Praktikumsbericht

Machine Learning for Natural Language Processing

am Lehrstuhl für Informatik X

Konstantin Herud Thomas Schaffroth Maximilian Meißner

Abgabedatum:	16. August 2020
Betreuer:	Daniel Schlör
	Albin Zehe
	Konstantin Kobs
	Tobias Koopmann

Julius-Maximilians-Universität Würzburg
Lehrstuhl für Informatik X
Data Science

Zusammenfassung

Dieses Dokument soll Studenten an unserem Lehrstuhl bei der Erstellung ihrer Abschlussarbeit unterstützen. Wir zeigen eine beispielhafte Gliederung einer Arbeit und beschreiben die Inhalten der einzelnen Kapitel. Zusätzlich geben wir an vielen Stellen auch Hinweise zur Benutzung von LaTeX für die Erstellung der Arbeit. Im Anhang **??** geben wir ein paar Hinweise zum Ablauf der Betreuung von Abschlussarbeiten an unserem Lehrstuhl.

Zur Handhabung dieses Pakets. In diesem Paket sind Vorlagen für verschiedene Dokumenttypen enthalten, die sie als Ausgangspunkt für ihre Arbeit verwenden können. Es gibt jeweils Vorlagen für deutsche und englische Arbeiten.

- `template_thesis_de.tex`, `template_thesis_en.tex`: Vorlage für Bachelorarbeit bzw. Masterarbeit
- `template_seminar_de.tex`, `template_seminar_en.tex`: Vorlage für Seminarausarbeitungen und Praktikumsberichte

Der Quelltext zu diesem Leitfaden ist ebenfalls im Paket enthalten. Diesen können Sie als praktisches Beispiel dafür verwenden, wie diese Dokumentenklasse angewandt wird.

Inhalt der Zusammenfassung. Schreiben Sie hier eine Zusammenfassung der Arbeit, vergleichbar mit dem Abstract auf wissenschaftlichen Papers. Sie dient dem Leser dazu, einen groben Überblick über die Inhalte zu gewinnen (Problemstellung, verwendeter Lösungsansatz, ggf. experimentelle Ergebnisse, gewonnene Erkentnisse). Der Umfang soll ca. eine halbe Seite betragen. Für Seminararbeiten ist diese Zusammenfassung nicht erforderlich.

Achtung: Bei Arbeiten auf Englisch fordern die Prüfungsordnungen, dass es eine deutsche Zusammenfassung gibt. Schreiben Sie in diesem Fall eine englische *und* eine deutsche Zusammenfassung (mit dem gleichen Inhalt). Die passenden LaTeX-Befehle dafür finden Sie in den englischsprachigen Vorlagen.

WARNUNG: Die vorliegende Version des Leitfadens ist eine Vorabversion, die noch nicht vollständig ist. Sie bezieht sich größtenteils auf die Ausarbeitung von Bachelor- und Masterarbeiten; Seminararbeiten unterscheiden sich davon etwas in Aufbau und Inhalt.

Inhaltsverzeichnis

1. Einleitung

Thomas Schaffroth

Nachdem der in diesen Tagen gängige Begriff der *Artificial Intelligence* (AI) auf der Dartmouth Conference 1956 erstmals geprägt wurde, bezeichnet dieser bis heute das entsprechende Forschungs- und Entwicklungsfeld. Was damals mit dem computergestützten Lösen algebraischer Probleme oder der Demonstration geometrischer Theoreme begann, brachte im Laufe der Jahre einen umfangreichen, sich schnell entwickelndem Bereich der Informatik hervor. Nicht zuletzt stellten Innovationen, wie die Bayesschen Lernverfahren 1960, die Einführung des „Parallel Computings“ 1980 oder die Erfindung des Backpropagation-Algorithmus für Neuronale Netze 1984, wesentliche Meilensteine in dieser Entwicklung dar.

Durch aufkommende Neuerungen zu Beginn der 2000er Jahre, wie der Herausforderung des Big Data sowie der Entfaltung des Internets und der mobilen Kommunikation wurden seitdem große Fortschritte in Feldern, wie der Computer Vision, Intelligenten Agenten und Mustererkennung erzielt. Es sind unter anderem ebendiese Weiterentwicklungen, die den Weg für moderne Herausforderungen, wie Spracherkennung, selbstfahrende Fahrzeuge und Natural Language Processing (NLP), geebnet haben [23]. Besonders das Themengebiet des NLP nimmt einen wichtigen Stellenwert in vielen Software-Anwendungen unseres täglichen Lebens ein. Einige der herausragendsten Beispiele sind E-Mail Plattformen, wie Microsoft Outlook (Spam-Klassifikation, Auto-Complete, etc.), sprachbasierte Assistenzsysteme, wie Apple Siri und Amazon Alexa oder Services für maschinelle Übersetzung, wie Google Translate [31].

