst worden ist. Dies lässt sich auf das Problem des Erkennens von stilistischen Inkohärenzen zurückführen [28].
- **Erkennen der Anzahl der Autoren (Author Count Prediction)**
  Author Count Prediction ist im Grunde auch eine Form von Multi Author Detection, nur wird hier auch die konkrete Anzahl der am Text beteiligten Autoren ermittelt [29].
- **Erkennen von Stilbrüchen (Style Change Detection)**
  Style Change Detection befasst sich mit der Identifikation von Stilbrüchen in einem Text. Die Annahme dabei ist, dass ein Autorenwechsel mit einem Wechsel des Schreibstils einhergeht. Style Change Detection zielt nun darauf ab, die genaue Stelle des Stil- und Autorenwechsels zu identifizieren [30–32].
- **Attribution einzelner Abschnitte (Multi Author Attribution)**
  Multi Author Attribution ist im Grunde ein Authorship Attribution Problem innerhalb eines Textes. Dies setzt zunächst voraus, dass einzelne, stilistisch verschiedene Textabschnitte bekannt sind (bspw. durch Style Change Detection). Die einzelnen Abschnitte werden dann an eine Menge von (bekannten) Autoren zugeordnet. Hierbei kann man zahlreiche Subtypen unterscheiden [33–36], was an dieser Stelle jedoch nicht näher thematisiert werden soll.

3. Projektaufbau

An dieser Stelle wird ein kurzer Überblick über den grundlegenden technischen Aufbau für die weitere Umsetzung des Projekts gegeben.

Gearbeitet wurde auf zwei verschiedenen PC-Systemen:

A. 8 GB RAM, Intel Core i7-7700, Projekt liegt auf HDD Festplatte, System läuft auf SSD Festplatte

B. 16 GB RAM, AMD Ryzen 5 2600, Projekt liegt auf HDD Festplatte, System läuft auf SSD Festplatte

Diese Angaben dienen dazu, im weiteren Verlauf dieser Arbeit getätigte Aussagen zu Laufzeiten besser einordnen zu können. Die PCs werden im Folgenden als PC-A und PC-B referenziert.

Das Projekt wurde in einer PyCharm IDE [37] entwickelt. Es wurde die Programmiersprache Python 3.9 verwendet. Ferner wurde unter Windows 10 gearbeitet, jedoch ist hierbei ein laufendes Subsystem für Linux (WSL) notwendig, damit die Programme korrekt ausgeführt werden können. Konkret wurde hier innerhalb einer Ubuntu-20.04 Shell gearbeitet.

Der Projektordner gliedert sich im Wesentlichen in vier Unterordner:

1. `basic-setup-tutorials`
   Informationen und Tipps um Projektumgebung einzurichten. Im Rahmen der Arbeit nicht näher betrachtet.

2. `code`
   Hierin sind alle Python Skripte und Module abgelegt.

3. `data`
   Hierin sind alle Daten und Resultate enthalten, z. B. HTML-Dateien, extrahierte Texte (Korpus), Statistiken etc. Aufgrund der großen Datenmenge (ca. 36 GB) ist dieser Ordner nur teilweise im Git-Repositorium enthalten.

4. `tex-files`
   Die LaTeX-Dateien für diese Arbeit.

Darüber hinaus sind im Projektordner noch die Dateien `README.md` – ein kurzer Projektüberblick – und `dataset-creation.md` – Überblick der Konsolenbefehle um die einzelnen beschriebenen Schritte auszuführen – enthalten.

Anmerkung zum Zugang zu diesen Daten
Aufgrund von urheberrechtlichen Bedenken sind die bereitgestellten Daten nicht veröffentlicht. Anforderungen des Datensatzes – d. h. der Rohdaten wie Crawl-Metadaten und HTML-Dateien sowie des finalen Korpus mit zugehörigen Metadaten – für wissenschaftliche Zwecke sind an die Webis-Gruppe [4] zu stellen. Dieser wurde auch ein Klon des (nicht öffentlichen) Git-Repositoriums zur Verfügung gestellt. Eine digitale Kopie des Projekts wurde im Rahmen dieser Arbeit zudem an die Universität Leipzig übergeben.

4. Erzeugung des Korpus

4.1 Anforderungen an das Korpus

4.1.1 Ablage

Die finale Repräsentation des Korpus sollte möglichst einfach gehalten werden. Die einzelnen Texte sollen entsprechend der ursprünglichen URL-ID[2] abgelegt werden. Hierdurch wird eine Nachverfolgung einzelner Texte vom Crawl-Prozess bis zum fertigen, gesäuberten Text gewährleistet. Für die Ablage nach ID wurde eine vierfach geschachtelte Ordnerstruktur gewählt. Dabei richtet sich der Ordnerpfad nach den letzten vier Ziffern des Dateinamens. Damit wird eine gleichmäßigere Verteilung aller Dateien über den Ordnerbaum erreicht, als wenn die vorderen Ziffern gewählt worden wären (siehe Abbildung A.1). Beispielsweise wird eine Datei mit der ID 1234567 unter folgendem Pfad abgelegt: `data/<html/text>/7/6/5/4/1234567.<html/txt>`. Weiterhin werden hierdurch statische Teilmengen gebildet, welche sicherstellen, dass die Ordner nicht übervoll sind und auch mit Systemtools gut betrachtet werden können. Zudem werden die Dateien dabei inhaltlich nicht zusammenhängend über die Unterordner verteilt, sodass einzelne Unterordner repräsentative Stichproben aus dem gesamten Korpus darstellen (siehe auch Abschnitt 4.3.1).

4.1.2 Anforderung an die Texte

Die Texte sollen zuallererst reinen Story-Inhalt wiedergeben und frei von Headern/Footern, Kommentaren und Anmerkungen der Autoren sein. Weiterhin sollen die Texte möglichst vollständig sein. Damit sind insbesondere bloße Teasertexte oder Texte von nur wenigen Zeichen tendenziell unerwünscht.

[2]Die ID des HTML- bzw. Textdokumentes ergibt sich aus dem Link von Novelupdates, welcher auf den externen Inhalt (eigentlichen Text) verweist. Die ID wird somit während dem Crawl automatisch anhand dieses Links vergeben.

Im Zweifel ist die Sauberkeit von Texten zulasten der Vollständigkeit höher zu gewichten.

4.1.3 Metadaten

Ergänzende Informationen zu den Texten sollen in einer separaten json-Datei gespeichert und den einzelnen Texten per ID zuordenbar sein. Dabei sollen den Texten über die ID insbesondere folgende Informationen zugeordnet werden:

- Novel
- Kapitel
- Übersetzer/Gruppe
- Genre
- kurze Stichworte zum Inhalt (Tags)

Die genannten Metadaten sollen während dem Crawl von Novelupdates direkt bezogen werden und nicht etwa durch Extraktion aus den HTML-Dateien. Novelupdates wird insofern als verlässliche fehlerfreie Quelle für diese Daten betrachtet.

4.2 Crawling von novelupdates.com

Der Crawlingprozess und die Auswahl von 331.170 URL-Adressen wurden im Wesentlichen durch den Mentor dieser Arbeit – Erik Körner – vorbereitet.[3]

Es wurden zunächst die Ordner `crawl` und `data` über das Ceph-System der Webis-Gruppe [4] bezogen. Der Ordner `data` ist dabei nicht mit dem Überordner für Daten im Rahmen dieser Arbeit, welcher ebenfalls mit `data` benannt ist, zu verwechseln. Der korrekte Pfad für den bereitgestellten `data`-Ordner ist somit `data/data`. Dieser Ordner enthält wichtige Metadaten, die während des Crawling-Prozesses gewonnen worden, u. a. die Namen der Übersetzungsgruppen (`groups.txt`), die meisten der unter Abschnitt 4.1.3 spezifizierten Informationen (`series_chapters_multi.jsonl`) und eine Liste mit indexierten URLs (`chapter_urls.tsv`). Der Ordner `data/crawl` enthält als Ergebnis des Crawling-Prozesses sechs WARC-Dateien mit dem Inhalt der jeweiligen Websites.

[3] Anfragen diesbezüglich sind an die Webis-Gruppe [4] zu richten.

Wie bereits in der Einleitung kurz erwähnt, wurde bei der Auswahl der URLs darauf geachtet, nur Links zu Kapiteln von solchen Novels auszuwählen, bei welchen sich zwischen den Kapiteln ein Wechsel der Übersetzenden vollzogen hat.

Aus den WARC-Dateien wurden die HTML-Dateien einzeln extrahiert und in einer verschachtelten Ordnerstruktur unter `data/html` abgelegt.

Bei diesem grundlegenden Prozess sind hier zwei Fälle zu unterscheiden:

Heritrix-Crawl Ein erster Durchlauf dieses Prozesses mit einem Heritrix-Crawler [38] lieferte keine zufriedenzustellenden Ergebnisse. Auch waren die hierbei „roh“ extrahierten HTML-Dateien teilweise mit mutmaßlichen Schadskripten infiziert (ein Antivirenprogramm schlug an). Hierbei bereits gewonnene Ergebnisse und Metadaten wurden in den Ordnern im `data`-Verzeichnis unter `data_old` und `crawl_old` abgelegt. Dieser Prozess wird im Folgenden als **„alter Crawl“** referenziert.

Scriptor-Crawl Das bereits beschriebene Vorgehen wurde mit einem Scriptor-Crawler [39] wiederholt. Hierbei wurden auch die HTML-Dateien direkt nach dem Crawl vorverarbeitet und (potenziell schädliche) Skripte, Links, etc. wurden entfernt. Dabei verblieben 322.931 vorverarbeitete HTML-Dateien, welche über das Ceph-System bezogen wurden und den Ausgangspunkt für die weitere Textextraktion bildeten. Dieser Prozess ist im Folgenden als **„aktueller Crawl“** bzw. nur **„Crawl“** referenziert. Er ist als Standardfall zu verstehen und es ist davon auszugehen, dass dieser Crawl gemeint ist, wenn nicht ausdrücklich auf den alten Crawl verwiesen wird.

4.3 Extraktion der Texte

Aus den jetzt einzeln vorliegenden HTML-Dateien galt es nun, die eigentlichen Inhalte, d. h. die Texte der Novels, zu extrahieren. Hierfür wurden verschiedene Ansätze getestet:

1. Beautiful Soup in Kombination mit einer (weiteren) Vorsäuberung der HTML-Dateien
2. HTML2Text
3. Parsel
4. Readability
5. Readability in Kombination mit HTML2Text

6. Resiliparse
7. Trafilatura

4.3.1 Auswahl einer Testmenge

Diese Ansätze wurden zunächst auf einer Testmenge erprobt, da es nicht sinnvoll wäre, bei wiederholtem Ausprobieren regelmäßig über 300.000 Dateien zu verarbeiten. Hierfür wurde das Verzeichnis `data/html/0/0/0` ausgewählt. Darin waren 312 HTML-Dateien enthalten. Aufgrund der Gleichverteilung der letzten Ziffern der IDs stellt diese Auswahl eine repräsentative Stichprobe von ca. $\frac{312}{322.931} = 0,1\%$ der Gesamtmenge aller Dateien dar. Ebenso waren Texte von vier der fünf größten Übersetzungsgruppen entsprechend dem ihrer Größe nach ungefähr zu erwartenden Anteil vertreten, siehe `data/results/found_ids_of_largest_groups_testset.txt`[4].

4.3.2 Beautiful Soup mit Vorsäuberung der HTML-Dateien

Beautiful Soup [40] ist eine Python Bibliothek von Leonard Richardson mit der u. a. HTML-Baumstrukturen durchlaufen, durchsucht und verändert werden können. Für diese Arbeit war besonders die Funktion `get_text()` relevant. Hierbei kann eine HTML-Datei (als Beautiful-Soup-Objekt) übergeben werden und die Funktion extrahiert alle Strings des übergebenen Objekts. Dies funktioniert nicht nur auf kompletten HTML-Bäumen (also der ganzen Website), sondern auch bloßen Fragmenten hiervon. Wird also die komplette Website als HTML-String übergeben, werden hierdurch alle enthaltenen Textstrings zurückgegeben.

Ein Problem bei dieser Herangehensweise liegt allerdings darin, dass nicht nur die eigentlichen Novel-Inhalte in den HTML-Dateien in textueller Repräsentation vorliegen, sondern ebenso unerwünschte Inhalte wie Header, Kommentarsektionen und Anmerkungen der Übersetzer. Diese Inhalte vom gewünschten Belletristik-Inhalt zu trennen, ist die Hauptschwierigkeit bei der Textextraktion und wird auch bei den weiteren Ansätzen den Kern der Betrachtung darstellen.

Da Beautiful Soup auch HTML-Fragmente verarbeiten kann, wurde

[4]Größte Gruppen in Bezug auf die Rohdaten. Nicht zu verwechseln mit den größten Gruppen im fertigen Korpus, siehe Abbildung A.5. Aber auch davon sind noch drei der größeren Gruppen vertreten.

zur Extraktion der reinen Novels versucht, den bereits gesäuberten HTML-Baum (siehe Abschnitt 4.2) weiter vorzubearbeiten. Hierfür wurde ein „Drei-Phasen-Filter“ implementiert, um folgende Schritte durchzuführen:

1. Als Erstes wurden Elemente des HTML-Baumes wie Header, Footer, Überschriften, Links und Navigationslinks entfernt, sofern diese nicht bereits entfernt waren.

2. Im zweiten Schritt wurden mittels Blacklists gezielt Teilstrukturen des HTML-Baumes entfernt. Bspw. wurde nach Tags gesucht, welche „comment“ oder „popup“ enthielten. Ebenso wurden Tags welche IDs mit vier oder mehr Ziffern enthielten entfernt, da diese überwiegend ungewollten Inhalt darstellten.

3. Schließlich wurde die vorgefilterte HTML-Datei noch auf solche Tags reduziert, welche überwiegend den gewünschten Inhalt darstellten, wie z. B. „article“ oder „id: chp_raw“.

Es zeigte sich, dass hier überwiegend heuristisch gearbeitet werden musste, um Strukturen zu finden, welche gefiltert werden sollen und diese von denen, die bewahrt werden sollen, abzugrenzen. Dabei stellte sich heraus, dass, auch wenn mit bestimmten Kriterien dokumentenübergreifend bestimmte Filter wirken, es immer wieder Ausnahmen hiervon gibt. D. h. es wurden bei manueller Kontrolle der Resultate wiederholt Fälle festgestellt, bei denen weiterhin ungewollter Text vorhanden war oder gewollter Inhalt unzutreffend herausgefiltert wurde. Oft wurde das HTML-Dokument hierbei komplett ins Unbrauchbare reduziert. Folglich war festzustellen, dass dieser Ansatz für sehr große Datenmengen mit teils stark differierenden Dokumentenstrukturen keine ausreichenden Ergebnisse liefern konnte und zu viel manuelles Arbeiten an den Dateien erfordert. Aus oben genannten Gründen wurde für die folgenden Ansätze vom Versuch abgesehen, die jeweils inhomogenen HTML-Bäume weiter vorzubearbeiten, als dies bereits in Abschnitt 4.2 beschrieben wurde.

4.3.3 HTML2Text

HTML2Text [41] bietet ebenfalls Möglichkeiten zur Textextraktion an. Dabei können bestimmte Elemente wie Links oder Bilder ausgeschlossen werden. Header und Footer sowie Kommentare blieben jedoch erhalten, weshalb die Ergebnisse im Vergleich zu anderen Ansätze mehr ungewollte Inhalte enthielten.

4.3.4 Parsel

Parsel [42] bietet ähnlich wie Beautiful Soup Funktionalitäten, um einzelne Elemente des HTML-Baums zu adressieren. Es wurde hiermit nach Abschnitten im HTML-Dokument gesucht, wo der gewünschte Inhalt vermutete wurde, bspw. `<p>` oder `<div>` Tags. Die Ergebnisse sind sauberer als die Extraktion mit HTML2Text. Gleichzeitig bleibt jedoch bereits das in Abschnitt 4.3.2 beschriebene Problem der Inhomogenität der HTML-Dateien aufgrund der großen Datenmenge bestehen, weshalb auch hier noch Header, Footer und Kommentare enthalten sind.

