Technical Report

Untersuchungen zur Analyse von deutschsprachigen Textdaten

Katharina Morik, Alexander Jung, Jan Weckwerth, Stefan Rötner, Sibylle Hess, Sebastian Buschjäger, Lukas Pfahler

12/2015

technische universität dortmund

SFB 876 Verfügbarkeit von Information durch Analyse unter Ressourcenbeschränkung

Part of the work on this technical report has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis", project A1.

Speaker: Prof. Dr. Katharina Morik
Address: TU Dortmund University
Joseph-von-Fraunhofer-Str. 23
D-44227 Dortmund
Web: http://sfb876.tu-dortmund.de

Inhaltsverzeichnis

Kapitel 1

Die Vorlesung

von Katharina Morik

Die Verarbeitung natürlicher Sprache hat sich durch die großen Mengen digitalisierter Texte grundlegend gewandelt. Die meisten Daten im WWW sind Texte. Gerade durch sprachliche Daten ist *Big Data* gegeben. Im Sinne der Informationsgewinnung durch Datenanalyse, wie sie im Sonderforschungsbereich 876 [1] vorangetrieben wird, können Verfahren des maschinellen Lernens auf die Textdaten angewandt werden.

- Wurden früher Texte durch Annotationen, etwa in XML, erschlossen, wird nun die Erkennung von Eigennamen aus einigen annotierten Sätzen maschinell gelernt. Diese Lernaufgabe wird als *Named Entity Recognition* bezeichnet [26, 66].
  - Anhand annotierter Texte können dann Dienste entwickelt werden. Beispielsweise werden Gesetze oder Verordnungen indexiert und auf Bürgerplattformen ausgerichtet nach unterschiedlichen Anliegen geordnet zur Verfügung gestellt. Allgemein werden Geschäftsprozesse modelliert. So wird etwa die Weiterleitung von Mails automatisiert oder die Angebotserstellung wird unterstützt [29].
  - Aus Webseiten werden Ereignisse mit Datum, Ort, Beteiligten extrahiert, so dass Fragen beantwortet werden können.
  - XML-Annotationen können auch aus Layout-Informationen von Texten gelernt werden [33].
- Wurde früher die syntaktische Struktur von Sprachen durch Grammatiken beschrieben, die nur anhand einzelner Beispielsätze überprüft und in wohlgeformte und Sternchen-Sätze eingeteilt werden konnten, so können heute Syntaxbäume gespeichert und dann zur syntaktischen Analyse verwendet werden [48]. Dafür können unterschiedliche Formalismen genutzt werden. Sind Konstitutentenstrukturgrammatiken besser als Dependenzgrammatiken [41]? Allgemeine Parser werden durch bestimmte Baumbanken auf eine natürliche Sprache [45] oder ein bestimmtes Themengebiet [21] hin trainiert.

[1] http://sfb876.tu-dortmund.de/index.html

- Die Repräsentation von Texten geschieht meist als *Bag of Words* wobei meist jedem Wort des Vokabulars für einen Text seine Häufigkeit und bezogen auf die Dokumentensammlung seine inverse Dokumentenhäufigkeit zugeordnet wird [69]. Die semantische Vektorrepräsentation von Google geht über die einfachen Häufigkeiten hinaus [71, 80, 50, 51, 52].

- Die Untersuchung von Texten zur *Sentiment Analysis* untersucht unterschiedliche Textsammlungen im Hinblick auf spezifische Unterschiede in der Verteilung von Wörtern. Zeichnen sich unterschiedliche Genres oder unterschiedliche politische Haltungen durch unterschiedliche Verteilungen von Wörtern aus? Hat sich die Verteilung über die Zeit hinweg verändert? Solche Fragen sind für die Sprachbetrachtung interessant und werden in der Korpuslinguistik untersucht [2]

Die Vorlesung versuchte ein neues Lehrkonzept. Eine übliche Vorlesung besteht aus ein- oder zweimal wöchentlich 90 Minuten Rede der Professorin. Dazu geht sie Grundlagen nach, vergleicht Aussagen unterschiedlicher Autoren, vereinheitlicht die Notation, räumt Unstimmigkeiten aus – eine Arbeit, die beim ersten Durchgang bis zu 30 Stunden für 90 Minuten Vorlesung dauert. Die Studierenden finden den Stoff üblicherweise in der Stunde einleuchtend und stoßen erst bei den Prüfungsvorbereitungen auf Fragen. Eigentlich müssen sie jede Sitzung vor- und nachbereiten, aber selbst dann verbirgt die geglättete Präsentation einige Tücken, die aber gerade das tiefere Verständnis des Stoffes ausmachen. Die Übungen sind meist Aufgaben für eine Woche und voneinander isoliert.

In dieser Vorlesung wurde nun versucht, die Studierenden mehr an der Arbeit mit den aktuellen Artikeln zu beteiligen und ihnen eine aktivere Rolle zuzugestehen, ohne sie allein zu lassen. Insbesondere wurden Übungen und Vorlesung enger verzahnt.

- Einige Inhalte wurden per Vortrag der Professorin vermittelt, insbesondere solche, die einen Überblick geben. Auch konnten spontan Inhalte vermittelt werden, die benötigt wurden und den Studierenden noch nicht bekannt waren.

- Einige Originalartikel wurden von allen Studierenden gelesen und in der Vorlesungsstunde besprochen.

- Einige vertiefende Inhalte wurden von Gruppen von Studierenden erarbeitet und präsentiert. Da die Themen aufeinander bezogen waren, konnten alle miteinander diskutieren.

- Die Studierenden suchten sich ein Thema aus, zu dem sie kleine *Studien* anfertigten. Anhand von Literatur sollte eine Hypothese aufgestellt, anhand eigener Experimente getestet und das Ergebnis auf die Literatur zurück bezogen werden.