Die zunehmende Bedeutung von NLP lässt sich neben seiner Rolle in unserem Alltag auch in der Forschung beobachten. Die ACL Anthology (AA) ist ein digitales Repository, dass zehntausende Veröffentlichungen von NLP-Papern aus der Familie der ACL- sowie anderer NLP-Konferenzen beinhaltet. Lag die Anzahl der Veröffentlichungen im Jahr 2000 noch bei 1050 wurden 2018 bereits 4173 publizierte Paper verzeichnet [21].

In diesem Praktikum wird im Folgenden das Natural Language Processing an sich mit wissenschaftlichen Arbeiten, die zu diesem Thema veröffentlicht wurden, zusammengeführt. Ziel dieser Arbeit ist die Bearbeitung der Fragestellung, ob die Erkennung der Herkunft von Autoren wissenschaftlicher Publikationen aufgrund textueller Eigenschaften ihrer Veröffentlichungen möglich ist, sowie welche Methoden bessere oder schlechtere Ergebnisse liefern. Der hierfür zu Verfügung stehende Datensatz beinhaltet 18000 Paper verschiedener NLP- und AI-Konferenzen in PDF- und Text-Form.

Das Praktikum gliedert sich in zwei Bestandteile. Zunächst müssen die Paper einer grundsätzlichen Datenbereinigung unterzogen werden. Informationen, von denen direkt oder indirekt auf die Herkunft der Autoren geschlossen werden kann, müssen dabei aus den Arbeiten entfernt werden. Beispiele hierfür sind Autoren-Namen, E-Mail-Adressen,

Ländernamen, Institutionen oder Referenzen im Text. Der Grund dafür ist, das für den zweiten Teil der Arbeit sichergestellt werden muss, dass die Klassifikation der Paper aufgrund allgemeiner Texteigenschaften, wie der Syntax und dem verwendeten Vokabular stattfindet.
Informationen, die sich auf die Herkunft des Autors beziehen, würden für eine Verfälschung des Klassifikationsergebnisses sorgen. Der folgende Schritt beschäftigt sich mit der Feature-Modellierung, dem Training und der Evaluierung im Kontext verschiedener Techniken der AI und NLP auf dem genannten Korpus in Bezug auf die Klassifikations-Aufgabe. Ziel soll sein, einen Klassifikator zu trainieren, der für eine möglichst große Zahl der Paper das Herkunftsland des jeweiligen Autors korrekt vorhersagt.

2. Methodik

Konstantin Herud

2.1. Task 1: Dataset Preparation

Da verschiedene Informationen das Herkunftsland von Autoren entweder direkt oder indirekt preisgeben, müssen diese aus der Datenbasis entfernt werden. Die Zielsetzung sieht hierbei vor Namen der Autoren, E-Mails, Institutionen und Firmen, Herkunftsländer, Förderungen und Danksagungen, Persönliche Daten sowie Referenzen zu entfernen. Referenzen sind beispielsweise indirekte Hinweise, weil Arbeitsgruppen der gleichen Herkunft und somit dem selben Fachgebiet die selbe Literatur referenzieren.
Da diese Aufgabe essentieller Bestandteil der Validität späterer Ergebnisse ist, wurde eine mehrstufige Daten-Pipeline entwickelt, um alle Informationen bestmöglich aus den Texten zu entfernen. Jeder folgende Paragraph entspricht einer der Stufen.

Textextraktion Im ersten Schritt muss der rohe Text der Dokumente eingelesen werden. Die Daten liegen sowohl in PDF- als auch in Textform vor, wobei letztere durch das Befehlszeilenprogramm *pdftotext* erstellt wurde und somit bereits Potenzial für Fehler bietet. Ursprünglich wurde deshalb ein Ansatz mit dem Werkzeug *pdfminer* verfolgt, um den Text der Dokumente selbstständig zu extrahieren und dabei direkt Strukturinformationen ausnutzen zu können (z. B. um Referenz-Blöcke zu erkennen). Bei einigen Dokumenten entstanden jedoch Fehler mit pdfminer, weshalb dieser Ansatz verworfen wurde. So werden in der Pipeline zunächst die Textdateien sowie zugehörige Metadaten eingelesen [1]. Da verschiedene spätere Schritte außerdem Informationen über die Zeichenposition des Abstract-Starts brauchen, wird dieser zudem bereits mit einem regulären Ausdruck, unterstützt durch verschiedene Fehlermaßnahmen, bestimmt. So wird beispielsweise überprüft, ob der Titel des Dokuments das Wort Abstract enthält, um zu frühe Treffer des regulären Ausdrucks zu überspringen. Falls der Start nicht gefunden werden kann oder eine bestimmte Obergrenze überschreitet, wird ein mittelwert-basierter Defaultwert verwendet.