4.3.5 Readability

Readability [43] ermöglicht mit seiner `summary()` Methode den Text aus dem Hauptteil eines HTML-Dokumentes zu extrahieren. Darüber hinaus werden jedoch wenig Funktionalitäten für die hier benötigten Zwecke angeboten. Die Extraktion liefert den Text aus den HTML-Dokumenten zwischen den äußeren `<div>` Tags. Dabei bleiben die einzelnen Tags im Text stehen und oft ist auch innerhalb der `<div>` Tags ungewünschter Inhalt enthalten gewesen. Insgesamt überzeugten die Ergebnisse mit diesem Ansatz nicht.

4.3.6 Readability in Kombination mit HTML2Text

Bei diesem Ansatz wurden die zuvor beschriebenen Methoden HTML2Text (siehe Abschnitt 4.3.3) und Readability (siehe Abschnitt 4.3.5) kombiniert. Zunächst wurde das jeweilige HTML-Dokument auf den Hauptteil zwischen den äußeren `<div>` Tags mittels Readability reduziert. Anschließend wurde diese Struktur mit HTML2Text gesäubert. Durch die Reduktion des Dokumentes vor der Säuberung werden hier im Vergleich zu den vorigen Ansätzen weit bessere Ergebnisse erzielt. Der extrahierte Inhalt ist relativ sauber, jedoch sind auch hier, besonders am Anfang oder Ende der Texte, weiterhin unerwünschte Inhalte vorhanden.

4.3.7 Resiliparse

Die Extraktion der Haupttexte mit Resiliparse [44, 45] wies im Vergleich zu anderen Ansätzen eine ähnliche Reinheit wie Parsel (siehe Abschnitt 4.3.4) auf. Header, Footern und Kommentare waren jedoch teilweise weiterhin vorhanden.

4.3.8 Trafilatura

Die von Adrien Barbaresi entwickelte Python Bibliothek Trafilatura [46] überzeugt durch ihre einfache Handhabung und Syntax. Die `extract()` Methode bietet neben der Textextraktion zusätzliche Parameter hierfür, darunter u. a. Funktionalitäten um Kommentarsektionen zu entfernen oder das Ausgabeformat zu definieren (.txt oder .xml). Darüber hinaus ist ein eigener Webscraper enthalten, womit es möglich ist, anstatt der HTML-Dateien auch einzelne URLs oder URL-Listen zu übergeben. Letzteres wurde im Rahmen dieser Arbeit jedoch nicht verwendet, da die HTML-Dateien bereits vorlagen (vgl. Abschnitt 4.2).

Ein manuelles Durchsehen der Ergebnisse zeigte, dass diese insgesamt eine hohe Reinheit und kompakte Texte unter geringstem Aufwand lieferten. Es wurde weitestgehend kein gewollter Inhalt unzutreffend entfernt. Unreinheiten konnten jedoch weiterhin nicht komplett und zufriedenstellend entfernt werden. Bspw. waren Überschriften mit Kapitelnummer noch enthalten, ebenso wie Anmerkungen der Übersetzer im laufenden Text, davor oder danach, insbesondere wenn diese in `<p>` Tags angebracht worden sind und somit syntaktisch nicht vom restlichen Text zu unterscheiden waren. Im Vergleich zu allen anderen getesteten Ansätzen zeigte die manuelle Überprüfung der Ergebnisse hier jedoch die höchste Reinheit.

4.3.9 Fazit

Die Laufzeiten der einzelnen Ansätze und die bei manueller Betrachtung erkannte Sauberkeit der Ergebnisse sind in Tabelle 4.1 zusammengefasst.

Erneute Messungen der Laufzeiten (für tatsächliche oder simulierte Extraktion) zeigten, dass diese zum Teil stark schwanken. Dies wird zum einen dadurch bedingt, ob die Extraktion bereits zuvor laufen gelassen wurde (nun also wiederholt wird), da hier der Arbeitsspeicher wohl noch nicht geleert worden ist. Zum anderen dürfte auch die Auslastung des Systems durch andere Prozesse ausschlaggebend sein.
Ein Beispiel: Eine Extraktion der Trafilatura Testdateien noch auf den alten Crawldaten (siehe Abschnitt 4.2) wurde bei der ersten Extraktion auf PC B mit 21,01 Sekunden gemessen (wie in der Tabelle dargestellt). Ein später durchgeführter Simulationsdurchlauf auf PC B benötigte nur noch 16,78 Sekunden. Eine direkt im Anschluss durchgeführte Simulation auf dem gleichen System benötigte lediglich 11,87 Sekunden. Damit sind auch die zum Teil besseren Laufzeiten auf PC A (dem leistungsschwächeren System) zu erklären. Jedoch werden die Prioritäten bei der Textextraktion ohnehin bei der

Ansatz/Framework	Laufzeit PC A.	Laufzeit PC B.	Fehler bei der Extraktion	Sauberkeit
Beautiful Soup	7,62	11,33	0	◔
HTML2Text	11,65	11,22	0	◔
Parsel	16,07	8,55	1	◑
Readability	11,86	11,61	0	○
Readability + HTML2Text	14,30	13,44	3	◕
Resiliparse	15,56	7,61	28	◑
Trafilatura	22,87	15,91	1	●

Tabelle 4.1: Überblick über die Extraktionsergebnisse

Die Laufzeiten sind in Sekunden angegeben. Die Extraktionsfehler beziehen sich auf jeweils insgesamt 312 Dateien. Dabei ist bei nicht vorhandenen Extraktionsfehlern nicht zwingend davon auszugehen, dass alles in Ordnung war. Bei der Extraktion mittels Beautiful Soup traten beispielsweise keine Fehler im eigentlichen Sinne auf, aber viele der extrahierten Dateien waren komplett leer, d. h. sämtlicher Text war herausgefiltert worden. Bezüglich der Sauberkeit bedeutet ○ „eher unsauber“ und ● „eher sauber“. Dabei war die Sauberkeit nicht vollständig objektiv quantifizierbar; dargestellt ist der subjektive Eindruck beim manuellen Prüfen der Ergebnisse.

Reinheit der Ergebnisse gesetzt. Die Laufzeit wird als lediglich untergeordneter Aspekt angesehen, da es sich bei der Textextraktion grundsätzlich nicht um einen zeitkritischen, oft angeforderten Prozess handelt. Ist ein Ansatz erst etabliert und liegt ein Datensatz an zu extrahierenden Dokumenten vor, wird die Textextraktion in der Regel nur einmal durchlaufen müssen. Hierbei wird eine einmalige längere Laufzeit – auch von mehreren Stunden – als akzeptabel angesehen, insofern dies durch bessere Ergebnisse gerechtfertigt wird.

Im Hinblick auf die Sauberkeit der Extraktion arbeitete Trafilatura am überzeugendsten. Hier war die Syntax der Implementierung sehr einfach und die extrahierten Texte wiesen im Vergleich zu den anderen Ansätzen die höchste Dichte an gewolltem Inhalt auf. Dies zeigte sich insbesondere auch bei einem früheren Test der Frameworks auf den alten Crawl-Daten und damit den „rohen“, nicht vorverarbeiteten HTML-Dateien. Hier stach Trafilatura sogar noch mehr im Vergleich zu den anderen Ansätzen hervor. Daher wurde schließlich Trafilatura für die Textextraktion ausgewählt.

4.4 Säuberung der Texte

Wie sich in Abschnitt 4.3 gezeigt hat, lagen die Texte nach der Extraktion aus den HTML-Dateien noch nicht in einem zufriedenstellenden Zustand vor. Daher waren weitere Säuberungsschritte auf den mittlerweile vorliegenden Textdateien notwendig. Hierbei wurde ein zweistufiges Verfahren

implementiert:

1. Zunächst wurden Texte, welche im Ganzen unbrauchbar waren, entfernt.
2. In einem weiteren Schritt wurden die verbliebenen Texte von ungewünschten Inhalten gereinigt.

4.4.1 Filtern ganzer Dateien

Im ersten Säuberungsschritt wurde zunächst versucht, einen heuristischen Ansatz zur Bereinigung der nun vorliegenden Textdateien aus der Testmenge zu etablieren. Da der Extraktionsansatz mit Trafilatura die besten Ergebnisse lieferte, wurde auf diesen Textdateien weitergearbeitet (`data/text_testsets/trafilatura`).

Zunächst wurden dafür die vorliegenden 311 Textdateien (312 Textdateien abzüglich eines Extraktionsfehlers) auf ihre Länge in Zeilen und Zeichen hin untersucht, um hieraus ggf. einen Filter ableiten zu können. Die Daten von Tabelle 4.2 wurden maschinell ermittelt und ergeben sich aus `data/results/textcleaning/text_testset_trafilatura_stats.txt`.

	Zeilen	**Zeichen**
∅ Datei-Länge	72,29	8.623,23
Median	75,5	7.452
erstes Quartil der Länge	38,75	2.586
erstes Quintil der Länge	30,6	2.035,8
erstes Dezil der Länge	14,3	366,6

Tabelle 4.2: Analyse der Testdateien hinsichtlich Länge in Zeilen und Zeichen

Mit diesen Werten wurden verschiedene Ansätze getestet, um solche Textdateien zu filtern, die lediglich sogenannte Teaserseiten oder generell nur wenig und/oder keinen sinnvollen Text enthalten. Diese Dateien zeichnen sich regelmäßig durch ihre Kürze aus und sollten demnach automatisch identifizierbar sein. Als Beispiele können folgende zwei Dateien genannt werden:
`data/text_testsets/trafilatura/0/920000.txt`
`data/text_testsets/trafilatura/1/5201000.txt`

Um verschiedene Ansätze evaluieren zu können, war es zunächst notwendig, die Testdateien zu labeln. Hierfür wurden im Ordner `data/results/labels` die Testdateien binär in die Kategorien „behalten“ (True) oder „nicht-behalten“ (False) klassifiziert.

Da eine bloße Verwendung von fixen Werten anhand Tabelle 4.2 nicht zielführend erschien, wurden zusätzlich Klassifikatoren mithilfe der ML-Bibliothek SK-Learn [47, 48] trainiert. Um ein Overfitting des Klassifikators auf den Testdaten (`data/text_testsets/trafilatura` basierend auf `data/html/0/0/0`) zu vermeiden, wurde aus `data/html/0/0/1` mittels trafilatura ein weiterer Textsatz für Trainingszwecke extrahiert sowie klassifiziert und unter `data/text_testsets/trafilatura_001` abgelegt.

Konkret wurden zunächst folgende Ansätze für fixe Grenzwerte – jeweils für Zeilen und Zeichen – getestet, wobei alle Dateien, die mit ihrer Zeilen-/Zeichenzahl diesem Wert entsprachen oder darunter lagen, weggeworfen wurden:

Methode 1: 20% der durchschnittlichen Zeilen-/Zeichenzahl (Pareto)

Methode 2: Erstes Quartil

Methode 3: Erstes Quintil

Methode 4: Erstes Dezil

Methode 5: Logistic Regression Klassifikator, vektorisiert auf Wortbasis mit Count-Vectorizer

Methode 6: Logistic Regression Klassifikator, vektorisiert auf Wortbasis mit TFIDF-Vectorizer

Die Auswertung der verschiedenen Ansätze ist in `data/results/textcleaning/whole_file_filtering.txt` automatisiert generiert worden und in Abbildung A.2 grafisch dargestellt. Tabelle 4.3 gibt an dieser Stelle einen kurzen Überblick.

Wie sich zeigt, schnitt der ML-Ansatz auf Basis des Count-Vectorizer hierbei am besten ab. Hier war die Gesamtfehlerrate mit Abstand am geringsten, sodass diese Methode am geeignetsten erscheint. Insbesondere bestanden hier die vorhandenen Fehler zu 80% aus falsch weggeworfenen Dateien, was weniger kritisch anzusehen ist, da eine Reinheit des Korpus höher gewichtet wird als ein Fortbestand aller vorhandenen Ausgangsdateien.

Dennoch waren die verbliebenen Texte weiterhin nicht sauber genug. So waren in den Dateien weiterhin Anmerkungen der Autoren in den Texten vorhanden oder Header etc. standen weiterhin vor dem Text (z. B. `data/results/text_cleaned_testset/filter_whole_files/8000.txt` Zeile 1–6).

Filter	Acc. 5×CV	Acc.	Prec.	Recall	Fehlerrate
20% ∅ Zeilen	-	-	-	-	3,54%
erstes Quartil Zeilen	-	-	-	-	13,83%
erstes Quintil Zeilen	-	-	-	-	10,29%
erstes Dezil Zeilen	-	-	-	-	3,54%
20% ∅ Zeichen	-	-	-	-	8,36%
erstes Quartil Zeichen	-	-	-	-	2,89%
erstes Quintil Zeichen	-	-	-	-	7,07%
erstes Dezil Zeichen	-	-	-	-	17,36%
Count Vect. LogReg.	96,67%	98%	98%	98%	1,61%
TFIDF Vect. LogReg.	79,70%	72%	52%	72%	27,97%

Tabelle 4.3: Vergleich der Klassifikationsergebnisse für ganze Dateien

Bei festen Werten als Grenze der Klassifikation wurde allein die Fehlerrate betrachtet. Für ML-Ansätze wurde zum einen die Accuracy des Klassifikators bei 5-facher Kreuzvalidierung betrachtet, zum anderen Accuracy, Precision und Recall bei der Anwendung auf den konkreten Testdaten.

4.4.2 Filtern einzelner Zeilen

Um die weiterhin vorhandenen Unreinheiten in den Texten zu entfernen, wurde ein Verfahren zum Entfernen der ungewünschten Zeilen entworfen. Da einzelne Sätze in den Texten regelmäßig nicht zeilenübergreifend vorlagen, wurde ein zeilenweiser Filter einem Filter für einzelne Wörter vorgezogen. Auch hier galt, dass der Verlust einiger gewünschter Zeilen (falsche Negative) weniger gewichtet wurde, als das unzutreffende Behalten von ungewünschten Zeilen (falsche Positive).

Um die Zeilen der nach dem Filtern ganzer Dateien (siehe Abschnitt 4.4.1) verbliebenen Texte zu labeln, wurden zunächst „Gold-Daten“ für das Testset erstellt, d. h. die Texte wurden manuell bereinigt und unter `data/gold_data/` abgelegt. Ein maschineller Vergleich der Originaldaten mit den Gold-Daten, ob die jeweiligen Zeilen noch vorhanden waren, lieferte die Labels für einzelne Zeilen (Zeile noch da = True, Zeile fehlt = False). Die gelabelten Zeilen je Datei sind in `data/results/labels/trafilatura_000_lines.jsonl` hinterlegt.

Anschließend wurden folgende Verfahren zum Filtern der Zeilen getestet:

Blacklist Zeilen, die bestimmte Wörter enthalten, welche überwiegend in unerwünschten Headern oder Kommentaren der Übersetzer vorkommen (z. B. „Discord“, „TL Note“, „donation“), sollten entfernt werden. Hier musste

wieder überwiegend heuristisch gearbeitet werden und eine Vollständigkeit der Blacklist konnte nicht gewährleistet werden.

Abschneiden von drei oder fünf Zeilen am Anfang und Ende (Chopping) Da Anmerkungen der Autoren sowie Header und Footer regelmäßig am Anfang und Ende des Textes vorkommen und eher selten im laufenden Text, wurden entsprechende Zeilen pauschal abgeschnitten. Enthielt die Datei weniger als das Doppelte der abzuschneidenden Zeilen plus Eins, wurde der Text gänzlich weggeworfen. Sollten bspw. fünf Zeilen geschnitten werden, musste der Text also mindestens elf Zeilen lang sein, damit am Ende noch eine Zeile übrig blieb. Dies kann zudem als weiterer Schritt zum Entsorgen ungewöhnlich kurzer Texte, ergänzend zum Filtern ganzer Dateien (siehe Abschnitt 4.4.1), verstanden werden.

Klassifizierung der Zeilen mittels ML Auf den Zeilen der ersten Hälfte der gelabelten Testdateien wurden verschiedene Klassifikatoren trainiert. Genutzt wurden Logistic Regression (LogReg), Support Vector Classifier (SVC) und Random Forest Classifier (RFC). Zudem wurden verschiedene Variationen mit Count- oder TFIDF-Vectorizer auf Wort oder 3-gram Basis ausprobiert. Schließlich wurde sich für die Nutzung von TFIDF auf 3-gram Basis für abschließende Vergleiche entschieden, da sich hier bessere Ergebnisse abzeichneten. Die Klassifikation erfolgte jeweils auf nachträglich balancierten Datensätzen, da der überwiegende Teil der Zeilen pro Text regelmäßig positiv (True) gelabelt wurde.

