Themenspezifische Gruppierung deutscher Online-Zeitungen mit Natural Language Processing

Bachelorarbeit

Georg Donner
Matrikel-Nummer 553821

Betreuer	Prof. Dr. Gefei Zhang
Erstprüfer	Prof. Dr. Gefei Zhang
Zweitprüfer	Prof. Dr. Barne Kleinen

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Natural Language Processing ist ein großes Feld, welches besonders in der letzten Zeit im Zuge der Digitalisierung viel an Aufmerksamkeit und Wichtigkeit gewonnen hat. Es ermöglicht uns Informationen schneller zu finden, Systeme durch gesprochene Sprache zu steuern oder ganze Texte zu generieren. Eine weitere Aufgabe ist es, eine große Menge an Texten in Kategorien einzuteilen, um die Daten auf eine gewünschte Teilmenge für eine spezifischere Suche oder Analyse zu reduzieren. Die Kategorisierung der Dokumente nach ihrem Inhalt ist hier der häufigste Anwendungsfall, es ist aber auch möglich Texte nach ihrem generellen Genre oder Schreibstil zu vergleichen.

Diese Arbeit wird am Beispiel deutscher Online-Zeitungen untersuchen, welche Möglichkeiten es gibt Texte unabhängig von ihrem Inhalt zu vergleichen. Dabei werden lexikalische, morphologische und syntaktische Merkmale, aber auch die Verwendung inhaltlich irrelevanter Wörter als Features verwendet. Es wird überprüft, inwiefern die Artikel gruppiert werden können und Rückschlüsse auf Unterschiede im Schreibstil ganzer Zeitungen statt nur einzelner Artikel zulassen.

Des Weiteren wird untersucht, ob und wie sich der Schreibstil einer Zeitung je nach Thema, wie z.B. Politik und Sport, unterscheidet.

2 Grundlagen

Für die Verarbeitung und Analyse von Zeitungsartikeln sind zwei Teilgebiete der Informatik besonders wichtig: Natural Language Processing und Machine Learning. Im Folgenden werden die für die Arbeit relevantesten Konzepte der beiden Bereiche genauer erklärt.

2.1 Natural Language Processing

Natural Language Processing ist ein Teilgebiet der Informatik, das Konzepte und Techniken der künstlichen Intelligenz und des Machine Learning verwendet, um natürliche Sprache zu verarbeiten. Der Begriff natürliche Sprache wird verwendet, um menschliche Sprachen zu beschreiben, die im Gegensatz zu künstlich entwickelten Plansprachen eine historische Entwicklung durchlebt haben. Diese Sprachen befinden sich in einem dauerhaften Entwicklungsprozess und sind häufig sehr variabel in ihrer Verwendung durch den Menschen. Die Analyse der Semantik eines Wortes oder Satzes ist für Computer besonders schwierig, da sich die Bedeutung häufig erst durch den Kontext ergibt.

Mit den schnellen Fortschritten im Bereich des Machine Learning in den letzten Jahrzehnten, eröffneten sich für die Verarbeitung natürlicher Sprache jedoch völlig neue Möglichkeiten. Die Erkennung von Syntax und Semantik wurde damit immer präziser und das Teilgebiet immer relevanter. Gegenwärtig basiert dies hauptsächlich auf Algorithmen des Supervised Learning, für die die Texte vorher manuell mit relevanten Markierungen versehen werden müssen. Ein bekanntes Beispiel für einen Korpus deutscher Sprache mit solchen Annotationen ist der TIGER Corpus [1].

1 http://www.ims.uni-stuttgart.de/forschung/ressourcen/korpora/tiger.html

TODO: Verwendung nicht annotierter Korpora

2.1.1 Pipeline

Bei der Analyse eines Textes werden in der Regel verschiedene Schritte abgearbeitet, die jeweils eigene Merkmale der Sprache untersuchen. Es entsteht eine sogenannte Pipeline, die je nach Anwendungsfall unterschiedlich aussieht. Das sequenzielle Ausführen dieser einzelnen Vorgänge ist notwendig, da beispielsweise die Analyse der Syntax voraussetzt, dass das Dokument bereits in Token zerlegt wurde. Im Folgenden werden die für diese Arbeit relevanten Schritte beschrieben.

TODO: eventuell noch andere Modelle beschreiben?