Dokumentkopf Als Dokumentkopf definieren wir jeglichen Text (in Rohform), der vor dem Abstract erscheint. Die hohe Dichte unterschiedlicher zu entfernender Informationen, gepaart mit Artefakten und Strukturfehlern, machen diese Stufe zu einer der größten Herausforderungen der Aufgabe. So kommen beziehen sich vor allem Namen, E-Mails,

[1] `https://github.com/marekrei/ml_nlp_paper_data`

Herkunftsländer, Institutionen und persönliche Daten auf diesen Schritt. Initial wurde dafür die Idee verfolgt, *Named-Entity-Recognition* mittels *spaCy* [10] einzusetzen, um all diese Informationen (außer E-Mails, welche leicht durch reguläre Ausdrücke zu erkennen sind) zu entfernen. SpaCy bietet dafür verschiedene vortrainierte Modelle, welche für diese Arbeit mit im Text durch reguläre Ausdrücke erkannten Informationen nachtrainiert wurden. Da dieser Ansatz Entitäten nicht nur zu unverlässlich erkannte (insbesondere keine Phrasen, die über mehrere Tokens reichen), sondern auch zu schlecht klassifzierte wurde ein weiterer Ansatz entwickelt. So wurde ein *LSTM*-basiertes [9], neuronales Netzwerk implementiert, um jede Zeile (im Textdokument, also keine ganzen Sätze) des Vor-Abstract-Texts in die Klassen Autor, E-Mail, Private Informationen, Institution oder Anderes zu unterteilen. Länder werden in einem späteren Schritt entfernt. Zum Training wurden 250 Dokumente zeilenweise mit einer Kommandozeilenanwendung annotiert. Dabei wurden zufällig Dokumente aus dem Korpus gezogen, allerdings wurde sichergestellt, dass jede Konferenz vorkommt. Zudem sind verschiedene Fehlerfälle enthalten, bei denen der Start des Abstracts nicht korrekt erkannt wurde und somit späterer Text enthalten ist. Das Modell erreicht auf weiteren 50 Dokumenten einen Makro-F1-Wert von etwa 96%. Die Architektur verfolgt einen ähnlichen Ansatz wie Gleichung 2.3–2.6, nutzt jedoch nach X_{zd} ein zweites LSTM zur zeilenweisen Klassifikation.

Referenzen Referenzen unterteilen sich in zwei Typen: Verweise innerhalb des Texts und ausführlich Referenzblöcke. Erstere sind aufgrund ihrer einheitlichen Struktur leicht mittels regulären Ausdrücken zu ermitteln. Dafür wurden zahlreiche zufällige Dokumente betrachtet und drei Strukturtypen identifiziert, für die eigene reguläre Ausdrücke entwickelt wurden: Klammer-Ausdrücke, die Jahreszahlen im 20. und 21. Jahrhundert enthalten, Phrasen mit “et. al.” und Aufzählungen (z. B. verbunden durch “und”, “,” oder “&” etc.), gefolgt von Jahreszahlen. Referenzblöcke zu identifizieren stellt sich jedoch als zweitschwerste Aufgabe heraus. Ein initialer Ansatz sah vor, die zuvor im Text gefundenen Referenzen zusammen mit verschiedenen Schlüsselwörtern dazu zu nutzen, Zeilen-Blöcke als Referenzen zu erkennen. Da durch die Konvertierung von zweispaltigen, mehrseitigen Dokumenten in eindimensionalen Text jedoch häufig Strukturfehler auftreten, sind Teile der Referenzblöcke oftmals stark über die Dokumente verteilt, weshalb dieser initiale Ansatz zu keinen befriedigenden Ergebnissen führte. Deshalb wurde ebenfalls ein neuronales Netz, mit einer Architektur äquivalent zur vorherigen, trainiert, um zeilenweise zu annotieren, ob es sich um Referenzen handelt. Dafür wurden etwa 50 Dokumente manuell annotiert, wobei neben zufällig gewählten Werken insbesondere darauf geachtet wurde, Fehlerfälle zu verwenden. Das Netz verarbeitet den Text iterativ in Stücken von 80 Zeilen (um Padding und Grafikspeicher-Anforderungen möglichst gering zu halten, wurde kein kompletter *End-to-End* Ansatz gewählt). Um dennoch von Informationen über die Position dieser 80 Zeilen im Dokument zu profitieren, wird auf die d-dimensionalen $(1, ..., i, ..., d)$, internen Zustände des Netzwerks ein Positionssignal

PE addiert (vgl. [32], Gleichung 2.1 & 2.2).

$$PE(zeile, 2i) = \sin\left(\frac{zeile}{10000^{2i/d}}\right) \tag{2.1}$$

$$PE(zeile, 2i+1) = \cos\left(\frac{zeile}{10000^{2i/d}}\right) \tag{2.2}$$

Das Modell erreicht auf 20% separaten, zufälligen Daten einen Makro-F1-Wert von beinahe 99,7%. Sowohl das Dokumentkopf- als auch das Referenzblocknetz sind mit ihren Gewichten aufgrund der hoch-rekurrenten Struktur unter einem Megabyte groß.

Gutachter Generell wurden wenige Gutachter innerhalb der Dokumente festgestellt. Diese geringe Menge folgt jedoch einem festen Schema, welches mittels eines regulären Ausdrucks ermittelt wird.