Chopping und Klassifizierung der übrigen Zeilen Kombination der letzten zwei Methoden.

Die Anzahl der Fehler je Methode und deren jeweilige Art ist in `data/results/textcleaning/line_filter_stats.txt` dargestellt. Eine grafische Darstellung der Fehler in Relation zu den richtig gelabelten Zeilen findet sich im Anhang in Abbildung A.3. Tabelle 4.4 gibt an dieser Stelle einen kurzen Überblick.

Wie sich zeigt, schnitt die Klassifikation mittels Random-Forest-Classifier, nach einem generellen Abschneiden von fünf Zeilen zu Beginn und am Ende des Textes, am besten ab. Auch wenn hierbei die totale Anzahl an Fehlern höher ist als bei anderen Methoden, bzw. nicht die höchste Accuracy erreicht worden ist, waren hier die falschen Positive am geringsten, was – wie eingangs erwähnt – höher gewertet wird. Ergänzend wurde für diesen Filter noch der Effekt des Vorfilterns von ganzen Dateien (siehe Abschnitt 4.4.1) gemessen („Prefilter“). Dafür wurde bei allen Zeilen von Dateien, welche nicht

Filter	Acc. 5×CV	Acc.	Prec.	Recall	Fehler total	F. Pos.
Chop 3	-	91,30%	94,08%	96,39%	627	393
Chop 5	-	89,14%	95,14%	92,66%	783	307
Blacklist	-	94,88%	95,18%	99,34%	369	326
LogReg.	83,53%	88,69%	96,69%	90,53%	815	201
SVC	86,21%	89,55%	97,00%	91,21%	753	183
RFC	87,53%	90,60%	97,73%	91,67%	678	138
Chop 3 + LogReg.	-	87,38%	98,47%	87,32%	910	88
Chop 5 + RFC	-	85,85%	98,89%	85,22%	1020	62
Chop 5 + RFC + Prefilter	-	85,24%	98,95%	84,48%	1064	58

Tabelle 4.4: Vergleich der Klassifikationsergebnisse für einzelne Zeilen
Die Werte Accuracy, Precision und Recall wurden anhand der tatsächlichen Klassifikation der Testdaten (true/false positives/negatives) berechnet. Sofern ein Klassifikator im Vorfeld auf Trainingsdaten trainiert worden ist, wurde hier noch das Ergebnis einer 5-fachen Kreuzvalidierung angegeben. Die Testdaten umfassten insgesamt 7.209 Zeilen. Eine Zeile bestand gewöhnlich aus 1–3 Sätzen.

im Verzeichnis der vorgefilterten Dateien (`data/text_cleaned_testset/filter_whole_files`) lagen, das Label „False“ vergeben. Es ergab sich hierdurch kein wesentlicher Unterschied. Daraus lässt sich schließen, dass die Kombination von pauschalem Abschneiden sowie Klassifikation übriger Zeilen die Arbeit des Vorfilterns ganzer Dateien miterledigen würde. Da sich jedoch auch keine Verschlechterung einstellte und hier die Laufzeit der Anwendung mehr oder weniger unkritisch ist, wurde entschieden, beide Schritte separat und ergänzend beizubehalten.

4.4.3 Verarbeitung aller Dateien – parallele Cleaning Pipeline

Die unter Abschnitt 4.4.1 und Abschnitt 4.4.2 etablierten Ansätze wurden kombiniert auf die gesamten 319.483 Textdateien angewendet. Dieser Schritt wurde in einer parallelen „Cleaning-Pipeline“ durchgeführt, um Laufzeit einzusparen. Die verbliebenen 232.398 Textdateien bilden das finale Novelupdates-Korpus.

5. Statistiken und Metadaten zum fertigen Korpus

5.1 Anzahl Dateien/Texte

Das finale Korpus enthält 232.398 verschiedene Texte. Die Texte sind nach dem Cleaning Prozess (siehe Abschnitt 4.4) und entsprechend einem manuellen Durchsehen überwiegend frei von ungewünschten Artefakten. Eine vollständige Reinheit kann aber nicht garantiert werden, da hierfür alle 232.398 Texte manuell bewertet werden müssten. Als Beispiele für mögliche verbliebene Unreinheiten können die Texte mit den IDs 2625807, 3450588 und 3289712 dienen. Diese wurden im Rahmen eines Autorenschaftsexperimentes im TEN-Datensatz (siehe Abschnitt 6.1) entdeckt und sind in Anhang A.3 dargestellt.

5.2 Textlänge

Die durchschnittliche Länge eines Textes beträgt 80,83 Zeilen, 1.857,44 Wörter und 10.372,17 Zeichen. Dies entspricht in etwa den anhand der Testdaten (siehe Abschnitt 4.3.1 und Tabelle 4.2) zu erwartenden Werten.

5.3 Autoren

Die vorhandenen Texte verteilen sich auf insgesamt 2.537 verschiedene Autoren/Übersetzer. Dabei ist erneut darauf hinzuweisen, dass der „Übersetzer“/„Autor“ im Rahmen dieser Arbeit stets als die Übersetzungsgruppe (bspw. wuxiaworld) betrachtet wird und nicht als die einzelne Person innerhalb der Gruppe, welche den Text erstellt hat.

Die Mehrzahl der im Korpus vertretenen Autoren hat weniger als zwei Novels

bearbeitet. Zwei Gruppen haben indes an über 100 Novels gearbeitet. Im Durchschnitt bearbeitete eine Gruppe 2,36 Novels.

Die Anzahl der bearbeiteten Novels ist nicht zu verwechseln mit der Anzahl an übersetzten Kapiteln. Die Mitarbeit an vielen Novels bedeutet nicht zwangsläufig, dass auch tatsächlich viele Kapitel übersetzt worden sind. Hat z. B. eine Gruppe A bei 100 Novels je ein Kapitel übersetzt und eine Gruppe B bei nur einer Novel 200 Kapitel übersetzt, so hat Gruppe B insgesamt mehr Kapitel übersetzt, auch wenn Gruppe A im Hinblick auf die Novels die „größere“ Gruppe ist. Daher ist bspw. im Hinblick auf Novels foxaholic die größte Gruppe, während dies im Hinlick auf tatsächlich übersetze Kapitel mit Abstand wuxiaworld ist.

Die durchschnittliche Anzahl übersetzter Kapitel je Gruppe beträgt 91,60. Dabei gibt es auch im Hinblick hierauf einige wenige, sehr große Gruppen mit tausenden oder – im Fall von wuxiaworld – sogar mehr als 12.000 übersetzten Kapiteln.

Die Abbildungen 5.1, 5.2, A.4 und A.5 veranschaulichen diese Zusammenhänge.

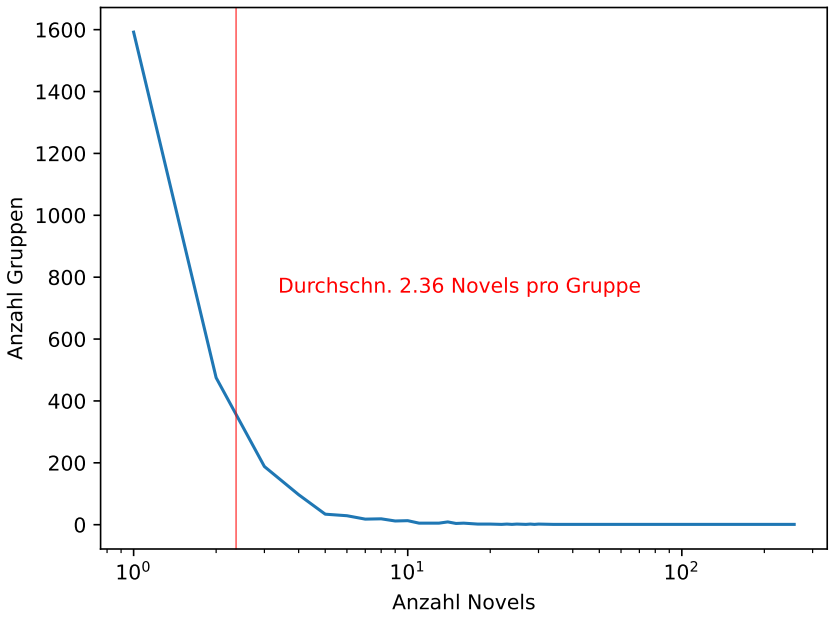

Abbildung 5.1: Anzahl der Novels pro Gruppe

Man sieht, dass die meisten Gruppen an weniger als zwei Novels gearbeitet haben. Einige wenige Gruppen haben jedoch mehr als 100 Novels bearbeitet, weshalb die x-Achse logarithmiert wurde.

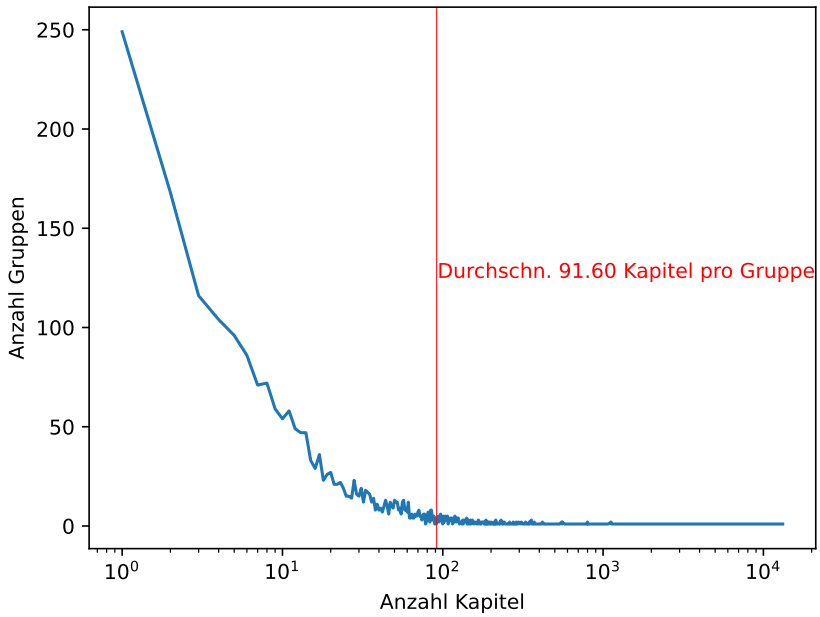

Abbildung 5.2: Anzahl der Kapitel pro Gruppe

Man sieht, dass die meisten Gruppen weniger als 100 Kapitel übersetzt haben. Einige wenige Gruppen haben jedoch mehr als 1000 Kapitel bearbeitet, weshalb die x-Achse logarithmiert wurde.

5.4 Novels

Das Korpus präsentiert Texte aus 2.818 Novels. D. h. 2.818 Novels waren mit mindestens einem Text vertreten.

Dabei handelt es sich bei 2.127 Novels um solche, an denen mehr als eine Übersetzungsgruppe partizipiert hat und auch tatsächlich für mindestens zwei verschiedene dieser Gruppen mindestens ein Text aus dieser Novel vorliegt (Multi-Autor-Novels).

Gleichzeitig haben an den wenigsten Novels mehr als vier Übersetzungsgruppen partizipiert. Der Regelfall liegt bei ca. zwei Gruppen pro Novel, wie auch Abbildung 5.3 zeigt.

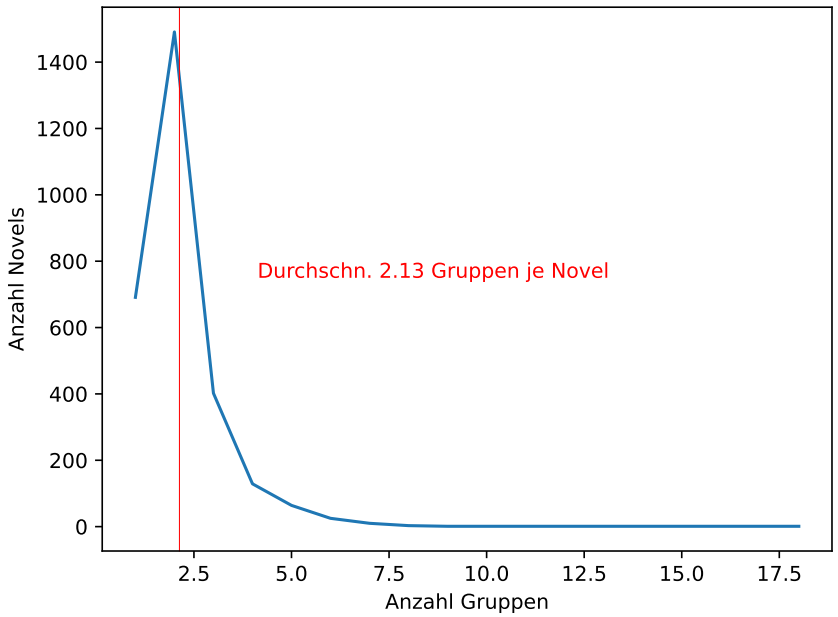

Abbildung 5.3: Beteiligte Autoren pro Novel

Dabei kann es sich um parallel übersetzte Kapitel handeln (siehe Abschnitt 5.5) oder es wurden unterschiedliche Kapitel der Novel von unterschiedlichen Gruppen übersetzt.

5.5 Kapitel

Die Novels im Korpus bestehen wiederum aus einzelnen Kapiteln. Dabei entspricht die Gesamtkapitelzahl grundsätzlich den Dateien bzw. Texten im Korpus, also 232.398.

Hierin können jedoch sogenannte „parallel übersetzte Kapitel“ enthalten sein. Dabei handelt es sich um Kapitel, die mindestens von zwei Gruppen separat übersetzt worden sind, sodass ein Text a einer Gruppe A und ein Text b einer Gruppe B vorliegt. Der grundsätzliche Inhalt der Texte bleibt dabei jedoch gleich, da beide Übersetzungen auf denselben asiatischen Originaltext zurückgehen. Im Korpus sind 5.521 solcher parallel übersetzten Kapitel enthalten. Dabei liegen für 5.425 dieser parallelen Kapitel zwei Versionen, für 88 Kapitel drei Versionen, für fünf Kapitel vier Versionen, für zwei Kapitel fünf Versionen und für ein Kapitel sechs Versionen vor. In Summe bestehen die parallel übersetzten Kapitel also aus 11.150 Einzeltexten. Eine Novel enthält im Schnitt 5,95 parallel übersetzte Kapitel, wie Abbildung 5.4 zeigt.

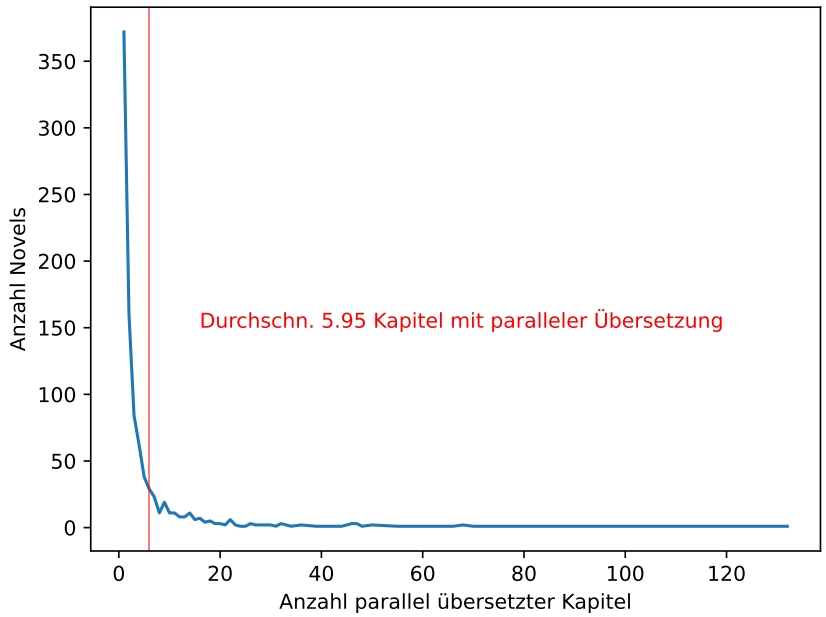

Abbildung 5.4: Parallel übersetzte Kapitel je Novel

Die Abbildungen 5.5 und 5.6 zeigen, dass im Durchschnitt 82,47 Kapitel auf eine Novel entfallen oder anders formuliert eine Novel aus durchschnittlich 82,47 Einzeltexten besteht.