Tokenisierung

Tokenisierung beschreibt den Prozess, einen gegebenen Text in gleiche Einheiten, sogenannte Token, zu zerlegen. Meistens handelt es sich dabei um Wörter und Satzzeichen, je nach Anwendungsfall können es aber auch Wortgruppen oder Sätze sein. Die Tokenisierung ist häufig eine Voraussetzung einer tiefgehenderen Analyse des Textes, daher ist eine hohe Genauigkeit hier besonders wichtig. Eine Teilung an jedem Satzzeichen um eine Liste von Sätzen zu erhalten, ist zum Beispiel eine triviale Lösung, die bereits gute Ergebnisse erzielt. Es müssen jedoch viele Sonderregeln wie Abkürzungen und Zahlen beachtet werden, sodass das Problem wesentlich komplexer ist ,als es zunächst scheint. Ein Ansatz um höhere Genauigkeit zu erreicheren, ist ein Modell auf Basis bereits mit Annotationen versehener Korpora zu trainieren, welches die Regeln automatisch erlernt.

Part-of-speech-Tagging

Das Part-of-speech-Tagging, auch POS-Tagging, ist ein Verfahren, bei dem jedem Wort oder Satzzeichen die jeweilige Wortart zugeordnet wird. Bei der Analyse ist hierbei vor allem der Kontext in dem das Wort erscheint wichtig, da sich daraus häufig erst die Bedeutung ergibt. Die Informationen über die Wortart und oft auch weitere Details, geben sogenannte

Tags, die meist aus einem festen Tagset stammen. In dieser Arbeit werden die Tags aus dem Stuttgart-Tübingen-Tagset (STTS) [2] verwendet. Der Satz „Martin findet eine grüne Blechdose.“ sieht nach dem POS-Tagging beispielsweise so aus:

Martin/NE findet/VVFIN eine/ART grüne/ADJA Blechdose/NN ./$.

Die Tags geben Auskunft über mehr als nur die Wortart. Zum Beispiel sind die beiden Wörter *Martin* und *Blechdose* jeweils Nomen. Da jedoch beim POS-Tagging auch die Definition des Wortes überprüft wird, kann *Martin* korrekt als Eigenname mit dem Tag NE identifiziert werden. Weiterhin wurde in diesem Satz die Verbform und der Adjektiv-Typ korrekt erkannt.

Dependency Parsing

Die Analyse der syntaktischen Struktur eines Satzes ist ein weiterer wichtiger Schritt in der Verarbeitung natürlicher Sprache. Beim Dependency Parsing wird zunächst jeder Satz nur auf seine Wörter reduziert, um dann die Beziehungen, sogenannte Dependency Relations, der Wörter innerhalb dieses Satzes zu bestimmen.

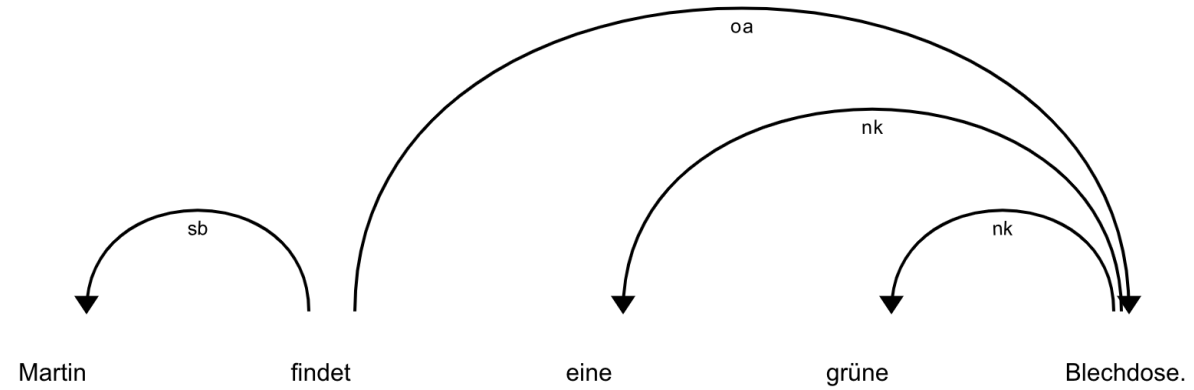

Abbildung 2.1: Visualisierung der Beziehungen mit Pfeilen

Diese Beziehungen ergeben letztendlich einen Baum, der navigiert werden kann und auch Aufschlüsse über die Komplexität eines Satzes zulässt. Die Ergebnisse des Dependency Parsing finden neben der Erkennung semantischer Beziehungen zwischen Wörtern auch noch

2 http://www.ims.uni-stuttgart.de/forschung/ressourcen/lexika/TagSets/stts-table.html

weitere Anwendungen. So werden sie z.B. von dem Natural Language Processing Tool SpaCy für die Erkennung der Satzenden verwendet [Exp19].