Danksagungen Danksagungen sind meist wenige Zeilen in geschlossenen Blöcken und somit leichter zu ermitteln als Referenzblöcke. Vorerst stand im Raum, ebenfalls ein neuronales Netz einzusetzen, allerdings genügen reguläre Ausdrücke. So wird nach dem letzten Vorkommen der gängigen Überschrift "Acknowledgment" gesucht (wobei unterschiedliche Schreibweisen zu beachten sind, i. e. Acknowledge?ments?). Das Ende der Danksagung wird durch eine Liste mit Schlüsselwörtern, kombiniert mit den zuvor gefunden Referenzen sowie einem regulären Ausdruck für Referenzen (da Danksagungen diesen häufig vorausgehen), ermittelt. Kann dennoch kein Ende gefunden werden, wird der Text bis zum Ende des Paragraphen gewählt.

E-Mails Der Großteil aller E-Mails wurde in diesem Schritt bereits durch die Kopfzielen-Stufe entfernt (insbesondere schwer identifizierbare Fälle, deren gezielt irreführende Schreibweise z. B. Spam verhindern soll). Der Rest aller E-Mails wird in diesem Schritt mit zwei weiteren regulären Ausdrücken entfernt.

Herkunftsländer Länder und Nationalitäten werden mit dem Python-Modul *geotext* [2] ermittelt. Das Modul arbeitet mit regulären Ausdrücken und der geographischen Datenbank *GeoNames*, die jegliche Länder und über elf Millionen Ortsnamen enthält.

Fußnoten Fußnoten sind unscharf definiert und aufgrund der fehlenden visuellen Struktur sehr schwer im rohen Text zu identifizieren. Um diese Informationen dennoch zu entfernen, wird der Text zunächst mittels spaCy in Sätze unterteilt. Anschließend werden alle Sätze mit einer Liste von Schlüsselwörtern, wie "sponsored" oder "internship", sowie bereits ermittelten Autor- und Organisationsnamen untersucht. Wird ein Schlüsselwort gefunden, wird der entsprechende Satz entfernt.

[2] `https://github.com/elyase/geotext`

Sprache Zuletzt wird nicht-englische Sprache entfernt. Diese Aufgabe stellt ebenfalls eine Herausforderung dar, da viele Rohtext-Fragmente aufgrund kryptischer Formeln oder anderen Artefakten von gängigen Sprachidentifikationssystemen fälschlicherweise als nicht-englisch erkannt werden. Dennoch wird die spaCy-Erweiterung *CLD* eingesetzt, welche die *Compact Language Detection 2* Bibliothek [3] anbindet. Diese ermöglicht die Klassifikation von rohem Text in über 80 verschiedene Sprachen mittels Zeichen-N-Grammen und einem Naiven-Bayes-Modell. Dieser Ansatz führt zu einer akzeptablen Sensitivität, allerdings zu einer schlechten Genauigkeit. Um letztere zu verbessern, wird lediglich eine geringe Teilmenge relevanter Sprachen der 80+ möglichen berücksichtigt.

Die einzelnen Stufen der Pipeline lassen sich nicht nur jeweils über verschiedene Prozesse verteilen und vervielfältigen, die Pipeline lässt sich auch als Ganzes parallelisieren. Zusätzlich ermöglicht die Anwendung Hardware-Beschleunigung für die neuronalen Netzwerke. Das Wissen über zu entfernende Informationen wird akkumuliert und letztlich in eine CSV-Datei geschrieben, außerdem wird der inkrementell bereinigte sowie der ursprüngliche Text zwischen Stufen weitergegeben und abschließend in eine Text-Datei geschrieben.

2.2. Task 2: Learning to Discriminate

Um die bereinigten Dokumente hinsichtlich des Herkunftslands ihrer Autoren zu klassifizieren, wurden zwei Ansätze verfolgt: Zum einen Klassifikation mittels verschiedenen *stylometrischen* Merkmalen, zum anderen ein *End-to-End-Deep-Learning*-Modell.

2.2.1. Stylometrische Klassifikation

Stylometrie befasst sich mit der Untersuchung von Sprachstil in der Regel zur Bestimmung des Autors [26]. Diesem Abschnitt liegt die Annahme zugrunde, dass sich die selben Mittel auch zur Bestimmung des Herkunftlands übertragen lassen. Statistische und rechnerische Zuweisung der Urheberschaft hat generell eine lange Geschichte, die bis ins 19. Jahrhundert reicht [28]. Dementsprechend existieren viele Methoden, die von einfachen Ideen, wie Mendenhalls Kurven zu Wortlängen [19], über Kilgarriffs χ^2-Tests zu Vokabularähnlichkeit [12] bis zu fortgeschritteneren Ideen, wie Burrows Delta Statistik [6], reicht. In dieser Arbeit wurden verschiedene linguistische Merkmale extrahiert.

- **Lexikalische Diversität**: Für verschiedene Maße zum Reichtum des Wortschatzes einzelner Dokumente wurde das Python-Modul *lexicalrichness* eingesetzt [4]. Insgesamt wurden die Maße Type-Token-Ration (TTR) [29], Guirauds Index [7], Corrected-TTR [3], Herdans Maß [8], Somers Maß [25], Dugasts Index [5], Maas Index [16], Mean-Segmental-TTR [30], Moving-Average-TTR [4], Measure of Textual Lexical Diversity [18] und die Hypergeometrische-Verteilungs-Diversität [17] ermittelt.