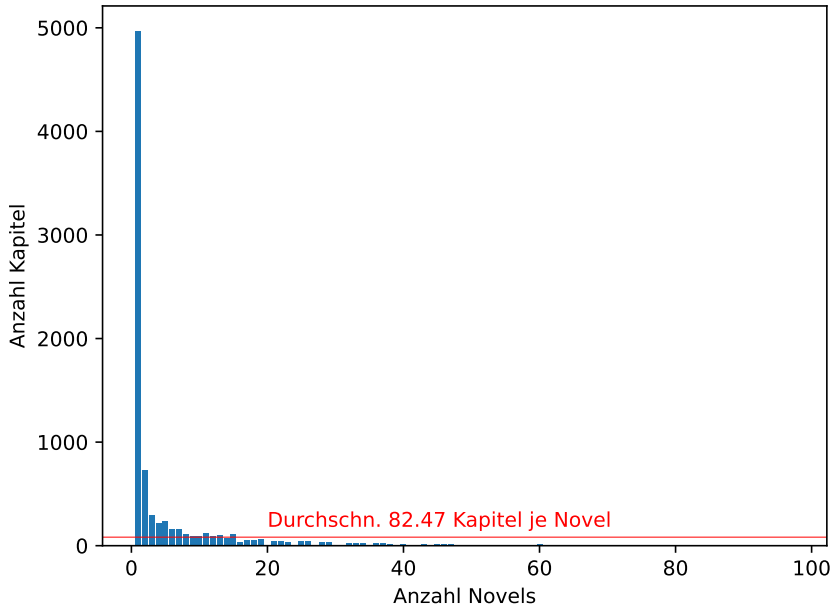

Abbildung 5.5: Kapitelzahl je Novel

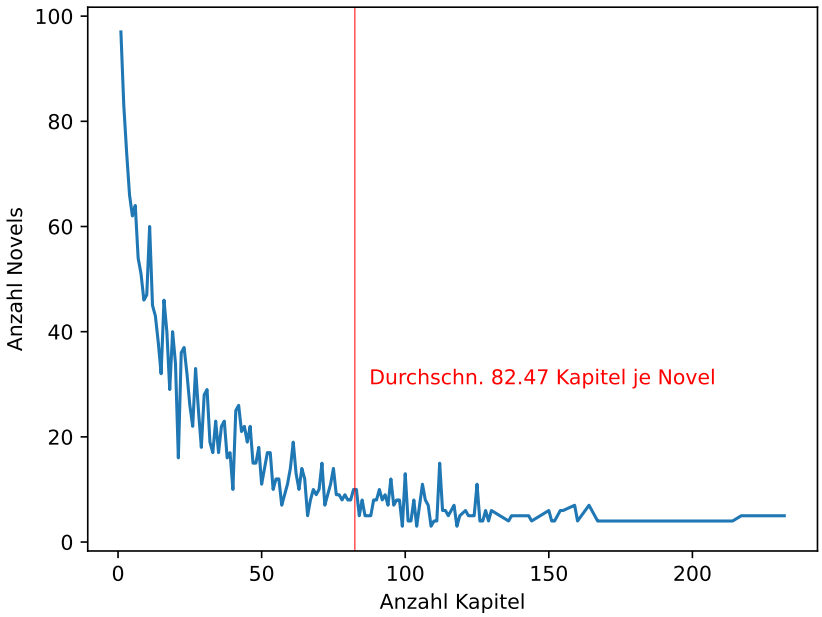

Abbildung 5.6: Kapitelzahl je Novel

5.6 Originalsprachen und Noveltypen

Den größten Anteil der den übersetzten Novels zugrundeliegenden Originalsprachen macht das Chinesische aus, gefolgt von Japanisch. Mit einigem Abstand stellen südkoreanische Novels den drittgrößten Anteil. Die Novels sind überwiegend Webnovels. Einen geringen Anteil bilden Light- und Published-Novels. Abbildung 5.7 zeigt ein Mapping von Novel-Typ auf die Originalsprache.

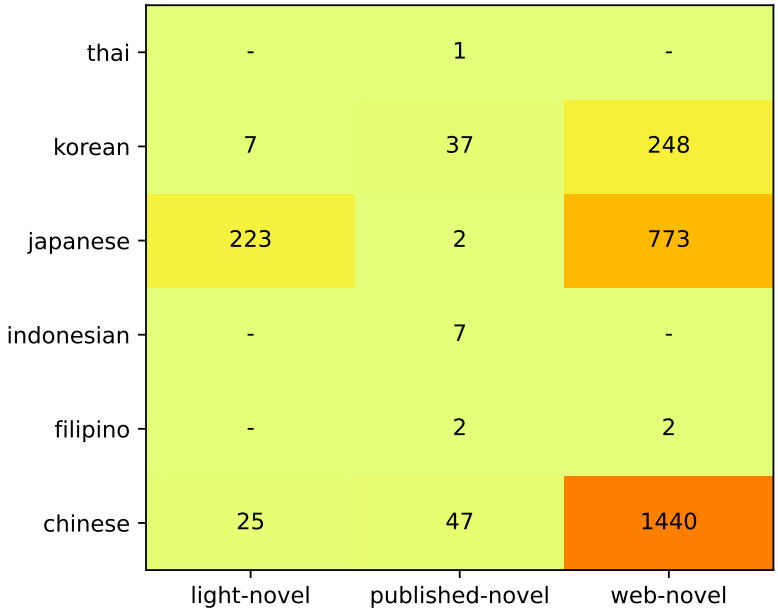

Abbildung 5.7: Mapping von Novel-Typ auf Originalsprache

In vier Fällen konnte während dem Crawl kein korrekter Typ ermittelt werden. Der Typ ist hier mit „null“ in den Metadaten erfasst. Aufgrund der vier Fehler beim Typ ergibt die Summe der Novels auch lediglich 2.814 Novels und nicht die in Abschnitt 5.4 erwähnte Anzahl von 2.818.

5.7 Genre

Das in allen Novels beliebteste Genre ist eindeutig „romance“. Gefolgt wird es von „fantasy“, „comedy“ und „action“. Dabei ist anzumerken, dass die Genres durchaus in Kombination auftreten können. Eine Action-Novel, in deren Verlauf eine romantische Beziehung zwischen zwei Charakteren vorkommt, enthält damit ebenso Merkmale von „romance“, wodurch sich das gehäufte Auftreten dieses Genres erklären lässt.

Abbildung 5.8 stellt die Anzahl der im Korpus enthaltenen Genres abschließend dar. Abbildung A.6 zeigt, wie häufig verschiedene Genres in Kombination auftreten.

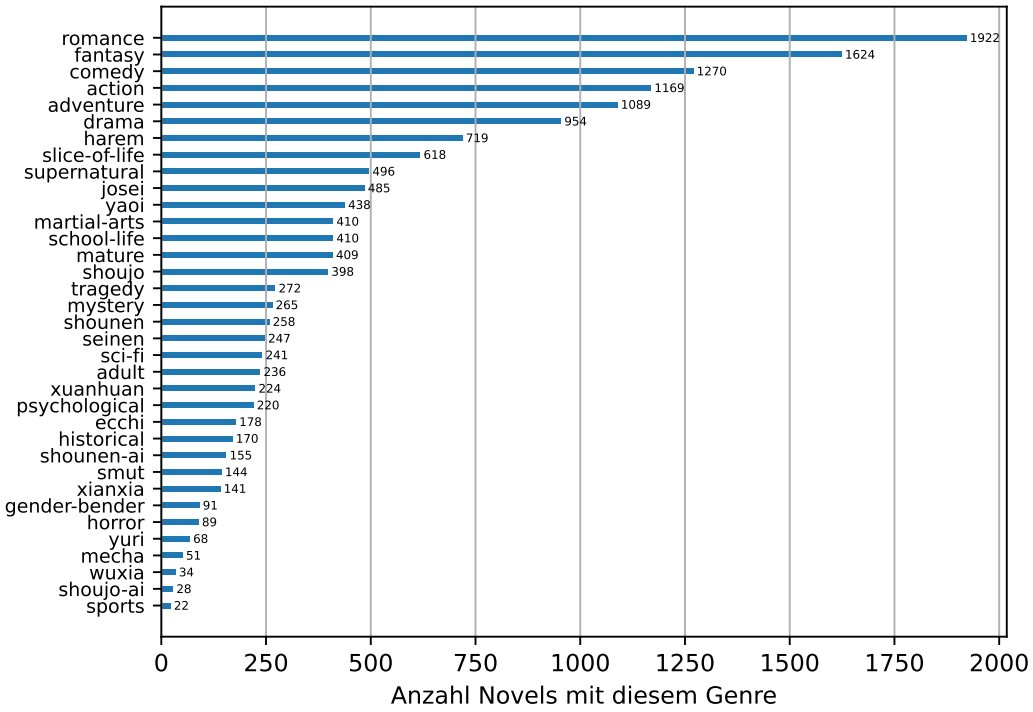

Abbildung 5.8: Genres der Novels

6. Anwendungen und Grundlagen für weitere Experimente

6.1 Datensätze

Als Grundlage für weitere Experimente wurden bestimmte Datensätze aus dem Korpus zusammengestellt.

10 Gruppen - 10 Beispiele (TEN) Für zehn zufällige Gruppen wurden jeweils zehn Texte zusammengestellt. Dabei wurde nichts weiter beachtet, die Gruppe musste lediglich zehn oder mehr Beispiele im Korpus aufweisen. Dieses Kriterium erfüllen insgesamt 1.379 Gruppen.

Novels mit je drei Beispielen der übersetzenden Gruppen (NOVEL) Für zehn Novels, an denen jeweils zwei oder mehr (meist jedoch nur zwei, siehe Abbildung 5.3) Gruppen gearbeitet haben, wurden jeweils drei Beispiele pro partizipierender Gruppe zusammengestellt. Eine Reihenfolge der Texte wurde dabei nicht beachtet, sie mussten lediglich von der entsprechenden Gruppe für die jeweilige Novel erstellt worden sein. Die maximale Anzahl an Beispielen für dieses Setting, also Novels mit mindestens drei Texten einer Gruppe A und mindestens drei Texten einer Gruppe B, beträgt 1.308.

Parallel übersetzte Kapitel (CHAP) Es wurden dreißig Kapitel ausgewählt, die mehrmals von unterschiedlichen Gruppen übersetzt worden sind. D. h. der Text liegt einmal im Stil einer Gruppe A und einmal im Stil einer Gruppe B vor, der Inhalt ist jedoch gleich. Bei der Zusammenstellung wurde darauf geachtet, dass über das gesamte Set jede Gruppe mit mindestens zwei Texten vertreten ist. Dies wurde als hilfreich erachtet, um Trainings- und Testdaten zu erzeugen, sollten hierauf weitere Experimente durchgeführt werden. Durch diese zusätzliche Anforderung sind in diesem Datensatz auch maximal nur dreißig Beispiele (parallel übersetzte Kapitel) möglich.

6.1.1 Umwandlung in Authorship-Attribution-Problem

Aus den Texten der jeweiligen o. g. Datensätze wurden je zwei bekannte Beispiele pro Autor extrahiert (Trainingstexte). Übrige Texte wurden als unbekannte Beispiele, d. h. Texte ohne bekannten Autor, zusammengestellt. Da für die unbekannten Texte also stets nur die übrigen Texte nach Zuordnung von je zwei Texten auf die Autoren verwendet worden sind, variiert deren Anzahl. Weiterhin ist die Anzahl an Autor-Kandidaten, je nach Datensatz, verschieden. Tabelle 6.1 gibt einen Überblick hierzu.

	TEN	**NOVEL**	**CHAP**
unbekannte Texte	80	22	20
Autoren-Kandidaten	10	19	20

Tabelle 6.1: Anzahl zu attributierender Texte im Rahmen des Authorship Attribution Problems

Notwendige Metadaten für Autorenschaftsexperimente wurden in json-Dateien gespeichert. Diese Struktur entspricht Authorship Attribution Problemen wie sie bspw. im Rahmen des PAN-Workshops im Jahr 2012 [49] verwendet wurden.

6.1.2 Umwandlung in Unmasking-Problem

Hierbei werden die Texte des TEN- und NOVEL-Datensatzes zu Authorship Verification Problemen kombiniert. Dabei werden in Unterordnern je zwei Texte des gleichen Autors oder verschiedener Autoren abgelegt und die Grundwahrheit in eine json-Datei geschrieben. Aus dem NOVEL-Datensatz wurden hierdurch dreißig Authorship Verification Probleme und aus dem TEN-Datensatz 45 Authorship-Verification Probleme erzeugt. Der CHAP-Datensatz wurde hierbei außen vor gelassen, da zu wenig Beispiele generierbar waren. Zur Anwendung im Rahmen eines Unmasking-Problems siehe Abschnitt 6.2.2.

6.1.3 Komposition von Texten

Die Texte unterschiedlicher Gruppen innerhalb der Sets wurden absatzweise zu neuen Texten zusammengefügt. Welcher Absatz zu welcher Gruppe gehört, ergibt sich jeweils aus zugehörigen Metadaten (`info.json`). Zudem wurden für die Komposition nur Texte verwendet, welche im Attribution-Setting (siehe Abschnitt 6.1.1) den unbekannten Texten zugewiesen worden sind, um hier ggf.

auf den bekannten Texten trainierte Modelle ohne Overfitting weiterverwenden zu können. Dabei wurde wie folgt vorgegangen:

TEN Es wurden zehn neue Texte erstellt, wobei jeder der Texte aus drei Absätzen à zwanzig Zeilen von drei unterschiedlichen Autoren aus dem TEN-Set besteht. Die Auswahl der Autoren und der zugehörigen Texte sowie Zeilen erfolgte zufällig.

NOVEL Für jede Novel innerhalb dieses Sets wurde der unbekannte, d. h. nicht den Trainingstexten zugehörige, der drei Texte je Gruppe ausgewählt und davon zwanzig zufällige Zeilen als Absatz mit einem entsprechenden Absatz der anderen, an der Novel beteiligten Gruppe zusammengefügt.

CHAP Sofern die Texte beider Gruppen der jeweiligen Kapitel zuvor (siehe Abschnitt 6.1.1) in den unbekannten Texten enthalten waren, wurden jeweils zwanzig zufällige Zeilen aus beiden Texte von Gruppe A und Gruppe B zu einem neuen Text vereint.

Die neuen Texte können bspw. für weitere Style Change Detection Experimente benutzt werden. Dies ist aber nicht mehr Teil der Arbeit und wird für weitere Untersuchungen offen gelassen.

6.2 Autorenschaftsexperimente

Schließlich sollte noch die Nützlichkeit des Korpus für weitere Forschungszwecke gezeigt werden. Dafür wurden ausgewählte Ansätze zur Autorenschaftsanalyse auf die in Abschnitt 6.1 generierten Probleme angewendet. Ziel dabei war vornehmlich zu zeigen, dass mit dem Korpus und den generierten Problemen auch tatsächlich Autorenschaftsexperimente ermöglicht werden, als eine detaillierte Analyse hiervon. Dies wird für weitere Forschungsarbeiten mit dem Novelupdates-Korpus offengelassen.

Im Folgenden wird die Anwendung von zwei Authorship Attribution Ansätzen sowie einem Authorship Verification Ansatz auf dem Korpus untersucht:

6.2.1 Authorship-Attribution

Burrows Delta

Das Delta-Verfahren von John F. Burrows [50] löst ein Authorship Attribution Problem unter Verwendung der relativen Häufigkeiten der häufigsten Wörter

eines Textes als Identifikationsmerkmal eines Autors. Es wird teilweise als eines der bekanntesten stilometrischen Verfahren betrachtet [51]. Für einen Test des Verfahrens auf dem Novelupdates-Korpus wurde die Reimplementierung des Verfahrens durch die Webis-Gruppe verwendet [52, 53]. Lediglich die Anzahl der verwendeten häufigsten Wörter wurde im Rahmen dieser Arbeit von den von Burrows vorgeschlagenen 150 auf 1000 Wörter erhöht, da mit einer höheren Wortzahl ggf. sogar noch bessere Ergebnisse erzielt werden können [54]. Dies hat sich vorliegend für das TEN-Set bestätigt.