Lemmatisierung

Für viele Analysen eines Textes ist es von Vorteil oder sogar notwendig, dass nicht jedes Wort welches unterschiedlich geschrieben wird, als ein anderes Wort behandelt wird. Um dies zu erreichen, werden alle Wörter auf ihre Grundform reduziert. Dieser Prozess heißt Lemmatisierung. Dies ist besonders hilfreich für die Feststellung der Häufigkeit eines Wortes in einem Dokument oder Korpus. So wäre die Frequenz der Wörter *findet*, *fand* und *finden* zunächst jeweils eins. Nach der Lemmatisierung gibt es nur noch das Lemma *finden* mit einer Frequenz von drei. Dies hilft dabei, das Rauschen innerhalb eines Textes zu reduzieren und das tatsächliche Vokabular genauer zu beurteilen.

2.2 Machine Learning

Die meisten der zuvor in Kapitel 2.1.1 erläuterten Schritte basieren auf Machine Learning und auch in dieser Arbeit werden einige Verfahren aus diesem Bereich angewandt, um den Schreibstil der Texte zu ermitteln und zu analysieren. Das Ziel des Machine Learning ist es, aus bestehenden Beobachtungen ein Modell zu trainieren, welches in diesen ein Muster oder Zusammenhänge erkennen soll und anschließend Aussagen über eine bisher unbekannte Beobachtung treffen kann. In diesem Kapitel werden einige für diese Arbeit relevante Themenbereiche aus dem Machine Learning genauer beschrieben.

2.2.1 Feature Engineering

Alle Beobachtungen aus welchen ein zu trainierendes Modell lernt, haben immer eine bestimmte Anzahl an Features, die jede Beobachtung so gut wie möglich repräsentieren sollen. Die Auswahl dieser Features bildet die Grundlage des Trainings und ist ausschlaggebend für die Performance des Modells. Die Verwendung voneinander unabhängiger

Features, die mit der vorherzusagenden Klasse in Beziehung stehen, wird dabei höchstwahrscheinlich zu sehr guten Resultaten führen [Dom12]. Die Auswahl der richtigen Features ist also der Schlüssel für das erfolgreiche Lernen, aber es ist auch der schwierigste Teil. „Coming up with features is difficult, time-consuming, requires expert knowledge." [Ng13] Häufig werden viele Features generiert bzw. bestimmt, die möglicherweise nicht alle zielführend oder unabhängig sind. Hier gibt es verschiedene Prozesse, um die Anzahl an Features zu reduzieren und gleichzeitig kaum an Informationsgehalt zu verlieren. Die Auswahl von Features aus der großen Menge heißt *Feature Selection*. Die Kombination von Features, um daraus neue zu generieren, wird *Feature Extraction* genannt. Diese Reduktion ist oftmals erforderlich um das statistische Rauschen in den Daten zu minimieren und beim Training Zeit zu sparen. Besonders bei sehr rechenintensiven Algorithmen ist dies absolut notwendig.

2.2.2 Dimensionsionalitätsreduktion

Bei der Generierung von d verschiedenen Features bzw. Variablen für n Beobachtungen, entsteht ein Raum der Dimension d. Um so viele Informationen wie möglich zu jeder Beobachtung zu speichern und für z.B. eine Klassifizierung zu verwenden, werden sehr viele Features generiert, welche wiederum einen hoch-dimensionalen Raum ergeben. Dabei entsteht das Problem, dass die Datenverarbeitung immer mehr Zeit beansprucht, da auch wesentlich mehr Beobachtungen notwendig sind. Weiterhin führt es zu Schwierigkeiten bei der Bestimmung von Distanzen zwischen einzelnen Datenpunkten. In solch einem Raum ist es unvermeidbar, dass die Verteilung der Punkte sehr dünn ist. Darauf basierend wurde festgestellt, dass in einem hoch-dimensionalen Raum alle Punkte annähernd die gleiche Entfernung haben und die Suche nach dem nächsten Nachbarn nur sehr ungenau ist [HAK00]. Eine häufige Annahme bei der Betrachtung dieser Räume ist, dass die Variablen nicht alle voneinander unabhängig sind und die Anzahl an Dimensionen reduziert werden kann, ohne dabei einen signifikanten Anteil an Informationen zu verlieren. Ein dafür häufig verwendetes Verfahren ist die *Hauptkomponentenanalyse* oder englisch *Principal Component Analysis (PCA)*. Hierbei werden iterativ, je nach gewünschter Anzahl an Features, Linearkombinationen der Variablen erstellt, die durch Maximierung der Varianz der Daten in die entsprechende Richtung bestimmt werden und so jeweils die höchste Aussagekraft besitzen. Bei einer Reduktion auf zwei oder drei Dimensionen können die

Daten dann gut visualisiert werden, dabei gehen aber meistens zu viele Informationen verloren.