[3] `https://github.com/CLD2Owners/cld2`
[4] `https://github.com/LSYS/lexicalrichness`

- **Wort- und Satzlängen**: Für Wort- und Satzlängen werden Histogramme der Dokumente gebildet, die mittels der Anzahl an Wörtern beziehungsweise Sätzen insgesamt normalisiert wird. Wortlängen werden in Zeichen gemessen und auf 50 beschränkt, Satzlängen mittels ihrer Anzahl an Wörtern bis 100 (Anhang A zeigt solche Histogramme für alle Dokumente kombiniert).
- **Interpunktion**: Häufigkeiten von genutzten Interpunktionszeichen (jegliche in Python durch *string.punctuation* definierte Zeichen), normalisiert über die gesamte Zeichenanzahl im Dokument.
- **Funktionswörter**: Häufigkeiten von insbesondere Stopwörtern und weiteren Kandidaten (insgesamt 277 Funktionswörter [5]).
- **N-Gramme**: Den wichtigsten Teil bilden die N-Gramm-Histogramme. Dafür werden jeweils die 200 häufigsten 2- und 3-Gramme über die Lemmata der Wörter, über OntoNotes5-POS-Tags und über universelle POS-Tags mittels spaCy ermittelt [33].

Für jedes Dokument werden alle Merkmale extrahiert und anschließend mit einem einfachen neuronalen Netzwerk klassifiziert. Das Netzwerk besteht aus einer beliebigen Anzahl Blöcken, die jeweils aus einer verdeckten Schicht, Batch-Normalisierung [11], SELU-Aktivierung [14] und anschließend Dropout [27] bestehen. Aufgrund der einfachen und schnell zu trainierenden Architektur werden die Hyperparameter (wie Anzahl Blöcke, Anzahl Neuronen, Dropout etc.) mit einer umfangreichen Zufallssuche ermittelt. Neben einem Netzwerk mit der Kombination aller Merkmale existieren separate Netzwerke für alle Merkmalstypen alleinstehend, mit der Intention diese vortrainierten Subnetzwerke später mit dem *End-To-End*-Modell zu kombinieren.

2.2.2. End-to-End Klassifikation

Als *End-to-End* werden Modelle bezeichnet, die keine vorherigen oder manuellen Schritte, wie Merkmalsextraktion, benötigen, sondern ausschließlich anhand der rohen, bereinigten Texte klassifizieren. Dafür ist zunächst eine Methode zur Textrepräsentation notwendig. Gängige frühe Verfahren zur Repräsentation ganzer Wörter sind latente *Word2Vec*- oder *GloVe*-Vektoren [20, 22]. Diese sind jedoch aus zahlreichen Gründen problematisch, insbesondere werden die Wörter dabei in der Regel auf Lemmata, Wortstamm oder Ähnliches normalisiert, um die Anzahl benötigter Vektoren zu reduzieren. Zudem existieren für falsche Schreibweisen, seltene oder zusammengesetzte Wörter und andere Eigenheiten keine individuellen Vektoren. All diese Informationen spielen aber potentiell eine wichtige Rolle zur Klassifizierung. *FastText* löst diese Probleme teilweise, indem Wörter als Summen von Zeichen-n-Gramm-Vektoren dargestellt werden und somit auch unbekannte Sequenzen repräsentiert werden können [2]. Der gegenwärtige Standard und auch in dieser Arbeit gewählte Ansatz ist jedoch eine *Byte Pair Encoding*-Subwortdarstellung [24]. Hierbei werden ausgehend von einer Repräsentation auf Zeichenebene inkrementell die häufigsten Symbolfolgen zu einzelnen Subwörtern kombiniert,

[5] `https://semanticsimilarity.wordpress.com/function-word-lists`

wodurch letztlich häufige Wörter als eigene Vokabeln existieren, unbekannte Wörter jedoch auch durch Subwortsequenzen ausgedrückt werden können. Dies ermöglicht trotz geringer Vokabulargröße Texte kompakt zu repräsentieren. Aufgrund der eigenen wissenschaftlichen Semantik der Konferenzen und der spezifischen Syntax der Textdateien mit ihren Strukturfehlern und Artefakten, werden keine vortrainierten Embeddings verwendet, sondern während dem Trainingsprozess von Grund auf erlernt. Zudem wurde probiert, die Embeddings eigenständig vorzutrainieren, indem das spätere, anschließend weiterverwendete LSTM (Gleichung 2.3 & 2.4) mittels *Masked Lanugage Modelling* trainiert wird. Die Idee dabei ist, einzelne Wörter zu maskieren, um anschließend mit dem unmaskierten Kontext vorherzusagen, welche Wörter an den jeweiligen Stellen standen, wodurch das Modell Syntax und teilweise Semantik erlernt und die Embeddings entsprechend anpasst. Dabei zeigte sich zwar keine Verbesserung der späteren Ergebnisse, allerdings eine erhebliche Beschleunigung des Trainingsfortschritts, was eine effiziente Hyperparametersuche erlaubt.
Um die zentrale, für diese Arbeit entwickelte End-to-End-Netzarchitektur zu beschreiben werden zunächst einige Indizes definiert.