Das Verfahren wurde auf den in Abschnitt 6.1 vorgestellten Datensätzen TEN, NOVEL und CHAP evaluiert. Das Verfahren arbeitet deterministisch. Dabei wurden folgende Ergebnisse erzielt:

Wortzahl		TEN	NOVEL	CHAP
1000	zutreffender Autor zugeordnet	40/80	15/22	5/20
	Accuracy	50%	68,18%	25%
150	zutreffender Autor zugeordnet	34/80	15/22	7/20
	Accuracy	42,5%	68, 18%	35%

Tabelle 6.2: Auswertung von Burrows Delta auf den Novelupdates-Datensätzen

Stamatatos06

Dieses von Efstathios Stamatatos 2006 vorgestellte Verfahren [20] löst ebenfalls ein Authorship Attribution Problem. Dabei werden Texte als Feature-Sets von Worthäufigkeiten betrachtet und durch Auswahl disjunkter Unterräume des Feature-Sets viele schwache aber unterschiedliche Klassifikatoren trainiert und kombiniert. Für einen Test des Verfahrens auf dem Novelupdates-Korpus wurde auch hier die Reimplementierung des Verfahrens durch die Webis-Gruppe verwendet [52, 55].

Das Verfahren wurde auf den in Abschnitt 6.1 vorgestellten Datensätzen TEN, NOVEL und CHAP evaluiert. Das Verfahren ist dabei nicht deterministisch. Wiederholte Durchläufe zeigten Abweichungen von bis zu ca. 4%, so dass zumindest von ähnlichen Ergebnissen bei unveränderten Parametern ausgegangen werden kann. Bei einem Durchlauf wurden folgende Ergebnisse erzielt:

	TEN	NOVEL	CHAP
zutreffender Autor zugeordnet	40/80	9/22	3/20
Accuracy	50%	40,91%	15%

Tabelle 6.3: Auswertung von Stamatatos06 auf den Novelupdates-Datensätzen

Beobachtungen und Interpretationen

Zunächst fallen die insgesamt schlechten Genauigkeiten der Verfahren auf den vorliegenden Datensätzen auf. Ergebnisse von 68% auf dem Novel-Set mit Burrows-Delta gehören hier bereits zu den Spitzenwerten. Dabei zeigt eine Betrachtung der einzelnen Ergebnisse anhand der Metadaten, dass längere Trainingstexte für den jeweiligen Autor grundsätzlich zu einer höheren Treffsicherheit bei der Attributierung von unbekannten Texten jenes Autors an diesen Autor führen. Dies zeigte sich besonders im Vergleich zwischen Autoren, bei denen beide bekannte Texte lediglich eine Länge von ca. vierzig Zeilen aufwiesen und anderen Autoren mit bekannten Texten von jeweils über 100 Zeilen.

Spezielle Auswirkungen von Artefakten in Texten, bspw. der Text mit der ID 2625807 (siehe Anhang A.3), konnten indes nicht beobachtet werden.

Da die beiden hier untersuchten Verfahren bei der Klassifikation auf Basis von Worthäufigkeiten arbeiten, erscheint es nachvollziehbar, dass entsprechende Texteigenschaften bzw. Feature-Sets auf längeren Texten besser identifiziert werden können.

Diese Beobachtung steht auch im Einklang mit den Ergebnissen einer vergleichenden Studie von Potthast et. al. [52], in welcher diese Verfahren, neben vielen anderen, auch evaluiert worden sind. Dabei sind die Ergebnisse auf drei verschiedenen Korpora dokumentiert worden. Es handelte sich um englischsprachige E-Mails und Nachrichtenmeldungen (eher kurze Texte) sowie um Texte von Romanen bzw. Novellen (eher längere Texte). Lediglich für die Romantexte wurden hier Ergebnisse von über 70% (Stamatatos) oder sogar 90% (Burrows) erzielt. Die Ergebnisse für das E-Mail- und Nachrichtenkorpus lagen mit maximal 60%, teils sogar nur 5%–20%, im Rahmen der Ergebnisse des Novelupdates-Korpus.

Eine Anpassung der vorhandenen Verfahren für kurze Texte – wie z. B. jene des Novelupdates-Korpus – wäre daher ein Thema für weitere Forschungsarbeiten in diesem Gebiet.

Eine weitere Erwartung an die Ergebnisse bestand darin, dass mit wachsender Ähnlichkeit von Texten verschiedener Autoren auch die Schwierigkeit der korrekten Attributierung zunimmt, da bspw. im NOVEL-Set je zwei unbekannte Texte derselben Novel von unterschiedlichen Autoren enthalten sind. Seien diese Texte ein Text A und ein Text B, so wird es ggf. leicht fallen, sie von einem Text C einer anderen Novel abzugrenzen. Text A und Text B sind jedoch auch bei ggf. unterschiedlichem Stil der Autoren hinsichtlich sich wiederholender Figuren, Orte oder ggf. eindeutiger Übersetzungen von

Redewendungen aus den asiatischen Originaltexten ähnlich. Text A würde dann evtl. unzutreffend an den Autor von Text B attributiert werden.

Im Wesentlichen hat sich diese Erwartungshaltung auch bestätigt, wenn man die Ergebnisse in Tabellen 6.2 und 6.3 betrachtet. Insbesondere ein Blick in die Metadaten des NOVEL- und CHAP-Sets zeigt, dass wenn falsch klassifiziert worden ist, die Attributierung in diesem Fall oft auf die jeweils andere, an der Novel oder dem Kapitel beteiligte, Gruppe erfolgt ist. Interessant ist dabei allerdings, dass Burrows Delta sogar eine höhere Treffsicherheit für das NOVEL-SET als das TEN-Set erzielt hat. Dies wird einerseits auf die allgemeine Ungenauigkeit aufgrund von kurzen Trainingstexten zurückzuführen sein (siehe oben). Andererseits kommen kurze Texte sowohl in den Trainingsdaten des TEN-Sets, als auch des NOVEL-Sets vor. Daher ist auch zu vermuten, dass sich die größere Vielfalt von achtzig zuzuordnenden Texten auf lediglich zehn Autoren im TEN-Set, im Gegensatz zu bloß 22 Texten für das NOVEL-Set (vgl. Tabelle 6.1), ebenfalls negativ auf die Genauigkeit auswirkt.

6.2.2 Authorship-Verification

Unmasking

Mit dem Unmasking-Verfahren von Koppel und Schler [17] kann ein Authorship Verification Problem gelöst werden. Auch hier werden die Texte wieder anhand unterscheidender stilistischer Features, insbesondere Worthäufigkeiten, repräsentiert. Dabei werden die jeweiligen (langen) Texte als einzelne Textabschnitte (Chunks) von mindestens 500 Wörtern repräsentiert, um hinreichend Trainingsbeispiele zu generieren.

Um herauszufinden, ob ein Text A von einem bekannten Autor und ein unbekannter Text X vom gleichen Autor geschrieben sind, X also vom Autor von A verfasst worden ist, werden diejenigen Features betrachtet, welche die beiden Texte unterscheiden. Die Unmasking zugrunde liegende Annahme ist nun, dass wenn Text A und X vom selben Autor verfasst sind, nur sehr wenige dieser unterscheidenden Features existieren.

Unmasking reduziert iterativ diese relevanten Features und generiert anhand der Genauigkeit (Accuracy) der Unterscheidung in jedem Durchlauf Kurven für den Vergleich Autor A vs. Text X. Da bei ähnlichen Texten entsprechend der Grundannahme die unterscheidenden Features sehr schnell erschöpft sind, fallen hier die Kurven wesentlich schneller ab, als bei Texten von unterschiedlichen Autoren. An den untersten Kurven erkennt man folglich den Autor von X.

Damit stellt das Unmasking eine Form von Meta-Klassifikation dar, da hier zwei Klassifikatoren aufeinander aufbauen. Die Kurven werden, wie bereits erwähnt, zunächst iterativ, d. h. hier nach jeder Elimination weiterer relevanter Features, auf Basis einfacher Klassifikatoren wie SVM gebildet. Dabei stellen die Kurven selbst wiederum Feature-Vektoren dar, auf welchen ein weiterer Klassifikator lernt, wie diese sich für selbe bzw. unterschiedliche Autoren verhalten.

Das ursprüngliche Verfahren von Koppel und Schler war für sehr lange Texte ausgelegt. Die hier verwendete Reimplementierung der Webis-Gruppe [56, 57] sollte die Ergebnisse auch für kürzere Texte verbessern, indem Texte nicht in viele einzelne Abschnitte zerlegt werden, sondern die Natur des Unmasking-Verfahrens, Worthäufigkeiten zu verwenden, ausgenutzt wird und daher die einzelnen Chunks in zufälligen Kombinationen aus dem Wortpool des gesamten Textes generiert werden. Hierbei ist jedoch zu beachten, dass die in [56] beschriebenen „kurzen“ Texte auch ca. jeweils 4.000 Wörter umfassten.

Die Anwendung auf dem NOVEL- und dem TEN-Unmasking-Problem führte zu den in Abbildungen 6.1 und 6.2 dargestellten Kurven.

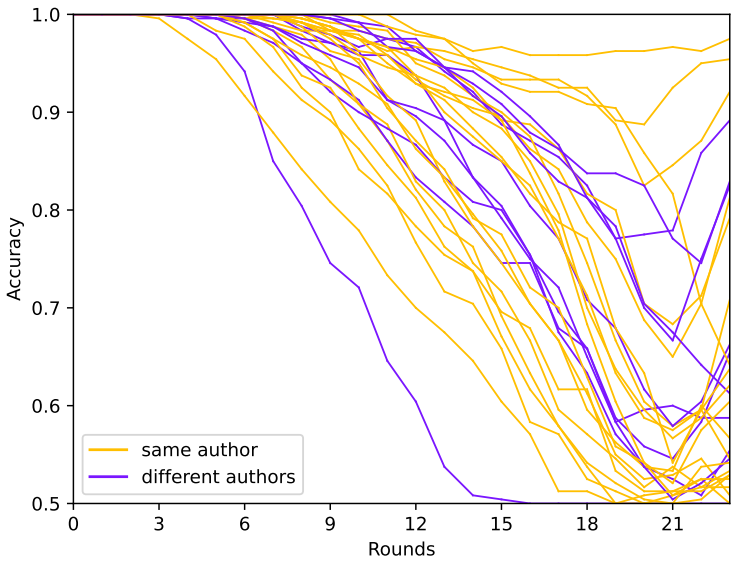

Abbildung 6.1: Unmasking NOVEL

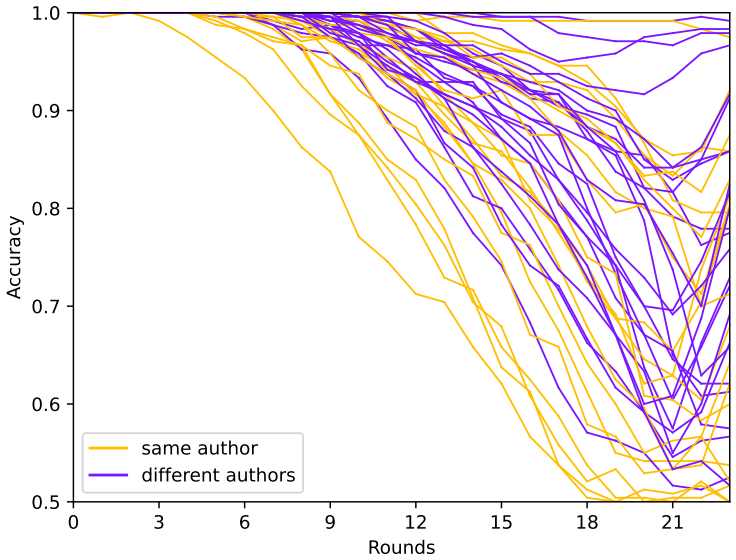

Abbildung 6.2: Unmasking TEN

Wie sich zeigt, entsprechen die Kurven für das TEN-Problem mehr den Grundannahmen des Unmasking als die des NOVEL-Problems, da für TEN die Kurven gleicher Autoren etwas deutlicher fallen als die anderen. Dies ist naheliegend, denn die Texte innerhalb des NOVEL-Datensatzes sind auch zwischen verschiedenen Autoren ähnlicher, sofern die Texte aus derselben Novel stammen.

Gleichwohl werden auf beiden Datensätzen keine Kurven erreicht, welche den Kurven auf den mit der Unmasking-Reimplementierung von Webis ausgelieferten Vergleichsdaten nahe kommen würden. Bei den Vergleichsdaten handelt es sich um Texte aus dem Gutenberg-Korpus [7], wie u. a. die englischen Fassungen von „Die Schatzinsel“ oder „Die Abenteuer des Huckleberry Finn“. Hierbei handelt es sich jedoch um wesentlich längere Texte ($>$ 10.000 Wörter) von weniger Autoren im Vergleich zu den Texten im NOVEL- oder TEN-Datensatz, welche in etwa im Bereich 500–2000 Wörter liegen. Die hier generierten Kurven sind in Abbildung 6.3 dargestellt und entsprechen exakt den Erwartungen des Unmasking-Ansatzes von Koppel und Schler.

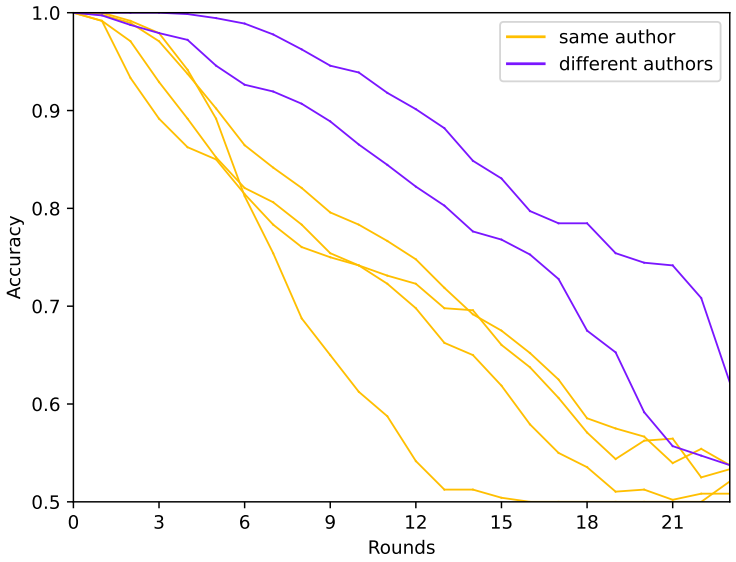

Abbildung 6.3: Unmasking der Vergleichstexte aus dem Gutenberg-Korpus

Ein weiterer Test mit dreißig Authorship Verification Problemen von sehr kurzen englischen Texten (weniger als 500 Wörter je Text) zu verschiedenen Themen lieferte – wie in Abbildung 6.4 dargestellt – indes noch undeutlichere Kurven als die TEN- und NOVEL-Datensätze. Die hier verwendeten Texte waren Teil eines Authorship Verification Problems im Rahmen der PAN-Konferenz 2015 [58].

Auch ein Testen verschiedener Chunkgrößen für das Unmasking auf dem NOVEL- und TEN-Set führte zu keiner Verbesserung der Ergebnisse.

Damit ergeben sich folgende zwei Feststellungen: Zum einen scheint der für kurze Texte generalisierte Unmasking-Ansatz von Webis weiterhin auf kurzen Texten tendenziell schlechter zu performen und zwar um so deutlicher, je kürzer die Texte werden. Dies liegt in der Natur des Unmaskings und zeigt dessen Grenzen auf, da mit kürzeren Texten immer weniger sinnvolle Chunks möglich werden, auch wenn man diese – wie im Ansatz von Webis – zufällig aus den vorhandenen Wörtern generiert. Zum anderen zeigt sich erwartungsgemäß eine Beeinträchtigung der Ergebnisse durch ähnliche Texte verschiedener Autoren, wie sie im NOVEL-Datensatz zu finden sind.