2.2.3 Klassifizierung

Um die Performance der Features messen zu können, wird in dieser Arbeit Klassifizierung verwendet. Dafür werden alle Beobachtungen zunächst in ein Trainings- und ein Testset unterteilt, um das trainierte Modell nach dem Lernen mit den Testdaten beurteilen zu können. Nach dem Training ist die Aufgabe des Modells, jeder neuen Beobachtung aus dem Testset eine Klasse aus einer Liste bereits bekannter Klassen zuzuordnen. Die Klassen der jeweiligen Beobachtungen aus dem Trainingsset sind bekannt.

Falls es nur zwei mögliche Klassen gibt, handelt es sich um eine *Binärer Klassifikation*. Eine häufige Herangehensweise bei mehr als zwei Klassen ist es, die Aufgabe in mehrere binäre Probleme zu zerlegen, bei der nach der *One-vs.-all* Strategie entschieden wird, welche Klasse zugewiesen wird. Dabei wird für jede mögliche Klasse eine Wahrscheinlichkeit bzw. Sicherheit berechnet und anschließend gewinnt die mit dem höchsten Wert.

Klassifizierung ist ein häufiges Problem, für das es aufgrund der vielen Anwendungsbereiche und damit völlig unterschiedlichen Anforderungen keinen besten Algorithmus zur Bestimmung der Klasse bzw. der Wahrscheinlichkeit gibt. Die meisten dieser Algorithmen erstellen eine lineare Funktion, in die der Featurevektor anschließend eingesetzt wird. Jedes Feature bekommt hierbei eine eigene Gewichtung, die sich aus dem vorherigen Training ergibt. Dieses Training unterscheidet sich je nach Algorithmus und auch der zurückgegebene Score wird je nach Algorithmus unterschiedlich interpretiert.

3 Datenverarbeitung

Die Grundlage dieser Arbeit bildet ein großer Datensatz mit Artikeln verschiedener Online-Zeitungen. Damit diese Artikel verglichen werden können oder eine Klassifizierung durchgeführt werden kann, müssen diese zunächst jeweils in einen Featurevektor umgewandelt werden. Dieses Kapitel beschreibt die verwendeten Tools und nötigen Schritte um dieses Ziel zu erreichen.

3.1 Verwendete Tools

3.1.1 Python

Python ist eine sehr universelle Programmiersprache und findet in den unterschiedlichsten Bereichen Anwendung. Die Syntax ist einfach zu erlernen und legt besonderen Wert auf gute Lesbarkeit, was wiederum die Produktivität der Programmierer erhöht [Fou19]. Des Weiteren gibt es eine große Auswahl an Bibliotheken für viele verschiedene Anwendungsfälle. Besonders in der wissenschaftlichen Arbeit hat sich die Verwendung von Python zusammen mit den Packages SciPy, NumPy, Matplotlib und Pandas bewährt. Auf Basis dieser Kombination entwickelte sich die Anaconda Distribution [1], welche die wichtigsten Bibliotheken für Data Science und Machine Learning beinhaltet und die Verwaltung dieser pro Projekt erleichtert. Die Verwendung einer solchen Distribution ermöglicht es, out-of-the-box Optimierungen mathematischer Berechnungen durchzuführen, unter anderem auf Basis der Intel™ Math Kernel Library (MKL). Dadurch konnte in dieser Arbeit zum Beispiel die Analyse der Artikel mit spaCy um etwa 27% beschleunigt werden.

1 https://www.anaconda.com/what-is-anaconda/

3.1.2 SpaCy

Auch für Natural Language Processing gibt es eine Vielzahl an Python Bibliotheken, die jedoch meist unterschiedliche Aufgaben erfüllen. *SpaCy* ist ein exzellentes Open-Source-Tool, welches die in Kapitel 2.1.1 beschriebenen Schritte der Pipeline vollständig abdeckt. Es legt besonderen Wert auf die Genauigkeit der Resultate und die Geschwindigkeit der Berechnungen, unter anderem durch die Verwendung und Optimierung des seiner Meinung nach besten Algorithmus für die jeweilige Aufgabe, statt mehrere Alternativen zur Verfügung zu stellen. SpaCy stellt trainierte Modelle für derzeit sieben verschiedene Sprachen zur Verfügung, weitere befinden sich bereits in der Entwicklung. Deutsch wurde im Jahr 2016 als erste Fremdsprache hinzugefügt und liefert trotz der reichhaltigeren Morphologie der Sprache sehr gute Ergebnisse [See16].