Für die Vorlesung wurden Werkzeuge bereit gestellt. Das System RapidMiner ist *open source* (Java). Dazu haben wir auch noch eine ganze Reihe an Tools, die für die Verarbeitung natürlicher Sprache nützlich sind, mit RapidMiner verbunden. Insbesondere gibt es

[2] Das Projekt KobRA, in dem Angelika Storrer (Universität Mannheim) und Katharina Morik gemeinsam mit Kollegen vom Institut für deutsche Sprache, der Berlin-Brandenburgischen Akademie der Wissenschaften und der Universität Tübingen gearbeitet haben, wurde 2012 – 2015 vom Bundesministerium für Forschung und Entwicklung gefördert.

die speziellen Daten und Tools der Linguistik des Deutschen, damit wir nicht immer nur englische Texte betrachten: Digitales Wörterbuch der Deutschen Sprache (DWDS) mit vielen Informationen zu Wörtern und ihren Vorkommen. Das Google Tool Word2Vector wurde ebenfalls bereit gestellt. Als Datensätze standen ein Ausschnitt aus Wikipedia, eine Sammlung von Twitter Tweets und Texte aus dem RSS der *Welt* sowie einige Newsgroup Beiträge bereit.

Einige der Studien sind hier in leicht überarbeiteter Fassung zusammen gestellt. Wir hoffen, dass andere Studierende davon einen Nutzen haben!

Kapitel 2

Named Entity Recognition mit Wikipedia Trainingskorpus

von Alexander Jung

2.1 Einleitung

Named Entity Recognition (NER) beschäftigt sich mit der Erkennung von Eigennamen (siehe Kapitel 2.2.2) und ist ein wichtiger erster Schritt vor anderen Aufgaben der natürlichen Sprachverarbeitung. Klassifikatoren werden für diese Aufgabe typischerweise überwacht oder – wie beim Bootstrapping [66, S. 60-61] – teilüberwacht trainiert. Dies erfordert jedoch einen ganz oder teilweise annotierten Korpus, dessen manuelle Erstellung mit hohen Kosten verbunden ist. Ein Rückgriff auf vorgefertigte Korpora ist im Deutschen nur eingeschränkt möglich. Zur Verfügung steht etwa der CoNLL 2003 Korpus [76, S. 143], auf den der Zugriff aber restriktiv gehandhabt wird. Eine Alternative stellt der GermEval 2014 Korpus[1] dar, welcher seit kurzem zur Verfügung steht. Eine weitere Möglichkeit bildet die Verwendung von Web-Resourcen, welche anhand von Regelmäßigkeiten in ihren Daten automatisiert annotiert werden. Beliebt ist in diesem Zusammenhang die Wikipedia, welche über mehr als eine Million Artikel [1] und 50 Millionen interne Links verfügt, welche frei zum Download angeboten werden. In dieser Arbeit wird untersucht, inwiefern sich diese nicht-annotierten Daten als Grundlage für das Trainieren eines Klassifikators zum Erkennen von *Personeneigennamen* verwenden lassen. Die Einschränkung auf Personen wird gewählt, da Artikel über diese vergleichsweise leicht und eindeutig in der Wikipedia identifiziert werden können (siehe Kapitel 2.3.2). Darüber hinaus soll geprüft werden, ob die Daten aus der Wikipedia zur Generierung von Listen häufiger Eigenname (Gazetteers) hinreichend sind und die Klassifikationsleistung verbessern.

Für das Deutsche hat McCallum [46, S. 191] mit Conditional Random Fields, welche nachfolgend verwendet werden, eine Güte von 72,17 (auf CoNLL 2003, nur Personeneigennamen) erreicht, gemessen als F1-Wert (Definition folgt). Chrupała und Klakow erzielen 87,85 (ebenfalls auf CoNLL 2003 und nur Personeneigennamen) und greifen zur Unterstützung auf Wikipedia-Daten zurück [18, S. 4]. Toral und Muñoz stellen einen

[1] `https://sites.google.com/site/germeval2014ner/`

Ansatz vor, um aus Wikipedia-Artikeln beliebig viele Gazetteers abzuleiten, arbeiten dabei aber mit dem englischen WordNet [77, S. 57-58]. Notham et. al. beschäftigen sich mit Techniken zum Ableiten von mehrsprachigen NER-Korpora aus der Wikipedia [61, S. 155ff.] und wenden dabei einen Vorverarbeitungsansatz an, der dem in dieser Studie ähnelt. Auf Nothams Arbeit bauen Al-Rfou et. al. auf und entwerfen ein abstrakteres Verfahren mit dem sie NER-Korpora in 40 Sprachen generieren [2, S. 3-5]. Über Wikipedia hinaus gehen Whitelaw et. al., die anhand von Listen und Tabellen Named Entities im gesamten Web suchen [82, S. 124-126].

2.2 Grundlagen

Nachfolgend werden Grundlagen besprochen, welche für diese Studie relevant sind. Begonnen wird mit einigen Definitionen (Kapitel 2.2.1), einer kurzen Erläuterung des Begriffs „Named Entity Recognition“ (Kapitel 2.2.2) und einer Beschreibung des verwendeten Klassifikators (Kapitel 2.2.3). Danach wird auf Konzepte eingegangen, welche im Rahmen der Erzeugung von Merkmalen Bedeutung haben. Dazu gehören die Berechnung von Wortvektoren (Kapitel 2.2.4), die Generierung von Brown-Clustern (Kapitel 2.2.5) und das LDA-Modell (Kapitel 2.2.6).

2.2.1 Definitionen

Im weiteren Verlauf dieser Studie werden folgende Definitionen verwendet:

- *Wort* (auch: *Token*): Eine zusammenhängende Zeichenkette, welche keine Tabulatoren, Zeilenumbrüche oder Leerzeichen enthält. Auch ein Punkte oder beispielsweise eine Jahreszahl können demnach ein Wort sein.
- *Dokument*: Eine Sequenz von Wörtern. Das selbe Wort darf in einer solchen Sequenz mehrfach vorkommen.
- *Korpus*: Eine Menge von Dokumenten.
- *Precision*: Ein Bewertungsmaß für Klassifikatoren. Gibt unter allen Elementen, denen der Klassifikator ein Label y zugewiesen hat, den Anteil derjenigen an, die laut Validierungsdaten auch tatsächlich das Label y haben sollten. Es wird demnach die Formel
$$Prec(h) = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} = \frac{f_{++}}{f_{++} + f_{+-}} \tag{2.1}$$
verwendet. [35, S. 29]
- *Recall* Ein weiteres Bewertungsmaß für Klassifikatoren. Gibt unter allen Elementen, welche laut Validierungsdaten das Label y haben, den Anteil derjenigen an, denen der Klassifikator y zugewiesen hat. Anhand von
$$Rec(h) = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} = \frac{f_{++}}{f_{++} + f_{-+}} \tag{2.2}$$

erfolgt die Berechnung. [35, S. 29]