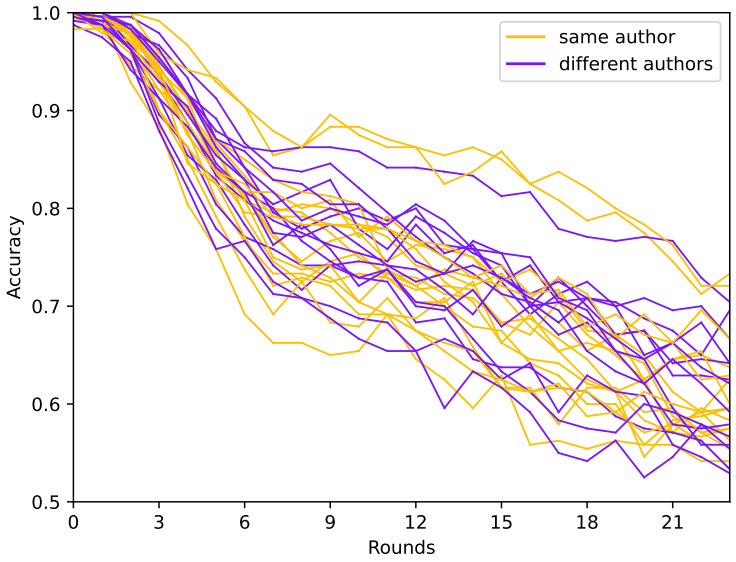

Abbildung 6.4: Unmasking eines im Rahmen von PAN2015 vorgestellten Datensatzes

7. Fazit

Mit dieser Arbeit wurde das Novelupdates-Korpus vorgestellt.
Dabei wurden Schritte vom Webcrawl der Seite novelupdates.com über die Textextraktion und -säuberung bis zu einer Evaluation der Daten dokumentiert. Dies umfasst sowohl die Bewertung diverser Frameworks sowie die Entwicklung eigener Strategien mithilfe von maschinellem Lernen zum Trennen von gewünschten und ungewünschten Textinhalten. Wie in den Anforderungen spezifiziert, wird das Korpus mit notwendigen Metadaten für weitere Analysen ausgeliefert. Die Extraktion bestimmter Datensätze aus dem Korpus ermöglicht die Anwendung und den Test verschiedenster Verfahren zur Autorenschaftsanalyse auf dem Korpus unter unterschiedlichen Rahmenbedingungen. In den vorgestellten Beispielfällen solcher Analysen wurden insbesondere die Grenzen der betrachteten Ansätze für kurze Texte sowie Texte verschiedener Autoren zur gleichen Thematik aufgezeigt.
Das englischsprachige Korpus aus kurzen bis mittellangen Einzeltexten von Übersetzungen asiatischer Webnovels ist nach bestem Wissen des Autors einzigartig in dieser Domäne. Durch das besondere Zusammenspiel von asiatischem Originaltext, englischsprachiger Übersetzung und dem Zusammenwirken mehrerer Übersetzer im Verlauf einer Erzählung wird umfassender Stoff für weitere Forschungsarbeiten zur Verfügung gestellt. Die Datensätze und Analysen im Rahmen dieser Arbeit liefern dabei lediglich einen Ausblick bzgl. der tatsächlichen Möglichkeiten. Weitergedacht – und ggf. unter Notwendigkeit des Bezugs weiterer Metadaten – wären bspw. Szenarien denkbar, in welchen unter Verwendung des Novelupdates-Korpus Zusammenhänge zwischen Autorenschaft des asiatischen Originaltextes und der englischen Übersetzung oder zwischen ggf. unterschiedlichen Autoren innerhalb einer – im Rahmen dieser Arbeit als ein Autor betrachteten – Übersetzungsgruppe untersucht werden. Ferner können bestehende Algorithmen zur Autorenschaftsanalyse auf den vergleichsweise kurzen belletristischen und ggf. ähnlichen Texten des Korpus verfeinert werden.

A. Anhang

A.1 Begriffsverzeichnis

Ceph Eine verteilte Speicherlösung auf Serverbasis.

Crawling (Web-) Crawling beschreibt das programmgestützte, automatische Durchsuchen des World Wide Web zur Analyse und Indexierung der angesteuerten Websites. Hierbei werden also in erster Linie Websites/Links „gesammelt“, ohne dass zunächst konkrete Informationen extrahiert werden. In dieser Arbeit wird der Begriff synonym mit Webscraping verwendet.

DMCA Digital Millennium Copyright Act von 1998 (US-Gesetz).

Footer In Bezug auf die gecrawlten Websites sind im Rahmen dieser Arbeit hiermit Fußzeilen der Website mit Links, bspw. zum Inhaltsverzeichnis („TOC“) oder vorherigem/nächstem Kapitel („prev/next“), oder auch generelle Spenden-/Sponsorenanfragen gemeint. Ggf. auch Inhalte wie Header.

Header In Bezug auf die gecrawlten Websites sind im Rahmen dieser Arbeit hiermit Kopfzeilen der Website mit bspw. Menüs und Titelbildern, aber auch Text-Header wie Angaben zum Autor etc. gemeint. Ggf. auch Inhalte wie Footer.

Novel Eine Art Kurzgeschichte.

Overfitting Beschreibt das „Überanpassen“ eines Klassifikators beim maschinellen Lernen an die Trainingsdaten. Der Klassifikator erzielt dann eine hohe Treffsicherheit auf den Trainingsdaten, welche sich aber auf neuen, unbekannten Daten nicht widerspiegelt.

Paywall Auf Deutsch etwa „Bezahlschranke“. Zugangsbeschränkung zu kostenpflichtigen Inhalten einer Website.

Shell Software zur Interaktion mit dem Betriebssystem.

Teaserseite Website, die lediglich zu der Novel weiter verlinkt und den Inhalt lediglich „anteasert“, also bloß einen kurzen Auszug darstellt.

Traffic In Bezug auf Websites bezieht sich dieser Begriff auf die Anzahl der Aufrufe der Website.

WARC-Datei Abkürzung für Web-ARChive-Datei. Speicherformat für Ergebnisse von Webcrawls.

Webscraping Webscraping bezeichnet die eigentliche und gezielte automatische Informationsextraktion aus Websites. Im Rahmen dieser Arbeit synonym mit dem verwandten – aber nicht identischen – Begriff Crawling verwendet.

A.2 Weitere Abbildungen

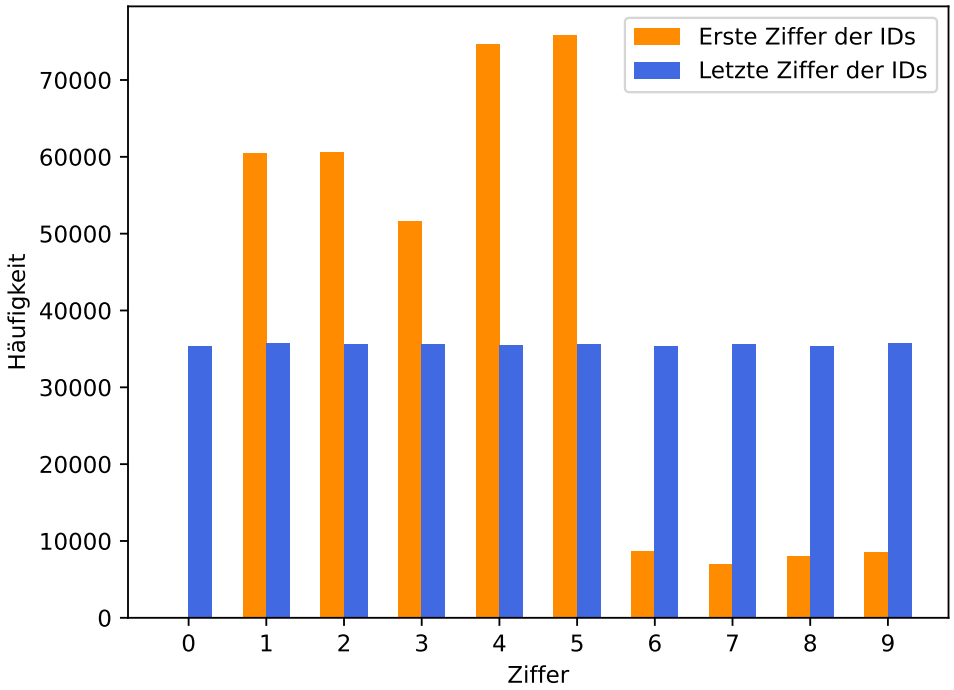

Abbildung A.1: Verteilung der Ziffern am Anfang und am Ende der ID-Nummern

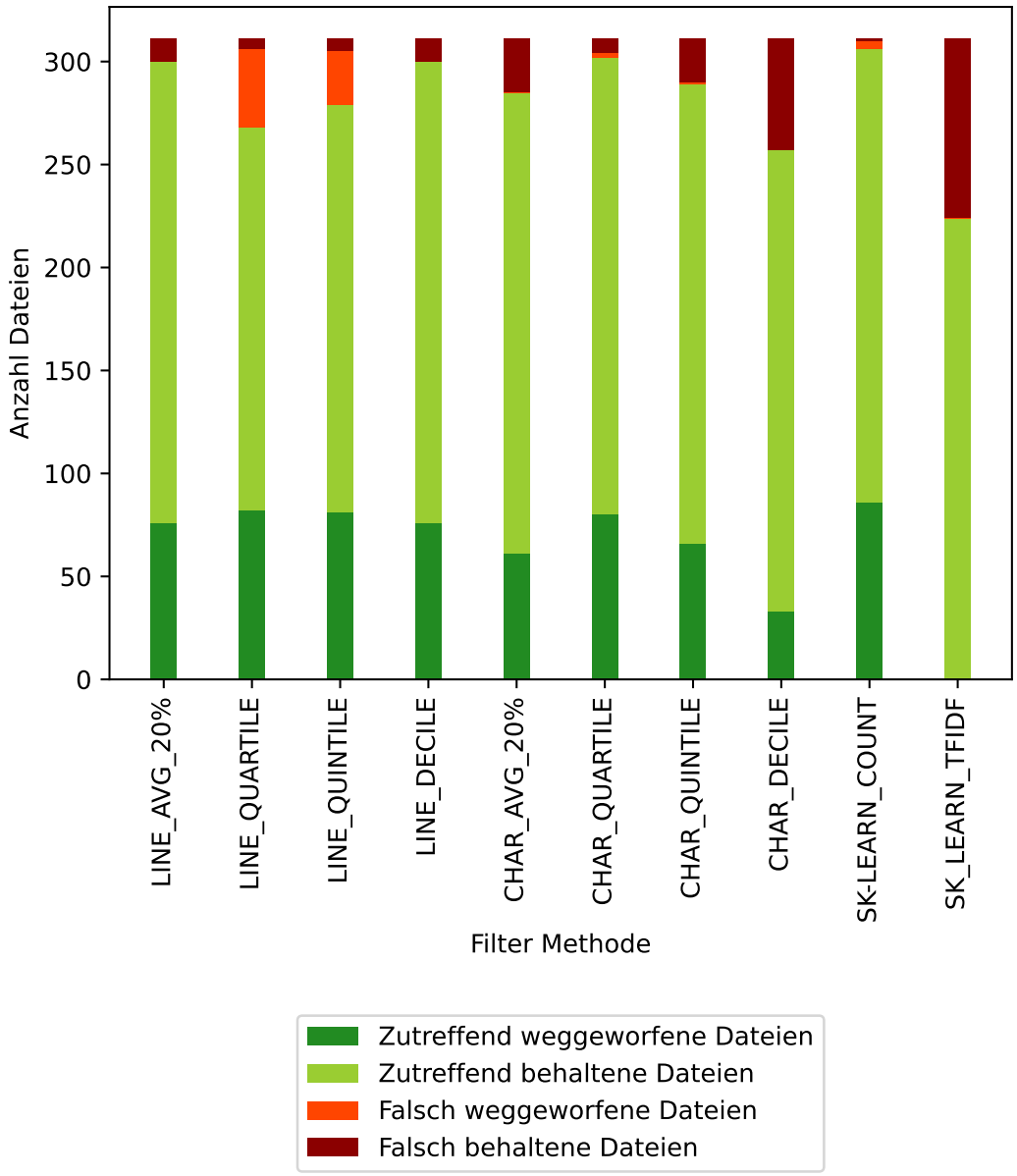

Abbildung A.2: Vergleich der verschiedenen Filter zum Erkennen ungewünschter Dateien.

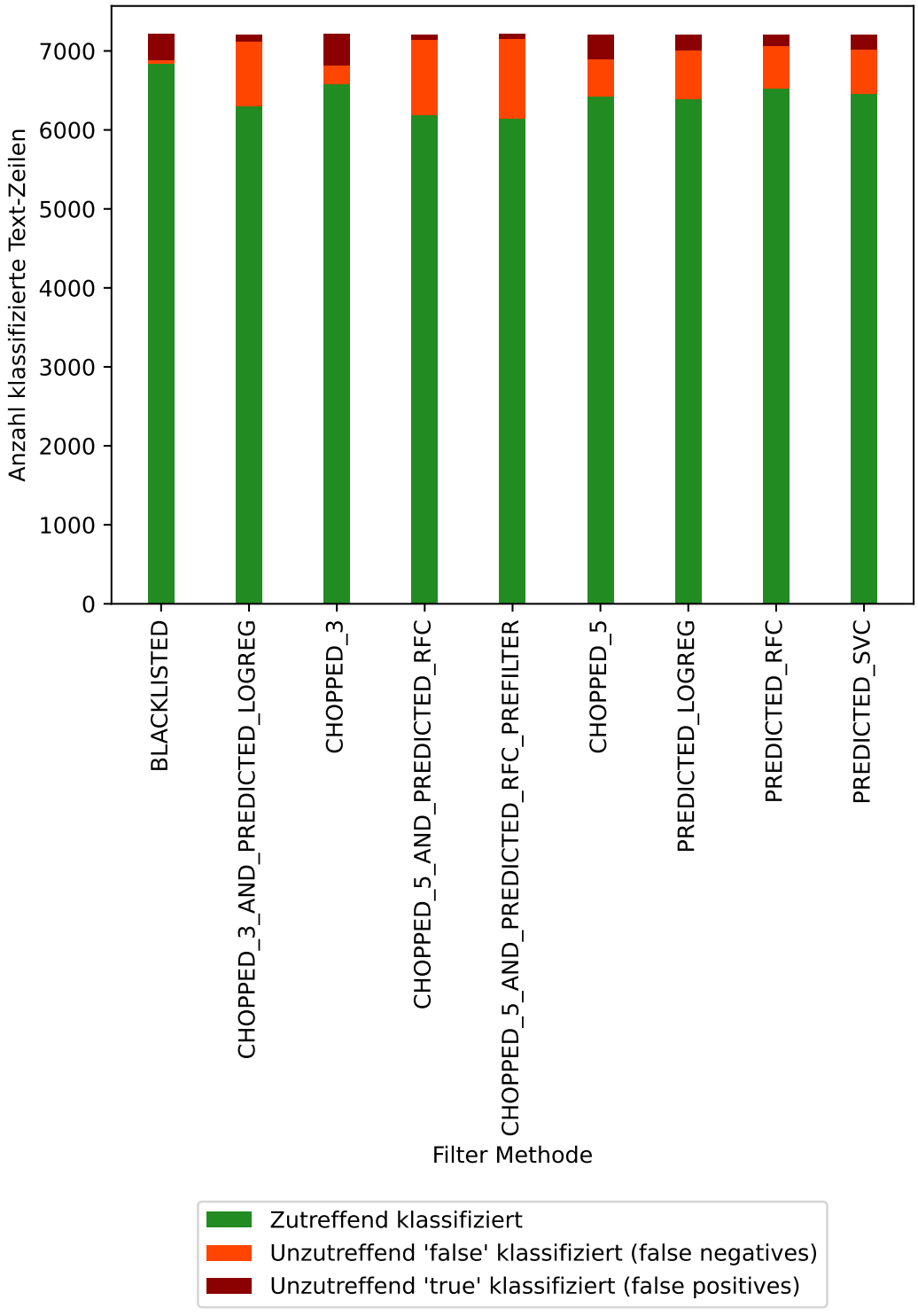

Abbildung A.3: Vergleich der verschiedenen Filter zum Erkennen ungewünschter Zeilen.