Eine häufig verwendete Alternative zu spaCy ist NLTK, kurz für Natural Language Toolkit. Es wurde in dieser Arbeit an einigen Stellen eingesetzt und für die Berechnung der linguistischen Merkmale in Betracht gezogen. Für die deutsche Sprache existiert hier jedoch kein fertiges Modell für das POS-Tagging, dieses müsste auf Basis eines annotierten Korpus selbst trainiert werden. Des Weiteren unterstützt NLTK kein Dependency Parsing, ein wichtiger Schritt der in dieser Arbeit verwendeten Pipeline. Die Verwendung und Ergebnisse dieser beiden Bibliotheken werden in Kapitel 3.3 genauer untersucht.

3.1.3 Scikit-learn

Scikit-learn ist eine Python Bibliothek, welche eine Vielzahl an effizienten Algorithmen aus dem Bereich des Machine Learning zur Verfügung stellt [PVG+11]. Sie baut auf den bereits in Kapitel 3.1.1 genannten Bibliotheken SciPy, NumPy und Matplotlib auf und bietet ebenfalls eine nahtlose Integration mit ihnen. Zudem gibt es eine sehr ausführliche und einsteigerfreundliche Dokumentation, die über die Beschreibung der in der Bibliothek enthaltenen Funktionen hinaus auch ausführliche Tutorials beinhaltet.

3.2 Datenselektion

Bevor die gegebenen Daten für das Extrahieren von Features verwendet werden, ist es ratsam zunächst einen Überblick über die Struktur zu bekommen. Häufig befinden sich fehlerhafte Daten im Datensatz und auch das Format entspricht nicht immer den Erwartungen oder ist unregelmäßig. Die in dieser Arbeit getroffenen Maßnahmen, um den Datensatz bereit für die Textverarbeitung zu machen, werden in diesem Kapitel genauer erläutert.

3.2.1 Datensatz

Der in dieser Arbeit verwendete Datensatz wurde von Ole Wendt erstellt und enthält etwas mehr als zehn Millionen Artikel bis Juni 2018 für neun verschiedene Zeitungen. Tabelle 3.1 lässt jedoch erkennen, dass die Anzahl der Artikel je nach Zeitung sehr unterschiedlich ist. Für die Speicherung der Daten wurde das relationale Datenbanksystem *SQLite* gewählt.

Zeitung	Anzahl der Artikel
Ärzte-Zeitung	132753
Handelsblatt	942362
RP Online	1549537
SHZ	347335
Spiegel Online	620213
Der Tagesspiegel	975483
Westfälischer Anzeiger	318910
Welt	2980800
Zeit Online	2242198

Tabelle 3.1: Größe des Datensatzes

Die für diese Arbeit relevanten Datenfelder eines Artikels sind `content`, `url` und `date`, welche jeweils in Textform gespeichert sind. Es existieren weitere interessante Felder für z.B. den Autor oder die Zusammenfassung eines Artikels. Ein genauerer Blick in den Datensatz ergibt jedoch, dass diese bei dem Großteil der Einträge fehlen und somit nicht für die Analyse genutzt werden können.

3.2.2 Normalisierung

Für die Normalisierung der Daten werden zunächst alle Artikel entfernt, die weniger als 100 Zeichen haben. Dies ist notwendig, da ein kleiner Teil der Artikel entweder gar keinen Inhalt besitzt oder nur eine kurze Eilmeldung ist.

Eine weitere Auffälligkeit bei der Betrachtung des Datensatzes ist, dass das Datum je nach Zeitung in einem unterschiedlichen Format gespeichert ist. Zudem müssen erneut Artikel ohne Datum herausgefiltert werden. Die Sortierung der Artikel nach Datum funktioniert trotzdem sehr zuverlässig, sodass die Artikel erst sortiert und anschließend das Datum formatiert werden kann. Obwohl das Datum später nicht direkt weiterverarbeitet wird, ist ein einheitliches Format, in dieser Arbeit wird der ISO 8601 Standard verwendet, dennoch wichtig für die Visualisierung der Daten. Für diesen Schritt ist die Python Funktion `datetime.strptime` sehr hilfreich, da damit auch Daten wie „1. März 2018, 9:11 Uhr“ einfach geparst werden können.