- *F1-Wert*: Ein allgemeines Maß für die Qualität eines Klassifikators. Es stellt eine Mischung zwischen Precision und Recall dar. Der Wert wird anhand von

$$F_1(h) = \frac{2 \cdot \text{true positives}}{2 \cdot \text{true positives} + \text{false positives} + \text{false negatives}} \tag{2.3}$$

  berechnet. [35, S. 30]

- *Eigenname*: Ein Name für eine einzelne Entität. *Eigen*namen repräsentieren daher in dieser Studie nicht Gruppen von Entitäten (wie bspw. im Falle von „A380“, „Königspython“ oder ähnliche).

- *Gazetteer*: Ein Verzeichnis von Wörtern, welche häufig als Eigennamen verwendet werden. (Im Kontext dieser Studie nur *Personen*eigennamen.)

- *One-Hot-Repräsentation* bzw. *1-of-N*: Eine Repräsentation eines multinomialen Wertebereichs anhand eines Vektors, dessen Dimensionalität der Anzahl der möglichen Werte entspricht und der stets an nur einer einzigen Stelle eine 1 enthält, sodass wiederum jede Komponente des Vektors einen Wert des Wertebereichs repräsentiert. Beispiel:

$$\{\text{klein}, \text{mittel}, \text{groß}\} \rightarrow \{(1,0,0)^T, (0,1,0)^T, (0,0,1)^T\}$$

2.2.2 Named Entity Recognition

Eine der häufigsten Aufgaben aus dem Bereich des Information Extractions ist die sogenannte Named Entity Recognition (NER). Gegenstand dieser ist es, Eigennamen in Dokumenten zu identifizieren und der richtigen Klasse zuzuordnen [26, S. 8]. Die Auswahl der verfügbaren Klassen variiert, häufig werden jedoch diejenigen der CoNLL 2003 verwendet, welche zwischen Personen (kurz „PER“), Orten (kurz „LOC“), Organisationen (kurz „ORG“) und sonstigen Eigennamen (kurz „MISC“) unterscheiden [76, S. 143]. Ein dazu passendes Beispiel [76, S. 142] ist

$$[ORG \text{ U.N.}] \text{ official } [PER \text{ Ekeus}] \text{ heads for } [LOC \text{ Baghdad}] \text{ .}$$

2.2.3 Conditional Random Fields

Conditional Random Fields (CRF)[2] sind ein diskriminatives, probabilistisches Modell, welches zur Segmentierung und Annotierung (engl. „labeling“) von sequentiellen Daten genutzt werden kann [73, S. 9]. CRFs können als sequentielle Erweiterung der logistischen Regression oder auch als diskriminative Version des Hidden Markov Modells aufgefasst werden [73, S. 7-8]. Typische Anwendungsbeispiele sind das Part of Speech Tagging oder auch die bereits genannte Named Entity Recognition.

[2] An dieser Stelle wird nur auf *Linear Chain* CRFs eingegangen und nicht auf deren Verallgemeinerung, *General* CRFs, da sich der Rest der Studie auf erstere Modelle bezieht.

Sei die Menge möglicher Sequenzen von Beobachtungen $\mathcal{X}$ und die Menge möglicher Label-Sequenzen $\mathcal{Y}$. Jedes einzelne Label wird aus einem endlichen Label-Alphabet entnommen, beispielsweise „PER“, „LOC“, „ORG“, „MISC“ und „O“ (für „Other“ bzw. „kein Named Entity“). Jede Sequenz von Beobachtungen $\vec{x}$ und Labeln $\vec{y}$ hat jeweils die Länge $(n+1)$. Eine CRF kann nun in Faktorschreibweise (vgl. Abbildung 2.1) definiert werden als

$$p(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y}), \tag{2.4}$$

wobei Z ein Normalisierungsterm ist für den

$$Z(\vec{x}) = \sum_{\vec{y}'} \prod_{j=1}^{n} \Psi_j(\vec{x}, \vec{y}'). \tag{2.5}$$

gilt [38, S. 15-16]. Seien nun die Faktoren definiert als

$$\Psi_j(\vec{x}, \vec{y}') = \exp(\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \vec{x}, j)), \tag{2.6}$$

dann ergibt sich die Wahrscheinlichkeit einer Sequenz $\vec{y}$ unter den Beobachtungen $\vec{x}$ aus

$$p_{\vec{\lambda}}(\vec{y}|\vec{x}) = \frac{1}{Z_{\vec{\lambda}}(\vec{x})} \cdot \exp(\sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \vec{x}, j)), \tag{2.7}$$

wobei analog der Normalisierungsterm über

$$Z_{\lambda'}(\vec{x}) = \sum_{\vec{y} \in \mathcal{Y}} \exp(\sum_{j=1}^{n} \sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \vec{x}, j)). \tag{2.8}$$

berechnet werden kann [38, S. 16].

Dabei ist $(n+1)$ die Anzahl der Wörter in der beobachteten Sequenz, m die Anzahl an Merkmals- bzw. Feature-Funktionen und f_i die i-te Feature-Funktion, welche das Gewicht λ_i hat [38, S. 16-17]. Jedes f_i darf den Wert 0 oder 1 annehmen [73, S. 10]. Eine Feature-Funktion könnte etwa den Wert 1 haben, falls das j-te beobachtete Wort mit einem Großbuchstaben beginnt und sonst 0. Liegt ein multinomiales Feature vor (bspw. POS-Tags), so muss dieses in mehrere binomiale Features zerlegt werden (bspw. POS-Tag ist NP, POS-Tag ist VP, ...).