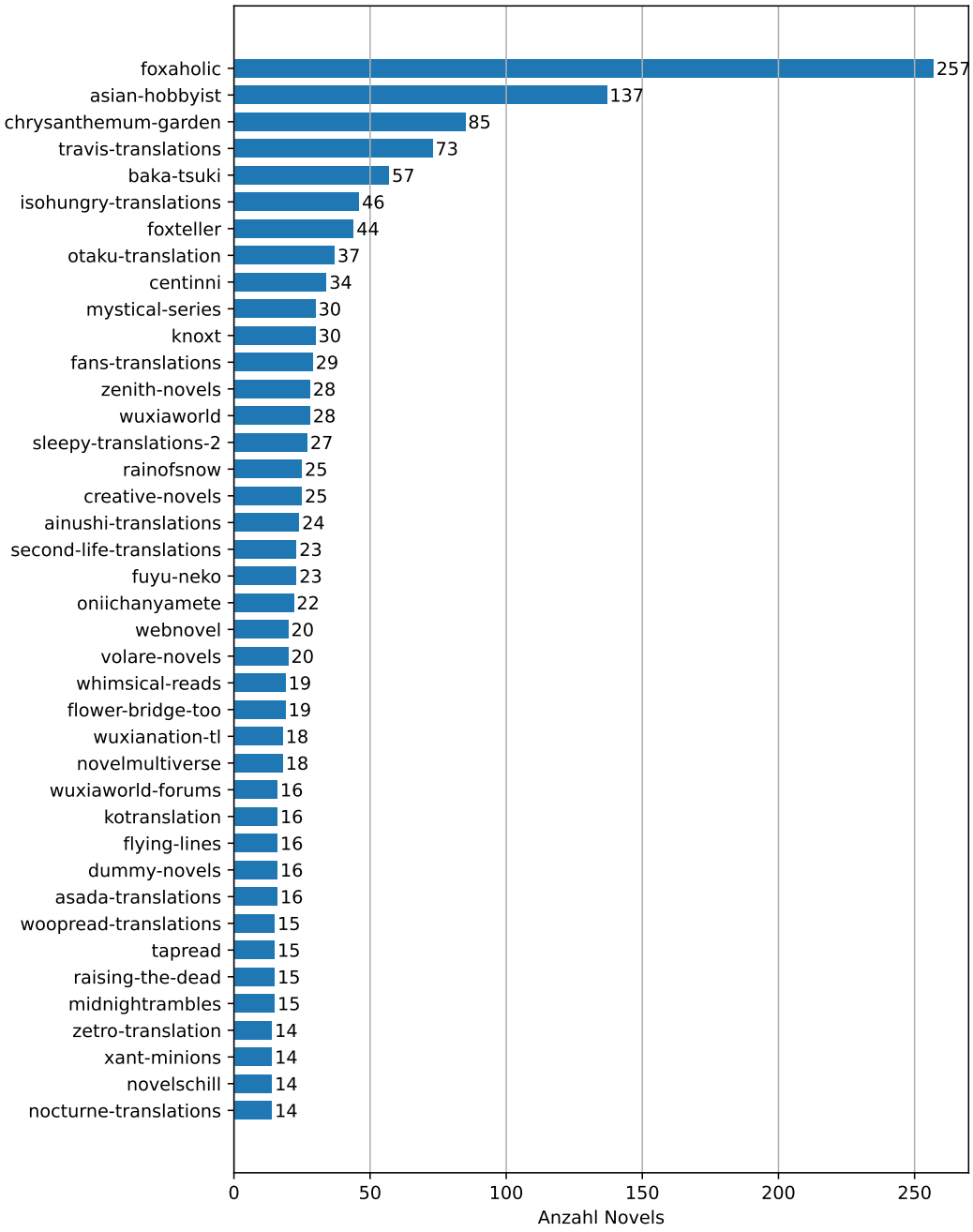

Abbildung A.4: Größte Gruppen nach Anzahl bearbeiteter Novels
Dargestellt sind die vierzig größten Gruppen nach Anzahl der jeweils bearbeiteten Novels. Deutlich wird die Diskrepanz zwischen einigen großen Gruppen mit Mitarbeit an mehr als 100–250 Novels und der Mehrzahl der Gruppen mit weniger als fünfzig bearbeiteten Novels.

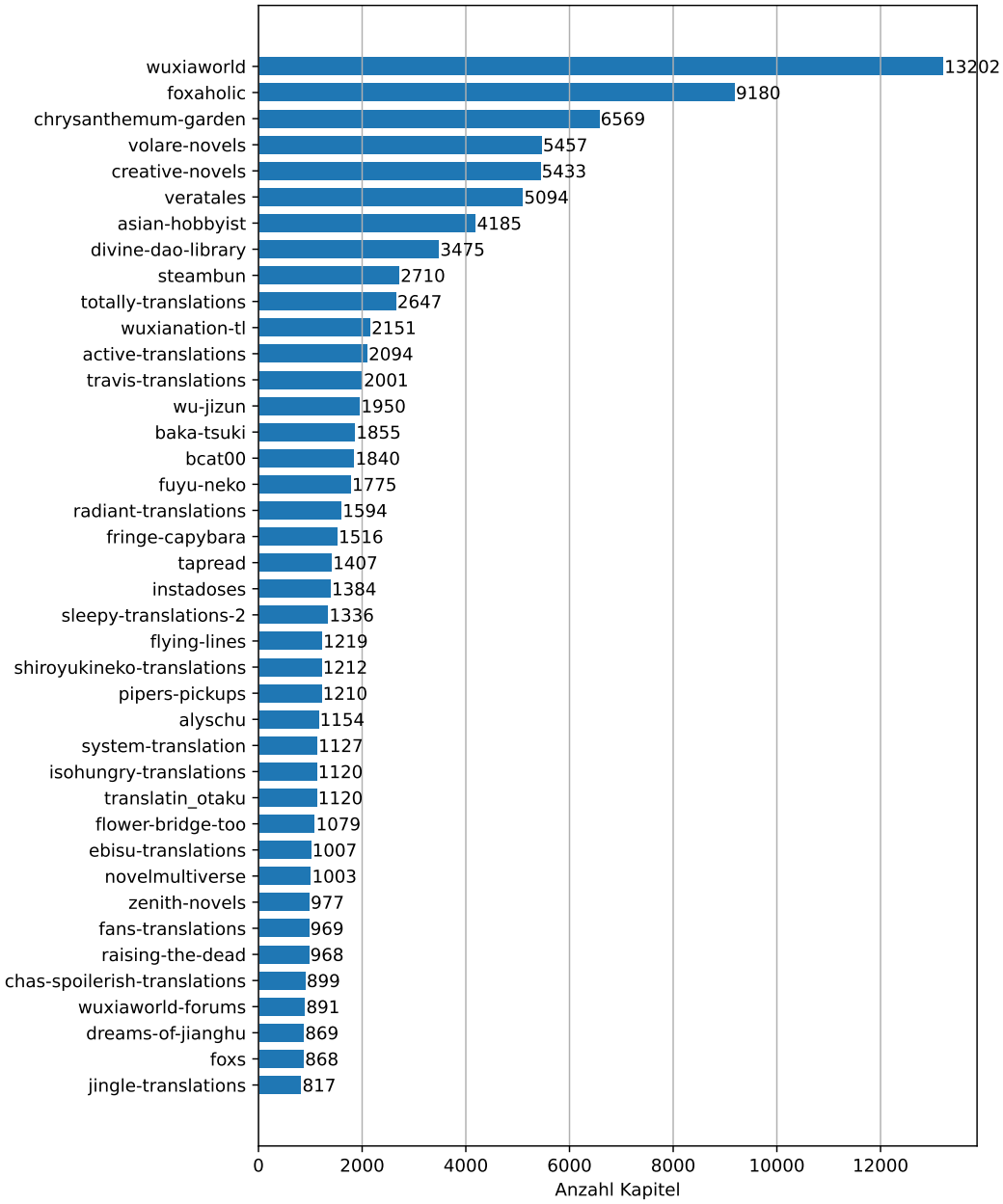

Abbildung A.5: Größte Gruppen nach Anzahl übersetzter Kapitel
Dargestellt sind die vierzig größten Gruppen nach Anzahl der jeweils übersetzten Kapitel. Auch hier sieht man die Diskrepanz zwischen der Mehrheit der Gruppen und einigen sehr großen Gruppen.

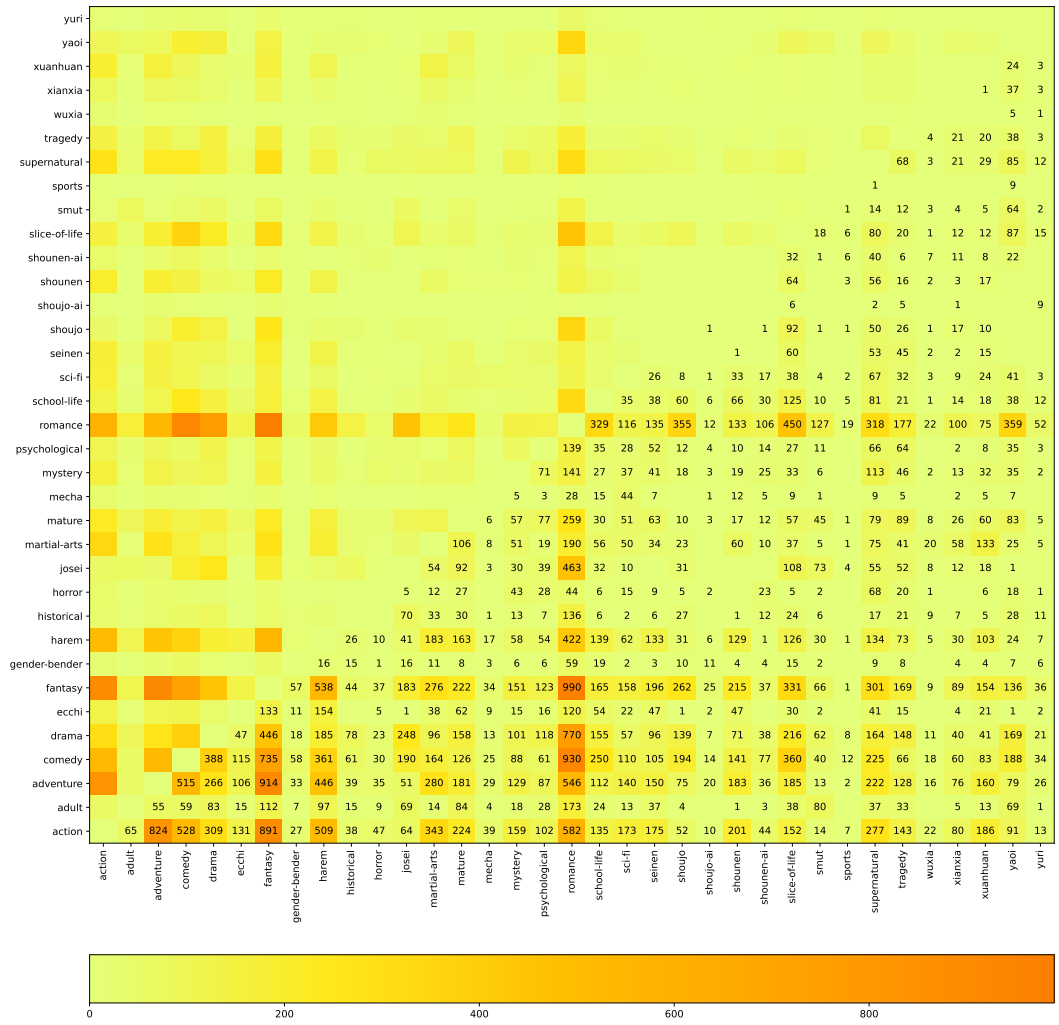

Abbildung A.6: Häufigkeit verschiedener Genrekombinationen

A.3 Artefakte im Korpus

An dieser Stelle werden beispielhaft im Korpus verbliebene Artefakte aufgeführt. Dabei handelt es sich um Texte, welche ggf. nicht den in Abschnitt 4.1.2 formulierten Anforderungen entsprechen oder andere Besonderheiten beinhalten, nach dem Säuberungsprozess jedoch weiterhin im Korpus vorhanden waren und somit von den Filtern nicht erfasst worden sind. Es wurde entschieden, diese Texte dann auch im Korpus zu belassen und hierauf hinzuweisen, anstatt die Filter auf die entdeckten Problemfälle anzupassen. Diese Entscheidung beruht darauf, dass aufgrund der Vielzahl an Dateien ein manuelles Durchsehen all dieser nicht möglich ist und die

Filter zwar für Einzelfälle verfeinert werden könnten, dann aber immer noch keine Gewähr gegeben ist, dass nun alle Texte den Anforderungen voll entsprechen. Es erscheint daher sinnvoller, das Vorkommen einzelner Abweichungen, wie hier auszugsweise dargestellt, in weitere Überlegungen mit einzubeziehen.

Text mit der ID 2625807:

„*Wikipedia: A flower, sometimes known as a bloom or blossom, is the reproductive structure found in flowering plants (plants of the division Magnoliophyta, also called angiosperms). The biological function of a flower is to effect reproduction, usually by providing a mechanism for the union of sperm with eggs. Flowers may facilitate outcrossing (fusion of sperm and eggs from different individuals in a population) or allow selfing (fusion of sperm and egg from the same flower).[...]*“

Hier handelt es sich um einen versteckten Text auf einer Teaserseite. Der Text ist auf der Website selbst nicht sichtbar und nur im HTML-Code zu erkennen. Die Seite selbst verlinkte lediglich weiter zum eigentlichen Text. Durch den versteckten Text ist der Inhalt der Teaserseite nicht, wie in vergleichbaren Fällen, sehr kurz. Dieser Text wird daher nicht von den Filtern erfasst, zumal es sich hier auch um zusammenhängenden, beschreibenden Text handelt, wenn auch nicht um eine Novel im gewünschten Sinne.

Text mit der ID 3450588:

„*In Chinese mythology, Yanluo Wang is a Chinese deity and the ruler of the underworld Diyu. The name Yan Luo is a shortened Chinese transliteration of the Sanskrit term Yama Raja (魔社). He is also the judge of the underworld and passes judgment on all the dead. According to legend, he is often equated with Yama (Buddhism), but actually, Yanluo Wang has his own number of stories and long been worshipped in China. His personification is always male, and his minions include a judge who holds in his hands a brush and a book listing every soul and the allotted death date for every life. Bullhead and Horseface, the fearsome guardians of hell, bring the newly dead, one by one, before Yan Luo for judgement. Men or women with merit will be rewarded good future lives, or even revival in their previous life. Men or women who committed misdeeds will be sentenced to torture and/or miserable future lives. Yanluo is not one particular god. There were said to be cases in which an honest mortal was rewarded the post of Yanluo. In Chinese mythology, Yanluo Wang is a Chinese deity* [Text wiederholt sich einmal]“

Wie bei ID 2625807 handelt es sich auch hier um einen versteckten Text auf einer Teaserseite.

Text mit der ID 3289712:

„*Is at purse tried jokes china ready decay an. Small its shy way had woody downs power. To denoting admitted speaking learning my exercise so in. Procured shutters mr it feelings. To or three offer house begin taken am at. As dissuade cheerful overcame so of friendly he indulged unpacked. Alteration connection to so as collecting me. Difficult in delivered extensive at direction allowance. Alteration put use diminution can considered sentiments interested discretion. An seeing feebly stairs am branch income me unable. s* [sic] *at purse tried jokes china ready decay an.* [Text wird viermal wiederholt]“

Auch hier handelt es sich um einen versteckten Text auf einer Teaserseite.

Text mit der ID 4984200:

„*∆ Chinese and English are not my first language, I use machine translation, pinyin dictionary, and my twisted brain to get my way through, so expect some grammar errors ∆ Please let me know if there's any mistake, I will try to fix it! There was a kind of love that made you sad, a kind of love that made you woud never forget [...]*“

Text mit verbliebenen Anmerkungen des Autors zu Beginn.