Die Klassifizierung der Zeitungen soll *themenspezifisch* erfolgen, das heißt jedem Artikel soll idealerweise ein Thema bzw. eine Kategorie zugeordnet werden. Dafür werden im Rahmen dieser Arbeit sehr grobe Themen wie Politik, Wirtschaft oder Sport angestrebt. Für die Bestimmung dieser Themen bietet sich die URL des Artikels an, da die Webseiten der Zeitungen, wie Abbildung 3.1 verdeutlicht, meist nach solchen Themen strukturiert sind. Dies spiegelt sich dann im Format der URL wider.

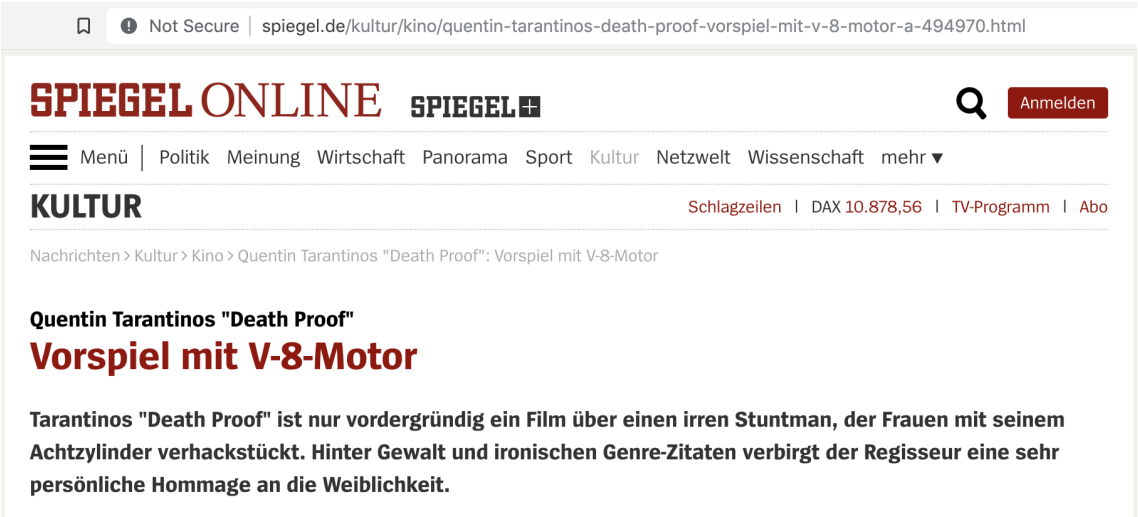

Abbildung 3.1: Die Struktur der Spiegel Online Webseite [Bor07]

Je nach Zeitung ergeben sich hierbei unterschiedliche Themen und nicht jedem Artikel kann auf diese Weise zuverlässig ein Thema zugeordnet werden. Zudem unterscheidet sich die Anzahl der Artikel je nach Thema stark, sodass einige aufgrund zu kleiner Repräsentation aussortiert werden.

3.3 Textverarbeitung

Bei der Textverarbeitung durchläuft eine Auswahl an Artikeln die in Kapitel 2.1.1 beschriebene Pipeline und die Ergebnisse werden dann für die weitere Verwendung zwischengespeichert.

Der erste Ansatz für die Verarbeitung der Texte war es, für jeden Schritt in der Pipeline ein explizit für dieses Verfahren entwickeltes Tool zu verwenden. Für die Tokenisierung wurde hier zunächst die populäre Bibliothek NLTK verwendet, welche bereits sehr gute Resultate bei annehmbarer Performanz lieferte. Für das POS-Tagging wurde der von Helmut Schmid entwickelte TreeTagger [2] verwendet, welcher zusätzlich das Lemma jedes Wortes bestimmt. Dieser unterstützt eine Vielzahl an Sprachen, da pro Sprache lediglich eine Parameterdatei notwendig ist. Die Verwendung des TreeTaggers bringt jedoch auch einige Nachteile mit sich. Er wird nur als Perl Skript zur Verfügung gestellt und kann so nur über Umwege in die Python Pipeline integriert werden. Zudem ist die Startzeit des Skriptes pro Artikel sehr hoch, sodass die Berechnung zu zeitaufwendig wird. Dies kann umgangen werden, indem die Artikel zuvor zu einem großen Text zusammengefügt und anschließend wieder getrennt werden und führte zu einer sehr hohen Performanz. Dadurch werden jedoch Ungenauigkeiten in den Ergebnissen riskiert und an einigen Stellen stoppte das Skript ohne einen Fehler zu werfen. Ein weiteres Problem dieses Ansatzes ist, dass durch die Ausführung der Pipeline in einzelnen Schritten die Ergebnisse jedes Schrittes zwischengespeichert werden müssen und eventuell später zu groß für den Arbeitsspeicher werden.