2.2.4 Wortvektoren

Grundlage von Wortvektoren ist die verteilte Repräsentation („distributed representation“) von Wörtern in einem Vektorraum. Während bei einer lokalen Repräsentation („local

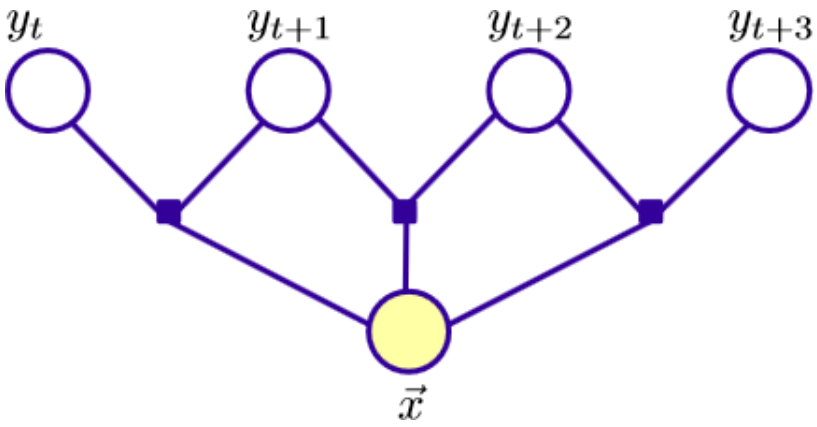


Abbildung 2.1: Ein Linear Chain Conditional Random Field in Faktordarstellung [38, S. 16].

representation“) jede Entität durch einen einzelnen klaren Eintrag repräsentiert wird, ergibt sich eine verteilte Repräsentation durch ein Muster von Aktivitäten bzw. durch die Kombination mehrerer einzelner Werte [30, S. 77-78]. Diese Werte können beispielsweise als snytaktische oder semantische Merkmale angesehen werden, wodurch für den Begriff „König“ etwa folgender Vektor x_{Koenig} denkbar wäre (jeweils Gewicht mit Merkmal in pseudo-mathematischer Schreibweise):

$$(0.99 \cdot \text{männlich}, -0.99 \cdot \text{weiblich}, 0.92 \cdot \text{Mensch}, -0.8 \cdot \text{Tier}, 0.91 \cdot \text{mächtig})^T$$

Auf die gleiche Weise könnten jeweils für „männlich“, „weiblich“ und „Königin“ passende Vektoren (mit anderen Gewichtungen) generiert werden, wodurch sich die Rechnung $x_{Koenig} - x_{maennlich} + x_{weiblich} \approx x_{Koenigin}$ ergäbe [52, S. 746]. Zur automatisierten Berechnung geeigneter Wortvektoren wird unter anderem das Skip-Gram-Modell verwendet[3]. Dieses stellt ein rekurrentes Neuronales Netz („recurrent neural net“) dar, welches ein Eingabewort $w(t)$ eines Satzes (als Vektor in One-Hot-Repräsentation) auf eine Wahrscheinlichkeitsverteilung über ein Wörterbuch $y(t)$ abbildet [52, S. 747]. Die Wahrscheinlichkeitsverteilung drückt aus, welche Wörter sich am ehesten in der Umgebung von $w(t)$ befinden [51, S. 2]. Die genaue Berechnung [51, S. 747] erfolgt anhand von

$$s(t) = f(\mathbf{U}w(t) + \mathbf{W}s(t-1)) \tag{2.9}$$

und

$$y(t) = g(\mathbf{V}s(t)) \tag{2.10}$$

mit

$$f(z) = \frac{1}{1+e^{-z}} \quad \text{und} \quad g(z) = \frac{e^{z_m}}{\sum_k e^{z_k}}. \tag{2.11}$$

Die Wahrscheinlichkeitsverteilung $y(t)$ ergibt sich also aus der internen Repräsentation $s(t)$ („hidden layer“), für deren Berechnung wiederum der vorherige Zustand des Netzes $s(t-1)$ und das aktuelle Wort $w(t)$ verwendet werden. Das Wort $w(t)$ wird jedoch nicht direkt im Hidden Layer als One-Hot-Repräsentation weiterverarbeitet, sondern stattdessen

[3]Eine Alternative stellt der CBOW-Ansatz (continuous bag of words) dar. Dieser wird nachfolgend nicht verwendet und daher an dieser Stelle übersprungen.

zuvor mit **U** verrechnet, um eine interne Repräsentation (bzw. eine verteilte Repräsentation) zu erzeugen. Die Matrix **U** kann als „Wissen“ des Netzes über die existierenden Merkmale und deren Gewichtung für das jeweilige Wort (ohne Kontext) aufgefasst werden. Entsprechend ergibt sich der Wortvektor eines Wortes $w(t)$ aus $\mathbf{U}w(t)$ (bzw. aus einer einzelnen Spalte in **U**, welche zu $w(t)$ gehört). Der Wortvektor kann abgespeichert werden, um später Vergleiche auf Ähnlichkeit zwischen Wörtern durchzuführen. Zudem kann jeder Wortvektor als Punkt in einem Raum aufgefasst werden, wodurch ein Clustering von ähnlichen Wörtern ermöglicht wird (siehe Tabelle 2.1). Zu beachten ist dabei, dass sich **U** nur auf das aktuelle Wort und nicht auf die vorherigen Wörter bezieht. Es ist daher denkbar, dass der Wortvektor bei Wörtern mit mehreren Bedeutungen (z. B. „Bank“, „Ring“) nur einen Sinn, wie etwa den häufigsten, erfasst. Möglich ist auch, dass er irgendwo „zwischen“ den Bedeutungen liegt, zumindest im Hinblick auf diejenigen Merkmale, welche sich unterscheiden.