- z: Zeilen im Dokument
- w: (Sub-)Worte pro Zeile (Padding kürzerer Zeilen)
- d: Dimension des Modells (z. B. 150)
- e: Embedding-Dimension
- c: Anzahl Klassen

Die Netzarchitektur lässt sich dann durch Gleichung 2.3 – 2.9 beschreiben (die Batchdimension wird hierbei ausgelassen).

$$X_{zwe} = \text{Embedding}(X_{zw}) \tag{2.3}$$
$$X_{zwd} = \text{LSTM}_w(X_{zwe}) \tag{2.4}$$
$$A_{zw} = \text{softmax}_w\left(\sum_d X_{zwd} W_d^1\right) \tag{2.5}$$
$$X_{zd} = \sum_w X_{zwd} A_{zw} \tag{2.6}$$
$$A_z = \text{softmax}_z\left(\sum_d X_{zd} W_d^2\right) \tag{2.7}$$
$$X_d = \sum_z X_{zd} A_z \tag{2.8}$$
$$Y_c = X_d W_{dc}^3 \tag{2.9}$$

So läuft zunächst eine LSTM-Schicht parallel und individuell über jede Zeile im Dokument. Für diesen Schritt wird die Zeilen- als Batchaxe behandelt (d. h. beide Dimensionen werden zusammengeführt), wodurch alle Zeilen in einem einzigen Vorwärtsschritt des Netzwerks verarbeitet werden können, was die Architektur extrem schnell macht. Das Netzwerk lernt anschließend eine Gewichtung W_d^1 (in Form eines einzelnen Neurons

oder Subnetzwerks mit einzelnem Ausgabeneuron, daher eindimensional), wie wichtig jeder verdeckte Zustand der LSTM-Schicht zu jedem Subwort ist. Anschließend werden die mit der *Softmax*-Funktion normalisierten Anteile für jedes Subwort innerhalb einer Zeile bestimmt, um damit die LSTM-Zustände zu gewichten und anschließend zu einem Vektor pro Zeile X_{zd} aufzuaddieren. Das Netzwerk lernt dann eine zweite, äquivalente Gewichtung W_d^2, um damit alle Zeilen gemäß ihrer Relevanz zu gewichten und zu einem finalen Vektor, der das gesamte Dokument repräsentiert (siehe Anhang B), zu summieren. Eine letzte lineare Schicht ergibt dann die Klassen-Logits Y_c. Eine zweite LSTM-Schicht zwischen Gleichung 2.6 und 2.7, um potentiell Abhängigkeiten zwischen Zeilen besser zu erfassen, führte empirisch zu keiner Verbesserung. Durch zufällige Suche wurde $e = 100$ und $d = 150$ empirisch ermittelt.
Um extremes Padding zu vermeiden, wird mit einer Batch-Größe von eins trainiert (andernfalls müssten alle Daten im Batch auf die Anzahl des Dokuments mit den meisten Zeilen angepasst werden). Die Subwortaxe wird auf die längste Zeile im Dokument, höchstens jedoch auf 50 Subwörter mittels Padding angepasst (was 99,9% aller Zeilen einschließt). Um sowohl Speicher- als auch Zeitkomplexität zu reduzieren, wird *Automatic Mixed Precision* [6], also auf Float16 statt Float32 basierende Parameter, eingesetzt. Zudem wurde mit Normalisierung und Dropout nach Gleichung 2.6 und 2.8 experimentiert, jedoch ohne resultierende Verbesserungen.
Da sich die Häufigkeiten der Dokumente zu jedem Herkunftsland extrem unterscheiden (Tabelle 2.1), wurden verschiedene Maßnahmen für das Training untersucht. Sowohl

Land	USA	CN	UK	DE	JP	CA	FR	IL	AU	CH	SG	IN
Häufigkeit	8699	2119	1042	770	557	542	470	278	277	256	254	200

Tab. 2.1.: Häufigkeiten der Herkunftsländer im Korpus

Auslassen von Datensätzen größerer Klassen (*Downsampling*) als auch Vervielfachen von Dokumenten seltener Länder (*Upsampling*) zeigten keinen Erfolg. Erstes verschlechterte die Ergebnisse maßgeblich, letzteres führte hauptsächlich zu einer längeren Trainingsdauer. Stattdessen wurde die Loss-Funktion (Kreuzentropie) gewichtet (Gleichung 2.10). Der Wert von α wurde dabei durch Ausprobieren aller Werte von 0.0–1.0 in Zehntelschritten ermittelt.

$$W_{C_i}^L = \left(\frac{|C_{\max}|}{|C_i|}\right)^{\alpha}, \quad \alpha = 0.6 \tag{2.10}$$

Außerdem wurden verschiedene andere Architekturen ausprobiert. Generell sind Transformer der vorherrschende Architekturstandard. Da deren Komplexität jedoch quadratisch mit der Anzahl an Eingabewörtern skaliert und die Parameteranzahl von vortrainierten Modellen meist in den Gigabyte-Bereich reicht, eignen sich entsprechende Modelle nur bedingt. Zudem ist der Domänentransfer problematisch, da sich die bereinigten Textdateien mit oftmals fehlenden (oder schlichtweg nicht vorhandenen, z. B. Titel, Überschriften etc.) Satzgrenzen und zahlreichen Artefakten stark von den strukturierten Daten, auf denen die Transformer vortrainiert wurden, unterscheiden.