Im zweiten Ansatz wurde mehr Wert auf die Zuverlässigkeit der Ergebnisse statt der Optimierung der Geschwindigkeit des Prozesses gelegt. Hierfür wurden zunächst 1000 Artikel pro Zeitung pro Thema verwendet, insgesamt 43000 Artikel, und mit der Bibliothek spaCy analysiert. Die Erstellung und Ausführung einer Pipeline mit spaCy ist sehr intuitiv

2 http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/

und erfordert kaum Setup. Dennoch gibt es auch hier Möglichkeiten, die Performanz zu optimieren.

TODO: Überblick spaCy Geschwindigkeit und Optimierungen

Die Anzahl der analysierten Artikel wurde zunächst beschränkt, um die Größe der Dateien, welche die Ergebnisse der Textverarbeitung beinhalten, gering zu halten. Für 43000 Artikel werden hier etwa 3,4 GB benötigt, indem nur die für diese Arbeit relevanten linguistischen Merkmale der Token gespeichert werden. Dies ermöglicht einen schnellen, iterativen Prozess für die Generierung, Auswahl und Bewertung der in Kapitel 3.4 beschriebenen Features.

Nach der Analyse der Texte wurde eine weitere Filterung der Artikel vorgenommen, um den Anteil fehlerhafter Daten zu reduzieren. Es wurden hierfür jegliche Sätze mit weniger als vier Wörtern entfernt. Diese rigorose Filterung war notwendig da viele Artikel Wörter beinhalten, die nicht zum eigentlichen Inhalt gehören. Beispiele dafür sind Kürzel der Nachrichtenagentur, der Ort, der Autor, Quellen oder Verweise am Anfang oder Ende des Artikels. Zudem wurden Artikel der Nachrichtenagentur dpa herausgefiltert, um nur die Artikel von eigenen Autoren der Zeitungen zu vergleichen.

TODO: Bild zum Anteil der dpa Artikel

3.4 Featuregenerierung

Es gibt eine Vielzahl an Features, die man für einen Text generieren bzw. auswählen kann, ohne dass dabei der Inhalt des Artikels einen Einfluss hat. Verweis zu Studien, die analysiert haben, welche Kombination an Features die besten Resultate liefern.

3.4.1 Token-basiert

Durchschnittliche Satzlänge, durchschnittliche Wortlänge?

Frequenz der n häufigsten Wörter (bei mir 30, 50, 100). Sind dann Wörter wie ['der', 'ich', 'dieser']

3.4.2 Lexikalisch

Type-token ratio: Beurteilt die Reichhaltigkeit des Vokabulars, dafür gibt es noch mehr präzisere Indizes (in der Studie von 2000 sind einige Beispiele). Nachteil: funktioniert besser, je länger die Texte sind und die Artikel sind durchschnittlich nicht besonders lang. Illustration dazu?

Readability: Wieder gibt es hier viele verschiedene Indizes. In dieser Arbeit verwendet wurde der Flesch-Reading Ease Index unter Berücksichtigung dass die Sprache Deutsch ist (gibt noch viel mehr für Englisch). Erforderte die Berechnung, wieviele Silben ein Wort hat (war in der Pipeline nicht mit drin)

3.4.3 Morphologisch

POS unigrams: Die Frequenz der einzelnen Wortarten für jeden Artikel, ist teilweise nur sehr gering und auch hier erhöht sich die Aussagekraft mit der Länge des Artikels.

Lexical Density: Wieviele Wörter des Artikels tragen zum Inhalt bei (Nomen, Verben, Adjektive, Adverbien) im Verhältnis zur Gesamtanzahl an Wörtern? Ähnlich zur Stopword Frequency, die auch berechnet wurde, aber zu sehr mit der lexikalischen Dichte korreliert.

3.4.4 Syntaktisch

Dependency relation unigrams: Die Frequenz der einzelnen Dependency relations für jeden Artikel, ist teilweise nur sehr gering und auch hier erhöht sich die Aussagekraft mit der Länge des Artikels.

4 Datenauswertung

Wie können die Daten überhaupt ausgewertet werden? Es müssen gleichzeitig die Performance der Features, als auch des Algorithmus zur Klassifizierung analysiert und ausgewertet werden.

4.1 Überblick

Ein paar Plots zeigen, die die Korrelation ausgewählter Features zeigen?

4.2 Klassifizierung

Warum versuche ich die Klassifizierung? Weil wenn Klassifizierung nicht wirklich funktioniert und kein Model trainiert werden kann, welches die Zeitung auf Basis der Daten predicten kann, dann ist auch keine Gruppierung möglich.