2.2.5 Brown-Cluster

Der Brown-Clustering-Algorithmus dient zur Identifizierung von semantisch ähnlichen Wörtern in einem Vokabular [43, S. 43-44]. Es wird bei diesem davon ausgegangen, dass sich der Sinn eines Wortes erfassen lässt, indem die Umgebungen betrachtet werden, in denen dieses vorkommt [15, S. 470-471]. Es werden solche Wörter als ähnlich angesehen, welche häufig gemeinsam auftreten, also eine hohe Bigramm-Wahrscheinlichkeit haben („class-based bigram language model“) [15, S. 470-471]. Gedanklich wird jedes Wort zunächst einem eigenen Cluster zugeordnet [43, S. 44]. Anschließend wird agglomerativ geclustert [43, S. 47]. Pro Schritt wird jeweils das Maß

$$\text{Quality}(C) = I(C) - H \tag{2.12}$$

maximiert, wobei C der Vereinigung zweier Cluster entspricht, $I(C)$ deren Mutual Information ist und H die Entropie [43, S. 46]. Das sich so ergebende Clustering kann als Baum dargestellt werden, in dem die Blätter die Wörter repräsentieren. Dadurch kann wiederum die Einordnung jedes Wortes als Folge von Bits (beginnend von der Wurzel an) beschrieben werden, wobei eine Eins beispielsweise dem linken Kindknoten und eine Null dem rechten Kindknoten entspräche (siehe Tabelle 2.2). Die Laufzeit ist $O(kw^2+T)$, wobei k die Größe des Vokabulars ist, w die Anzahl der Cluster und T der Gesamtlänge des Textes entspricht [43, S. 49]. Demnach läuft der Algorithmus insbesondere bei vielen Clustern eher langsam und ist – im Gegensatz zum Wortvektor-Clustering – nur bedingt auf sehr große Korpora, welche aus mehreren Milliarden Wörtern bestehen, anwendbar.

2.2.6 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) ist ein generatives, probabilistisches Modell, welches Verteilungen über Dokumente, Themen und Wörter abbildet [10, S. 996] und beispielsweise bei der Identifizierung von Themen in Korpora Anwendung findet [10, S. 1012-1013]. Jedem Dokument w_n ordnet es eine multinomiale Dirichletverteilung über die Themen z zu, wodurch entsprechend jedes Dokument zu mehreren Themen gehört. Genauso wird

Wort	Cluster
Tribut-Album	163
Tubular	163
TV-Werbespot	163
Unheilig	163
USA-Tour	163
US-Tour	163
US-Tournee	163
Vereinshymne	163
Pugatschow	184
Puschkin	184
Puschkins	184
Putin	184
Putjatin	184
Rabinowitsch	184
Rahr	184

Tabelle 2.1: Ausschnitte aus zwei Clustern von Wörtern, welche aus den zugehörigen Wortvektoren abgeleitet wurden.

Bitkette	Wort
001110011111110	wohnen
001110011111110	wohnten
001110011111110	lebten
001110011111110	leben
0011100111111110	lagen
0011100111111110	standen
0011100111111110	liegen
0011100111111110	stehen
0011100111111111	erscheinen
0011100111111111	entstehen
0011100111111111	bestehen
0011100111111111	kommen

Tabelle 2.2: Ausschnitte aus drei Brown-Clustern von Wörtern. Die Bitkette stellt die Reihenfolge von Abzweigungen dar, welche im Baum der Wortgruppen gewählt werden muss, um zu diesem Wort zu gelangen.

wiederum jedem Thema z_i eine multinomiale Verteilung über eine Menge von Wörtern zugeordnet, sodass sich auch diese durch mehrere Wörter (mit unterschiedlichen Gewichten) zusammensetzen [10, S. 996]. Die Verbundwahrscheinlichkeit zwischen einer Themenverteilung θ, einer Menge von Themen z und den Dokumenten w (bestehend aus N Wörtern) ergibt sich aus

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \theta), \tag{2.13}$$

wobei α einen Prior über über die Verteilung der Themen darstellt und β wiederum ein Prior über die Verteilung von Wörtern innerhalb eines Themas ist [10, S. 996-997]. Abbildung 2.2 stellt eine LDA in Form eines probabilistischen graphischen Modells dar.

2.3 Vorverarbeitung

Die Handhabung der großen Menge an Wikipedia-Artikeln in ihrer Rohform ist eine nicht-triviale Aufgabe, deren Ergebnisse die Leistungsfähigkeit des Klassifikators signifikant beeinflussen. Dieses Kapitel beschäftigt sich daher zunächst damit, welche Artikel verworfen werden sollten, wie die Linkstruktur aus dem Wikipedia-Markup („Wikitext“) ausgelesen wird und wie dieser anschließend in natürlichsprachigen Text (ohne Codierung) umgewandelt wird. Danach wird darauf eingegangen, welche Zeichenketten in den so erhaltenen Texten sinnvollerweise als Personeneigennamen gekennzeichnet werden sollten und welche nicht.

2.3.1 Extraktion von Artikeln und internen Links

Alle Artikel der Wikipedia stehen – nach Sprachen getrennt – als XML-Dumps zum Download zur Verfügung[4]. Nachfolgend wird ein solcher Dump der deutschen Wikipedia verwendet. Aus diesem werden mit einem XML-Parser alle Artikel (in Wikitext-Codierung) extrahiert. Um möglichst nur natürlichsprachige Texte zu erhalten, welche für die Klassifikation nützliche Informationen enthalten, werden folgende Arten von Artikeln ausgeschlossen:

- Alle Diskussionsseiten, Portalseiten und ähnliche. (Erkennbar am Namespace-Attribut des Textes in der XML-Datei.)
- Alle Weiterleitungen. (Erkennbar an der Zeichenkette `#WEITERLEITUNG` oder `#REDIRECT` am Anfang des Textes.)
- Alle Begriffserklärungsseiten, da diese nur wenig natürlichsprachigen Text enthalten. (Erkennbar an der Zeichenkette `{{Begriffserklärung}}`.)
- Alle Listen, beispielsweise `Liste von Autoren/A`, da diese Artikel keinen natürlichsprachigen Text enthalten. (Erkennbar an der Zeichenkette `Liste` am Anfang des Artikelnamens.)