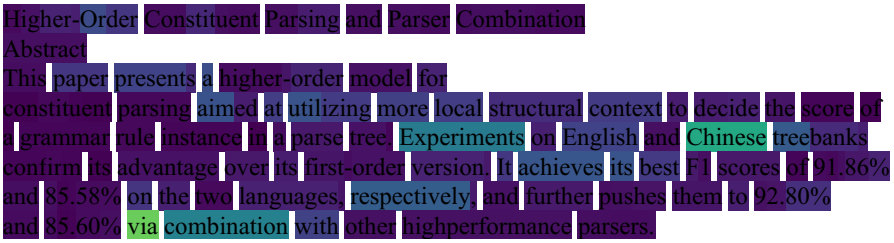
[6] `https://github.com/NVIDIA/apex`

- **CNN**: Hierbei wurden Gleichung 2.4–2.6 durch ein Satz-CNN ersetzt [13]. Anstatt den Text eindimensional in das CNN zu geben, wurde die zweidimensionale Form $z \times w$ beibehalten, was die Zeitkomplexität von $O(zw)$ maßgeblich auf $O(w)$ reduziert (dabei gehen jedoch potentiell Abhängigkeiten zwischen Zeilen verloren).

- **RoBERTa**: Dieses 125 Millionen Parameter starke, vortrainierte Sprachmodell basiert auf der Transformer-Architektur und kann Sequenzlängen bis 512 verarbeiten [15]. Der Text wird deshalb in entsprechende Stücke aufgeteilt, welche separat mit dem Modell verarbeitet werden. Anschließend läuft eine LSTM-Schicht über die gepoolte Ausgabe dieser Stücke, um das Dokument letztlich zu klassifizieren.

- **Longformer**: Dieses ebenfalls vortrainierte Modell verbessert die Transformer-Architektur, sodass Speicher- und Zeitkomplexität linear mit der Sequenzlänge skalieren [1]. Dadurch wird es möglich, gleichzeitig 4096 Subwörter zu verarbeiten, was durchschnittlich 53% einzelner Dokumente abdeckt. So werden die ersten 4096 der Subtokens mit dem Modell und einer einfachen linearen Schicht klassifiziert.

3. Ergebnisse



(a) Bereinigter Text

(b) Originaltext

Abb. 3.1.: Visualisierung der gelernten Gewichtung einzelner Subtokens zur Klassifikation des Herkunftslandes mit einem Ausschnitt des bereinigtem und ursprünglichem Text.

		Voraussage										
		AU	CA	CN	FR	DE	IN	IL	JP	SG	CH	UK	USA
Wahrheit	AU	6		8		1						7	6
	CA	1	8	4					2		1	7	32
	CN	1	1	188		1			2	1		1	17
	FR			1	36	3			1			2	4
	DE		2	2	2	47			1			12	11
	IN			1			6					2	11
	IL		1		1			14	2				10
	JP			2	1				45		1	1	6
	SG		1	6						9		2	8
	CH			3		7					4	4	8
	UK	3	5	1	6	1	2		1		2	66	18
	USA	3	13	38	17	6	1	2	8	2	3	32	746

Tab. 3.1.: Konfusionsmatrix der Klassen Australien (AU), Canada (CA), China (CN), Frankreich(FR), Deutschland (DE), Indien (IN), Israel (IL), Japan (JP), Singapur (SG), Schweitz (CH), Großbritannien (UK), Amerika (USA). Leere Zellen implizieren Nullen.

4. Zusammenfassung

asdasd

Literaturverzeichnis

[1] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer, 2020.

[2] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information, 2016.

[3] J. B. Carroll. *Language And Thought*. Prentice-Hall, 1964.

[4] M. A. Covington and J. D. McFall. Cutting the gordian knot: The moving-average type–token ratio (mattr). *Journal of Quantitative Linguistics*, 17(2):94–100, 2010.

[5] D. Dugast. Sur quoi se fonde la notion d'étendue théoretique du vocabulaire? In *Le francais moderne*, volume 46, pages 25–32, 1978.

[6] B. F. 'delta': A measure of stylistic difference and a guide to likely authorship. *Literary and Linguistic Computing*, 17:267–287, 09 2002.

[7] P. Guiraud. *Les caractères statistiques du vocabulaire: essai de méthodologie*. Presses universitaires de France, 1954.

[8] G. Herdan. *Quantitative Linguistics*. Butterworths, 1964.