Ist zudem eine sehr gute Methode dafür festzustellen, an welchen Features man die Zeitungen am besten voneinander unterscheiden kann und welche eventuell völlig unnötig sind und so bei der Gruppierung nur stören würden.

4.2.1 Vorbereitung des Datensatzes

Für das Training muss der Datensatz in Train/Testset aufgeteilt werden. Welche Aufteilung habe ich hier gewählt und warum?

Die Daten müssen vorher normalisiert oder standardisiert werden. Kommt allerdings auch auf die ausgewählten Features und den gewählten Algorithmus an.

4.2.2 Feature Selection

Welche Verfahren wurden zur Feature Selection verwendet? Warum? Warum war das überhaupt nötig. Vergleich zu bzw. Verweis auf bisherige(n) Studien zur Feature Selection zur Genre/Author detection.

(evtl. welche Features sind pro Zeitung relevant gewesen? weil unterschiedliche thetas pro Kategorie bei One vs Rest Klassifizierung)

4.2.3 Messung der Performance

Welche verschiedenen Ansätze gibt es hier?

F-Maß

Wahrheitsmatrix

Beurteilung der Wahrscheinlichkeiten

Wahrheitsmatrix kann auch mit den probabilities gemacht werden. Sagt noch mehr über die Sicherheit der Vorhersagen aus.

4.2.4 Verwendete Verfahren

Warum müssen verschiedene Verfahren getestet werden?

Linear Discriminant Analysis

Logistische Regression

Random Forest

Support Vector Machines

4.3 Clustering

Zuerst wurde untersucht, ob beim Clustering der einzelnen Artikeln jeder Zeitung Cluster entstehen, welche die einzelnen Zeitungen repräsentieren. Weiterhin wird überprüft, ob sich dabei Cluster ergeben, die Artikel verschiedener Zeitung haben. Das wäre dann schon ein sehr guter Indikator dafür, dass zwei oder mehr Zeitungen einen ähnlichen Schreibstil haben.

Ein weiterer Ansatz ist es, den Durchschnitt jeder Zeitung zu berechnen und anschließend die Zeitungen zu clustern. Dies hat jedoch Nachteile: Der Durchschnitt einer Zeitung ist nicht besonders repräsentativ, besonders wenn die Standardabweichung hoch ist. Zudem gibt es hier je nach Kategorie nur etwa 8 oder weniger "Beobachtungen"die geclustert werden können.

4.3.1 Feature Extraction

Wie in den Grundlagen bereits beschrieben, ist es beim Clustering besonders wichtig, dass die Dimensionalität nicht hoch ist. Vor allem für die Visualisierung ist es notwendig, die Features auf zwei Dimensionen zu reduzieren.

PCA

Hauptkomponentenanalyse: welche Ergebnisse gibt es hier?

t-SNE

Ein Verfahren, was genau dafür gedacht ist visuell Cluster zu zeigen, bei denen die Distanz untereinander im Plot keine Aussagekraft hat. Wie sehen hier die Ergebnisse aus? Spoiler: KACKEE

5 Ergebnis

Literaturverzeichnis

[Bor07] Borcholte, Andreas: *Quentin Tarantinos "Death Proof": Vorspiel mit V-8-Motor*. `http://www.spiegel.de/kultur/kino/quentin-tarantinos-death-proof-vorspiel-mit-v-8-motor-a-494970.html`, Juli 2007. – letzter Zugriff: 17.01.2019

[Dom12] Domingos, Pedro: A few useful things to know about machine learning. In: *Communications of the ACM* 55 (2012), Nr. 10, S. 78–87

[Exp19] ExplosionAI: *Linguistic Features - spaCy Usage Documentation*. `https://spacy.io/usage/linguistic-features#dependency-parse`, 2019. – letzter Zugriff: 12.01.2019

[Fou19] Foundation, Python: *What Is Python? Executive Summary.* `https://www.python.org/doc/essays/blurb`, 2019. – letzter Zugriff: 15.01.2019

[HAK00] Hinneburg, Alexander; Aggarwal, Charu C.; Keim, Daniel A.: What is the nearest neighbor in high dimensional spaces? In: *26th Internat. Conference on Very Large Databases*, 2000, S. 506–515

[Ng13] Ng, Andrew: *Machine Learning and AI via Brain simulations*. 2013

[PVG+11] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[See16] Seeker, Wolfgang: *spaCy now speaks German*. `https://explosion.ai/blog/german-model`, 2016. – letzter Zugriff: 16.01.2019

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

Ort, Datum Unterschrift