[4] `https://dumps.wikimedia.org/backup-index.html`

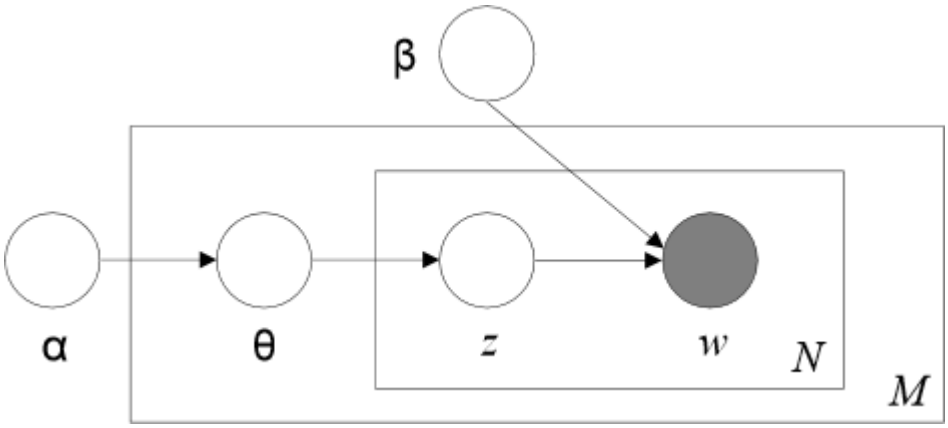


Abbildung 2.2: Darstellung einer LDA als probabilistisches graphisches Modell. α und β sind Prior. θ ist die Themenverteilung, deren Ausprägung die Erzeugung von Themen z beeinflusst. Abhängig von den Themen werden wiederum unterschiedliche Wörter w generiert. N ist die Anzahl der Wörter, M die Anzahl der Dokumente. [10, S. 996-997]

Anschließend werden alle internen Links (einschließlich der Kategorien) aus den so gewonnenen Artikeln ausgelesen und abgespeichert. Interne Links haben den Aufbau

```
[[Artikelname]] oder [[Artikelname|Linktext]]
```

und Kategorien wiederum

```
[[Kategorie:Name]] oder [[Kategorie:Name|Linktext]].
```

Zu beachten ist, dass die gleiche Syntax für andere Arten von Verweisen eingesetzt wird, wie beispielsweise im Falle von Bildern:

```
[[Datei:beispiel.jpg|Ein Beispielbild]].
```

Diese irrelevanten Verweise können ausgeschlossen werden, indem geprüft wird, ob die Zeichenkette vor dem ersten Doppelpunkt mit `Kategorie` identisch ist oder nicht.

Anschließend werden aus allen Artikeln für die Wikipedia typische Abschnitte entfernt, welche nur wenig natürlichsprachigen Text enthalten. Dazu gehören etwa „Weblinks“, „Einzelnachweise“ und „Siehe auch“.

Danach wird sämtlicher verbliebener Wikitext-Code entfernt. Dies umfasst etwa die Normalisierung von internen Links, Titeln (`== Titel ==`) und diversen stylistischen Auszeichnungen (z. B. `'''fettgedruckt'''`) zu reinem Text, sowie das Entfernen von Funktionsblöcken (zu denen Tabellen und Infoboxen gehören)

```
{{Funktion ...}}
```

und XML-/HTML-Code

```
<ref>Eine Referenz</ref>.
```

Ein beispielhaftes Skript, welches die meisten der vorherigen Schritte durchführt, wird von Matt Mahoney angeboten[5]. Zuletzt werden alle gewonnenen Texte unter Verwendung

[5] `http://mattmahoney.net/dc/textdata.html`

des Stanford Tokenizers[6] in Tokens unterteilt (engl. „Tokenization“).

2.3.2 Annotierung von Personeneigennamen

Nach dem zuvor beschriebenen Schritt der Vorverarbeitung stehen alle Artikel in ihrer bereinigten Fassung zur Verfügung (also ohne Wikitext-Code). Zudem liegt eine Liste aller internen Links vor, welche jeweils den Ausgangsartikel des Links, den Zielartikel und den Linktext erfasst. Zu den Links zählen alle Kategorie-Angaben am Ende der Artikel hinzu.

Bevor nun das weitere Vorgehen beschrieben wird, sollten folgende Einflussfaktoren bedacht werden:

- In der Wikipedia wird diszipliniert verlinkt. Ein Link, welcher auf einen Artikel über eine Person zeigt (nachfolgend vereinfachend „Personenartikel“), enthält auch fast immer eine Schreibweise des Namens der Person als Linktext (siehe Tabelle 2.3).

- Meistens wird nur die erste Erwähnung eines Namens in einem Artikel verlinkt. Daher kann ein Name mehrmals vorkommen, aber nur einmal verlinkt sein. Die nachfolgenden Erwähnungen können zudem eine andere Schreibweise haben (beispielsweise nur Nachname statt Vorname *und* Nachname). Es reicht daher nicht aus, nur die Links zu annotieren. Auch nicht verlinkte Textabschnitte müssen betrachtet werden („Oversampling“).

- Manche Namen werden gar nicht verlinkt.

- Die Namen in der Wikipedia sind sehr unterschiedlich und können daher ungewöhnliche Zeichen enthalten oder auch Wörter, welche nicht pauschal in Texten als Namen annotiert werden sollten. Beispiele dafür sind „Friedrich *der* Große“, „Wernherr *von* Braun“, „Johann Sebastian *Bach*“ und „Maria *von Ägypten*“ (enthalten alle Wörter oder Zeichenketten, welche meistens nicht Teil von Namen sind) oder auch „George W. Bush“ und „Ludwig XIV.“ (enthalten nach Tokenization freistehende Satzzeichen).