[9] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[10] M. Honnibal and I. Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[11] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[12] A. Kilgarriff. Comparing corpora. *International Journal of Corpus Linguistics*, 6, 11 2001.

[13] Y. Kim. Convolutional neural networks for sentence classification, 2014.

[14] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks, 2017.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[16] H. D. Maas. Zusammenhang zwischen wortschatzumfang und länge eines textes. In *Zeitschriftfiir Literaturwissenschaft und Linguistik*, volume 8, pages 73–79, 1972.

[17] P. M. McCarthy and S. Jarvis. vocd: A theoretical and empirical evaluation. *Language Testing*, 24(4):459–488, 2007.

[18] P. M. McCarthy and S. Jarvis. Mtld, vocd-d, and hd-d: A validation study of sophisticated approaches to lexical diversity assessment. *Behavior Research Methods*, 42(2):381–392, May 2010.

[19] T. C. Mendenhall. The characteristic curves of composition. *Science*, 9(214):237–249, 1887.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[21] S. M. Mohammad. The state of nlp literature: A diachronic analysis of the acl anthology, 2019.

[22] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[23] J. A. Perez, F. Deligianni, D. Ravi, and G.-Z. Yang. Artificial intelligence and robotics, 2018.

[24] R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units, 2015.

[25] H. H. Somers. Statistical methods in literary analysis. In *The computer and literary style,*, pages 128–140, Kent State University, 1966.

[26] B. Spillner. *Linguistik und Literaturwissenschaft: Stilforschung, Rhetorik, Textlinguistik*. Reihe Kohlhammer. W. Kohlhammer, 1974.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[28] E. Stamatatos. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol.*, 60(3):538–556, Mar. 2009.

[29] M. C. Templin. *Certain Language Skills in Children: Their Development and Interrelationships*, volume 26. University of Minnesota Press, ned - new edition edition, 1957.

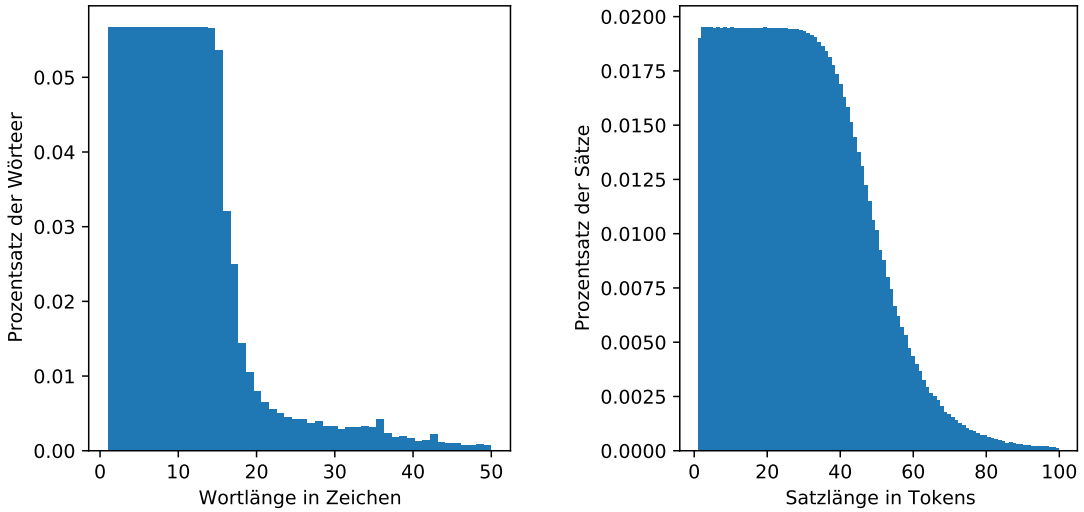
[30] J. Torruella and R. Capsada. Lexical statistics and tipological structures: A measure of lexical richness. *Procedia - Social and Behavioral Sciences*, 95:447 – 454, 2013. Corpus Resources for Descriptive and Applied Studies. Current Challenges and Future Directions: Selected Papers from the 5th International Conference on Corpus Linguistics (CILC2013).

[31] S. Vajjala, B. Majumder, A. Gupta, and H. Surana. *Practical Natural Language Processing*. O'Reilly Media Inc., 2020.

[32] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[33] R. Weischedel et al. Ontonotes release 5.0. Technical report, LDC2013T19 Web Download. Philadelphia: Linguistic Data Consortium,, 2013.

A. Wort- und Satzlängenhistogramme



(a) Wortlängen

(b) Satzlängen

Abb. A.1.: Histogramme über alle Dokumente hinsichtlich durchschnittlicher Zeichenlänge von Wörtern und Wortlänge von Sätzen. Artefakte und fehlerhafte Strukturinformationen verzerren die Ergebnisse.

B. PCA-Projektion des internen Modellzustands

Abb. B.1.: Dreidimensionale *Principal Component Analysis*-Projektion des internen Modellzustands X_d (siehe Gleichung 2.8) von den Klassen USA (blau), China (rot) und UK (rosa). Erstellt mittels `https://projector.tensorflow.org/`.