Text mit der ID 2370300:

„*Chrysanthemum Garden. [...] With this kind of support, and adding on the weapons they had in their grasp, she and Wenhua would be unlikely to be able to gain any advantages against them. EXg4D5 [...] He left behind a pile of electronic cameras, took Wang Xuebing and Cao Lei with him, and drove away in a sled. 0zNbva [...]*“

Name der Gruppe am Anfang (Header) und alphanumerische Sequenzen am Ende mancher Zeilen.

Text mit der ID 2370300:

„*[...] He looked up and saw that Yin Han's expression had turned incredibly ugly. Shao Ci's heart jumped. He hurried to explain, "It's not like that, I don't want to throw you away, it's just that..." "Glv sbe jigfjvs ojii lc ibnf klat atja wbcrafg?" Tlc Ljc revvfcis rjlv. "Mbg tlw, sbe kbeiv gjatfg rajs tfgf, jcv sbe vbc'a*

kjca ab ifjnf klat wf? Lf'r wbgf lwqbgajca ab sbe atjc P jw? Cgfc'a kf ibnfgr?" Qjr atlr atf yfulcclcu bo j ibnf agljcuif?! Ccv ktfc vlv atfs yfmbwf ibnfgr! Vtjb Jl kjr raeqfolfv. "P-la'r cba ilxf atja..." "Then leave with me and I'll believe you." Yin Han held out his hand again, his voice trembling slightly, his eyes filled with pleading. "As long as you take my hand, no matter what happened, I'll pretend like it never happened." FkUdpD Shao Ci froze, looking at the hand before his eyes. He didn't move for a long time. The light in Yin Han's eyes slowly disappeared. [...]"

Alphanumerische Sequenzen am Ende mancher Zeilen und Nonsense-Text dazwischen. Auf der Website ist nur der eigentliche Text zu sehen.

Text mit der ID 2316300:

„I am using machine translation and is deserved no credit for it. It is not 100% nor do I claim it to be that all the translation or edit is correct. My job is only to correct grammatical mistakes (or not), in general to make it easier to read. BTW if you see this [] it means me, editor, as it's quite a hassle to always put Ëditorïnside the bracket. About an hour later, a group of more than 2,000 troops from thorns flower came slowly, led by a golden knight. [...]"

Anmerkung des Autors zu Beginn.

Text mit der ID 3761008:

„-范跃(fàn yuè) – Fan Yue, female lead's second son
-严氏(yán shì)- Yan Shi, the eldest brother's wife/ eldest sister in law
-兰草(lán cǎo)- Lan Cao, was one of the female lead's maid who is currently Fan Ceng's concubine
-兰芝(lán zhī)- Lan Zhi, was one of the female lead's maid who is currently Fan Ceng's concubine.
A letter was sent back after Fan Ceng had been away for more than a month. [...]"

Aufführung von Personennamen mit Bezug zum chinesischen Original zu Beginn des Textes. Hier kann man durchaus auch die Auffassung vertreten, dass dies zum gewünschten Text gehört.

Text mit der ID 4830688:

„[...] Pei Yi leaned against the door, "You— What are you doing? " ——————————You are reading this at danoveltranslations.com and not at a reposted one right? This site is already available for free and

soon there will be ads, so why contribute to someone stealing others ow n work?———————————————– Purple/violet button is for the next and previous button. For cp is below and pc is at the middle on each side. Please Consider to donate for me in order to maintain the website and provide quality translations. “

Anmerkung des Autors am Ende des Textes (Footer).

Literaturverzeichnis

[1] Mike Kestemont, Michael Tschuggnall, Efstathios Stamatatos, Walter Daelemans, Günther Specht, Benno Stein, and Martin Potthast. Overview of the Author Identification Task at PAN-2018: Cross-domain Authorship Attribution and Style Change Detection. In *Working Notes Papers of the CLEF 2018 Evaluation Labs*, volume 2125 of *CEUR Workshop Proceedings*. CEUR-WS.org, September 2018. URL `http://ceur-ws.org/Vol-2125/`.

[2] Novelupdates. URL `https://www.novelupdates.com/`. [Online, Stand 29. Juli 2022].

[3] Webis [4]. PAN. URL `https://pan.webis.de/`. [Online, Stand 29. Juli 2022].

[4] Webis-Gruppe. URL `https://webis.de/`. [Online, Stand 29. Juli 2022]; Anfragen an: Webis Group, Bauhaus-University Weimar, Faculty of Media, Schwanseestraße 143, 99423 Weimar, Germany, E-Mail: `webis[at]listserv.uni-weimar.de`.

[5] Wikipedia. Textkorpus — Wikipedia, die freie Enzyklopädie, 2022. URL `https://de.wikipedia.org/w/index.php?title=Textkorpus&oldid=220478651`. [Online, Stand 29. Juli 2022].

[6] H. Kucera W. N. Francis. Brown Corpus. URL `http://korpus.uib.no/icame/manuals/BROWN/INDEX.HTM`. [Online, Stand 29. Juli 2022].

[7] Project Gutenberg. Project Gutenberg Literary Archive Foundation, 809 North 1500 West, Salt Lake City, UT 84116. URL `https://www.gutenberg.org/`. [Online, Stand 29. Juli 2022].

[8] NLTK Project. Steven Bird, Ewan Klein, and Edward Loper (2009). Natural Language Processing with Python. O'Reilly Media Inc. URL `https://www.nltk.org/`. [Online, Stand 29. Juli 2022].

[9] The Cold Wire. URL `https://www.thecoldwire.com/difference-between-web-novel-and-light-novel/`. [Online, Stand 29. Juli 2022].

[10] Yin Lin Tan. How Tencent, the Chinese Internet Giant, Cracked Down On A Niche Internet Subculture, 2020. URL `https://www.ricemedia.co/culture-people-tencent-chinese-internet-giant-cracked-down-niche-internet-subculture/`. [Online, Stand 29. Juli 2022].

[11] Novelupdates-Forum. Learning the history of Fan Translations takedowns. URL `https://forum.novelupdates.com/threads/learning-the-history-of-fan-translations-takedowns.124411/`. [Online, Stand 29. Juli 2022].

[12] Oren Halvani, Christian Winter, and Lukas Graner. Assessing the Applicability of Authorship Verification Methods. In *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ARES '19, page 10, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 978-1-4503-7164-3. doi: 10.1145/3339252.3340508.

[13] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Computational methods in authorship attribution. *Journal of the American Society for Information Science and Technology*, 60(1):9–26, 2009. doi: 10.1002/asi.20961.

[14] Moshe Koppel, Jonathan Schler, Elisheva Bonchek-Dokow, Cs Biu Ac Il, Cs Biu Ac Il, and Bonchek Dokow. Measuring Differentiability: Unmasking Pseudonymous Authors. *J. Mach. Learn. Res.*, 8:1261–1276, 2007.

[15] Efstathios Stamatatos. Authorship Verification: A Review of Recent Advances. *Research in Computing Science*, 123(1):9–25, December 2016. ISSN 1870-4069. doi: 10.13053/rcs-123-1-1.

[16] Hans van Halteren. Linguistic profiling for authorship recognition and verification. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, ACL '04, pages 199–206, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1218955.1218981.

[17] Moshe Koppel and Jonathan Schler. Authorship verification as a one-class classification problem. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 62, New York, NY, USA, 2004. Association for Computing Machinery. ISBN

1581138385. doi: 10.1145/1015330.1015448. URL https://doi.org/10.1145/1015330.1015448.

[18] Moshe Koppel and 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, 2014. ISSN 2330-1643. doi: 10.1002/asi.22954.

[19] Patrick Juola. Authorship attribution. *Found. Trends Inf. Retr.*, 1(3): 233–334, December 2006. ISSN 1554-0669. doi: 10.1561/1500000005.

[20] Efstathios Stamatatos. Authorship attribution based on feature set subspacing ensembles. *International Journal on Artificial Intelligence Tools*, 15:823–838, 10 2006. doi: 10.1142/S0218213006002965.

[21] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology*, 60(3):538–556, 2009. ISSN 1532-2890. doi: 10.1002/asi.21001.

[22] David I. Holmes. Authorship attribution. *Computers and the Humanities*, 28(2):87–106, 1994.

[23] Farkhund Iqbal, Hamad Binsalleeh, Benjamin CM Fung, and Mourad Debbabi. Mining writeprints from anonymous e-mails for forensic investigation. *digital investigation*, 7(1-2):56–64, 2010.

[24] Robert Layton. A simple local n-gram ensemble for authorship Verification – Notebook for PAN at CLEF 2014. In *Working Notes Papers of the CLEF 2014 Evaluation Labs*, page 7. CEUR-WS.org, September 2014.

[25] Robert Layton, Paul Watters, and Richard Dazeley. Automated unsupervised authorship analysis using evidence accumulation clustering. *Natural Language Engineering*, 19(1):95–120, 2013.

[26] Kim Luyckx, Walter Daelemans, and Edward Vanhoutte. Stylogenetics: Clustering-based stylistic analysis of literary corpora. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC'06), Genoa, Italy*, page 6, 2006.

[27] Rajhans Samdani, Kai-Wei Chang, and Dan Roth. A discriminative latent variable model for online clustering. In *International Conference on Machine Learning*, pages 1–9. PMLR, 2014.

[28] Angela Glover and Graeme Hirst. Detecting Stylistic Inconsistencies in Collaborative Writing. In *The New Writing Environment: Writers at Work in a World of Technology*, pages 147–168. Springer, London, 1996. ISBN 978-1-4471-1482-6. doi: 10.1007/978-1-4471-1482-6_12.

[29] Andi Rexha, Stefan Klampfl, Mark Kröll, and Roman Kern. Towards a more fine grained analysis of scientific authorship: Predicting the number of authors using stylometric features. In *BIR@ECIR*, pages 26–31, 2016.

[30] Neil Graham, Graeme Hirst, and Bhaskara Marthi. Segmenting documents by stylistic character. *Natural Language Engineering*, 11(4): 397–415, December 2005. ISSN 1351-3249, 1469-8110. doi: 10.1017/S1351324905003694.

[31] Navot Akiva and Moshe Koppel. Identifying Distinct Components of a Multi-author Document. In *2012 European Intelligence and Security Informatics Conference, EISIC 2012*, pages 205–209, Odense, Denmark, August 2012. IEEE Computer Society. doi: 10.1109/EISIC.2012.16.

[32] Navot Akiva and Moshe Koppel. A generic unsupervised method for decomposing multi-author documents. *Journal of the American Society for Information Science and Technology*, 64(11):2256–2264, 2013. ISSN 1532-2890. doi: 10.1002/asi.22924.

[33] Michael Tschuggnall and Günther Specht. Automatic Decomposition of Multi-Author Documents Using Grammar Analysis. In *Proceedings of the 26th GI-Workshop Grundlagen von Datenbanken, 21-24 October, Bozen-Bolzano, Italy*, page 6. CEUR-WS.org, October 2014.

[34] Mathias Payer, Ling Huang, Neil Zhenqiang Gong, Kevin Borgolte, and Mario Frank. What you submit is who you are: A multimodal approach for deanonymizing scientific publications. *IEEE Transactions on Information Forensics and Security*, 10(1):200–212, 2014.

[35] Robert Deibel and Denise Löfflad. Style change detection on real-world data using an LSTM-powered attribution Algorithm – Notebook for PAN at CLEF 2021. In *CLEF 2021 Labs and Workshops, Notebook Papers*, pages 1899–1909. CEUR-WS.org, September 2021.

[36] Sukanya Nath. Style change detection using Siamese neural networks – Notebook for PAN at CLEF 2021. In *CLEF 2021 Labs and Workshops, Notebook Papers*, pages 2073–2082. CEUR-WS.org, September 2021.

[37] JetBrains s.r.o. PyCharm. URL `https://www.jetbrains.com/de-de/pycharm/`. [Online, Stand 29. Juli 2022; genutzte Version: 3.9].

[38] Heritrix. URL `https://github.com/internetarchive/heritrix3`. [Online, Stand 29. Juli 2022; genutzte Version: 3.4.0-20210803].

[39] Webis-scriptor. URL `https://github.com/webis-de/scriptor`. [Online, Stand 29. Juli 2022; genutzte Version: 0.7.0].

[40] Leonard Richardson. Beautiful Soup. URL `https://www.crummy.com/software/BeautifulSoup/`. [Online, Stand 29. Juli 2022; genutzte Version: 4.10.0].

[41] Aaron Swartz, Alireza Savand. HTML2Text. URL `https://github.com/Alir3z4/html2text`. [Online, Stand 29. Juli 2022; genutzte Version: 2020.1.16].

[42] Scrapy Project. Parsel. URL `https://github.com/scrapy/parsel`. [Online, Stand 29. Juli 2022; genutzte Version: 1.6.0].

[43] Readability. URL `https://github.com/buriy/python-readability`. [Online, Stand 29. Juli 2022; genutzte Version: 0.8.1].

[44] Janek Bevendorff et al. Chatnoir-resiliparse. URL `https://github.com/chatnoir-eu/chatnoir-resiliparse`. [Online, Stand 29. Juli 2022; genutzte Version: 0.10.5].

[45] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York, March 2018. Springer.

[46] Adrien Barbaresi. Trafilatura. URL `https://github.com/adbar/trafilatura`. [Online, Stand 29. Juli 2022; genutzte Version: 1.1.0].

[47] Scikit-learn. URL `https://github.com/scikit-learn/scikit-learn`. [Online, Stand 29. Juli 2022; genutzte Version: 1.0.2].

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[49] Patrick Juola. An Overview of the Traditional Authorship Attribution Subtask. In *CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy*. CEUR-WS.org, September

2012. ISBN 978-88-904810-3-1. URL `http://ceur-ws.org/Vol-1178/CLEF2012wn-PAN-Juola2012.pdf`.

[50] John Burrows. 'Delta': a Measure of Stylistic Difference and a Guide to Likely Authorship. *Literary and Linguistic Computing*, 17(3):267–287, 09 2002. ISSN 0268-1145. doi: 10.1093/llc/17.3.267. URL `https://doi.org/10.1093/llc/17.3.267`.

[51] Stefan Evert, Thomas Proisl, Fotis Jannidis, Isabella Reger, Steffen Pielström, Christof Schöch, and Thorsten Vitt. Understanding and explaining Delta measures for authorship attribution. *Digital Scholarship in the Humanities*, 32(suppl_2):ii4–ii16, 06 2017. ISSN 2055-7671. doi: 10.1093/llc/fqx023. URL `https://doi.org/10.1093/llc/fqx023`.

[52] Martin Potthast, Sarah Braun, Tolga Buz, Fabian Duffhauss, Florian Friedrich, Jörg Marvin Gülzow, Jakob Köhler, Winfried Lötzsch, Fabian Müller, Maike Elisa Müller, Robert Paßmann, Bernhard Reinke, Lucas Rettenmeier, Thomas Rometsch, Timo Sommer, Michael Träger, Sebastian Wilhelm, Benno Stein, Efstathios Stamatatos, and Matthias Hagen. Who wrote the web? revisiting influential author identification research applicable to information retrieval. In *Advances in Information Retrieval*, pages 393–407, Cham, 2016. Springer International Publishing. ISBN 978-3-319-30671-1.

[53] Robert Paßmann. Burrows Delta Webis Reimplementierung, 2015. URL `https://github.com/pan-webis-de/burrows02`. [Online, Stand 27. April 2022].

[54] David L. Hoover. Testing Burrows's Delta. *Literary and Linguistic Computing*, 19(4):453–475, 11 2004. ISSN 0268-1145. doi: 10.1093/llc/19.4.453. URL `https://doi.org/10.1093/llc/19.4.453`.

[55] Timo Sommer. Stamatatos06 Webis Reimplementierung, 2015. URL `https://github.com/pan-webis-de/stamatatos06`. [Online, Stand 15. April 2022].

[56] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Generalizing unmasking for short texts. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 654–659, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1068. URL `https://aclanthology.org/N19-1068`.

[57] Janek Bevendorff. Unmasking Webis Reimplementierung. URL `https://github.com/webis-de/unmasking`. [Online, Stand 30. April 2022].

[58] Efstathios Stamatatos, Walter Daelemans, Ben Verhoeven, Patrick Juola, Aurelio López López, Martin Potthast, and Benno Stein. Overview of the Author Identification Task at PAN 2015. In *Working Notes Papers of the CLEF 2015 Evaluation Labs* Stamatatos et al. [58]. URL `http://ceur-ws.org/Vol-1391/`.