Aus den genannten Gründen wird für jeden Artikel eine eigene Menge an Wörtern generiert, welche als Namen annotiert werden sollen. Zunächst werden dafür alle Personenartikel identifiziert. Dies geschieht, indem jeder Artikel darauf überprüft wird, ob er zur Kategorie „*Mann*“ oder „*Frau*“ gehört, also auf diese Kategorie per (Kategorien-)Link verweist. Alle Personenartikel in der deutschen Wikipedia (und nur diese Artikel) sind einer der beiden Kategorien zugeordnet. Nachfolgend wird über alle Artikel $a_i \in A$ iteriert und für jeden eine Menge M_i von Wörtern und Wortkombinationen gebildet, welche als „Person“ (PER) annotiert werden sollen. Zu dieser Menge werden folgende Wörter und Wortkombinationen hinzugefügt:

[6] `http://nlp.stanford.edu/software/tokenizer.shtml`

- Alle Linktexte von Links, welche von a_i aus auf einen Personenartikel zeigen. Wenn also ein Text die Zeichenkette

  ```
  Der Komponist Ludwig van Beethoven schrieb
  am 17. Februar 1820, ...
  ```

  enthält und `Ludwig van Beethoven` auf den Artikel „Ludwig van Beethoven“ zeigt, dann würden diese Wörter zur Menge M_i hinzugefügt werden, da bekannt ist, dass der genannte Zielartikel ein Personenartikel ist.

- Falls a_i selbst ein Personenartikel ist, wird der Titel von a_i zu M_i hinzugefügt.

- Alle bisher zu M_i hinzugefügten Wortkombinationen werden zu einzelnen Wörtern getrennt und ebenfalls zu M_i hinzugefügt, sofern das jeweilige Wort mit einem Großbuchstaben beginnt und keine Ziffern oder Satzzeichen enthält. Der erste Teil der Bedingung greift in erster Linie bei Adelstiteln, der zweite bei Einschränkungen und Jahreszahlen aus Titeln von Artikeln (`Karl Jahn (Altphilologe)`, `Robert Clark (1777-1837)`), sowie bei Satzzeichen in Namen nach Tokenization (`Ludwig XIV .`).

- Die selben Wörter wie unter Punkt 3 werden noch einmal zu M_i hinzugefügt, diesmal aber jeweils ergänzt durch ein „s“ am Ende, um die meisten Genitive abzufangen.

Eine sich so für einen Artikel ergebende Menge könnte demnach

$$M_i = \{\text{Ludwig van Beethoven}, \text{Ludwig}, \text{Beethoven}, \text{Ludwigs}, \text{Beethovens}\}$$

sein. Durch einfaches Iterieren über die Wörter im Artikel und Vergleich mit der Menge, wird dann die Annotation durchgeführt. Einige Namen werden bei diesem Vorgehen nicht erkannt (da sie nirgendswo im Artikel verlinkt werden), einige werden nur teilweise erkannt (bspw. da nur eine andere Person mit gleichem *Nach*namen verlinkt wurde) und einige Wörter werden fälschlicherweise als Namen erkannt (vgl. `Maria von Ägypten`). Die Daten enthalten daher Rauschen, welches durch die vorgestellten Maßnahmen nur verringert, aber nicht ausgeschlossen werden kann. Dieses Rauschen hat Auswirkungen auf die Genauiggkeit des Klassifikators.

2.4 Merkmalsextraktion

Der Klassifikator arbeitet nicht direkt auf den Wörtern, sondern greift stattdessen auf Merkmale zurück. Diese werden in Kapitel 2.4.2 aufgelistet. Zuvor werden in Kapitel 2.4.1 einige vorbereitende Maßnahmen erläutert, die zur Generierung der Merkmale notwendig sind. Im Rahmen dieser Studie werden nur lokale Merkmale eingesetzt, also solche, welche aus einem Wort und gegebenenfalls seiner direkten Umgebung abgeleitet werden können [26, S. 10-11].

Linktext	Häufigkeit
Carl Friedrich Gauß	541
Gauß	91
gaußschen	4
Carl-Friedrich-Gauß	3
Gauss	3
C.F. Gauß	3
Carl Friedrich Gauss	2
C. F. Gauß	2

Tabelle 2.3: Häufigkeiten einzelner Beschreibungen beim Verlinken des Artikels „Carl Friedrich Gauß“ in der deutschen Wikipedia. (Weitere 13 Linktexte, welche jeweils nur einmal vorkommen, werden zur besseren Übersicht ausgelassen.)

2.4.1 Vorbereitende Maßnahmen

Für einige Features werden vorbereitende Maßnahmen durchgeführt, um diese im Klassifikator verwenden zu können. Im einzelnen sind diese:

- *Wortvektor-Cluster*: Anhand des Google-Tools word2vec[7] werden aus den Wikipedia-Artikeln 300-dimensionale Wortvektoren abgeleitet und (ebenfalls mit word2vec) in 1000 Cluster unterteilt, wodurch während des Trainings und der Validierung jedes Wort durch einen kurzen Wörterbuchzugriff einem Cluster (bzw. einer Gruppe ähnlicher Wörter) zugeordnet werden kann. Verwendet wird das Skip-Gram-Modell. Wörter, die seltener als 50 mal im Korpus vorkommen, werden ignoriert. Die RapidMiner *Text Processing Extension* bietet eine Reihe von Möglichkeiten, Dokumente einzulesen, Tokens zu extrahieren, Stoppwörter verschiedener Sprachen herauszufiltern, Stammformen einzusetzen, Synonyme zu ergänzen oder zu ersetzen, n-Gramme zu erzeugen und eine Bag of Words Repräsentation als Beispielsatz für alle Lernverfahren von RapidMiner bereitzustellen.

- *Brown-Cluster*: Anhand des Tools von Percy Liang[8] werden alle Wörter aus den Wikipedia-Artikeln in 1000 Brown-Cluster unterteilt. Wörter mit einer Häufigkeit unterhalb von 12 Vorkommnissen werden dabei ignoriert. Bei diesem Vorgang werden auch alle Brown-Cluster-Bitketten berechnet.

- *Gazetteer*: Zum Gazetteer werden alle Wörter hinzugefügt, welche wahrscheinlich Personeneigennamen sind. Wie bereits im Kapitel 2.3.2 (Annotierung von Personeneigennamen) beschrieben, sollten zu dieser Liste nicht pauschal *alle* Wörter hinzugefügt werden, welche in Personeneigennamen vorkommen. Daher werden nur solche Wörter zum Gazetteer hinzugefügt, welche in Namen vorkommen *und zugleich* in der Menge aller Namen eine höhere Auftrittswahrscheinlichkeit (bzw. Frequenz) haben als im sonstigen Korpus.

[7] `https://code.google.com/p/word2vec/`
[8] `https://github.com/percyliang/brown-cluster`

- *Unigramm-Liste*: Für alle Wörter wird die Häufigkeit im gesamten Korpus gezählt. Anschließend werden die Wörter nach Häufigkeit absteigend sortiert, wodurch sich etwa die 100 häufigsten Wörter einladen lassen.

- *LDA*: Normalerweise wird LDA zur Gruppierung thematisch ähnlicher Dokumente verwendet. In dieser Studie soll es jedoch zum Clustering thematisch ähnlicher Wörter genutzt werden. Dazu werden zunächst 50.000 Wikipedia-Artikel eingeladen, in Kleinschreibung umgewandelt und von seltenen Wörtern (weniger als 4 Vorkommnisse) bereinigt. Anschließend werden die Artikel in Blöcke von maximal 30 Wörtern unterteilt. Auf diese wird die LDA angewendet, um so eine Unterteilung in 100 Themen (Topics) zu erhalten.

- *POS-Tags*: Anwendung beim Ermitteln der Part of Speech Tags findet der Stanford-Parser[9]. Er stellt zum POS-Tagging des Deutschen mehrere Modelle zur Verfügung („german-dewac“, „german-fast“ und „german-hgc“), von denen „german-fast“ gewählt wird. Dessen Geschwindigkeit ist gegenüber den anderen beiden Modellen am höchsten, bei zugleich nur geringfügig schlechterer Genauigkeit.

2.4.2 Definition der Merkmale

Es werden insgesamt 16 verschiedene Features für den Klassifikator verwendet. Jedes Feature wird wiederum für jedes einzelne Wort in den Artikeln generiert. Tabelle 2.4 zeigt die Werte der Merkmale anhand von zwei verschiedenen Beispielen. Im Detail sind die Merkmale wie folgt definiert:

- Groß-/Kleinschreibung („starts with uppercase“, kurz „**swu**“): Gibt an, ob das Wort mit einem Großbuchstaben beginnt ($swu = 1$) oder nicht ($swu = 0$).

- Länge (kurz „**l**“): Gibt die Zeichenlänge des Wortes an. Der Maximalwert wird auf 30 begrenzt, da jeder Wert ($l = 1, l = 2, ...$) aus Sicht des Klassifikators wiederum ein einzelnes Feature darstellt, welches für ein Wort vorhanden ist oder nicht. Die Begrenzung auf 30 erzeugt somit aus Sicht das Klassifikators maximal 30 unterschiedliche Merkmale.

- Enthält Ziffern („contains digits“, kurz „**cd**“): Gibt an, ob das Wort mindestens eine Ziffer zwischen 0 und 9 enthält ($cd = 1$) oder nicht ($cd = 0$).

- Enthält Zeichensetzung („contains punctuation“, kurz „**cp**“): Gibt an, ob das Wort mindestens ein Satzzeichen enthält ($cp = 1$) oder nicht ($cp = 0$). Als Satzzeichen werden hier Punkte, Kommata, Semikolons, Ausrufezeichen, Fragezeichen, Doppelpunkte und Klammern gewertet.

- Enthält nur Ziffern („only digits“, kurz „**od**“): Gibt an, ob das Wort nur Ziffern zwischen 0 und 9 enthält ($od = 1$) oder nicht ($od = 0$).

- Enthält nur Satzzeichen („only punctuation“, kurz „**op**“): Gibt an, ob das Wort nur Satzzeichen enthält ($op = 1$) oder nicht ($op = 0$).

[9] `http://nlp.stanford.edu/software/lex-parser.shtml`

- Wortvektor-Cluster (kurz „**w2v**“): Gibt den Wortvektor-Cluster des Wortes an (z. B. $w2v = 578$ für ein Wort, welches zum Cluster 578 gehört). Dieses Merkmal hat daher maximal 1001 Ausprägungen (1000 Cluster sowie ein Wert für Wörter, die keinem Cluster zugeordnet werden können).

- Brown-Cluster (kurz „**bc**“): Gibt den Brown-Cluster des Wortes an (z. B. $bc = 112$). Analog zu den Wortvektor-Clustern ($w2v$) hat auch dieses Merkmal 1001 mögliche Ausprägungen.

- Brown-Cluster-Bits (kurz „**bcb**“): Gibt die ersten sieben Bits der Brown-Cluster-Bitkette an (z. B. `0000111` von `00001110100` für das Wort „Verlag“). Dieses Merkmal hat insgesamt ca. $(7^3+1) = 343$ verschiedene Ausprägungen, da jede Stelle den Wert 0, 1 oder „nichts“ einnehmen kann (nicht jede Bit-Kette muss sieben Zeichen oder mehr haben). Zudem wird der leere String für Wörter reserviert, welchen kein Brown-Cluster zugeordnet werden kann.

- Gazetteer (kurz „**g**“): Gibt an, ob das Wort im Gazetteer enthalten ist ($g = 1$) oder nicht ($g = 0$).

- Wortmuster (word pattern, kurz „**wp**“): Wandelt jedes Token in ein einheitliches Muster aus Zeichen um. Dabei werden Gruppen von Zeichen jeweils auf ein einzelnes Zeichen gemappt. Alle Buchstaben zwischen A und Z (einschließlich Ä, Ö, Ü) werden beispielsweise durch `A` ersetzt. Analoges gilt für alle Kleinbuchstaben (ersetzt durch `a`), alle Ziffern (ersetzt durch `9`), diverse Satzzeichen (ersetzt durch `.`) und Klammern (ersetzt durch `(`). Alle anderen Zeichen werden durch `#` ersetzt. Folgt nach dem Mapping mehrmals das gleiche Zeichen aufeinander, so wird dieses auf ein einzelnes Zeichen mit einem zusätzlichen Plus begrenzt (z. B. wird aus `aaaa` ein `a+`).

- Unigram-Position (N-Grams of length 1, kurz „**ng1**“): Die Position des Wortes in der zuvor beschriebenen Liste der – nach Häufigkeit sortierten – Unigramme, beispielsweise $ng1 = 3$ für das dritthäufigste Wort. Die Liste wird auf 100 Wörter begrenzt. Selteneren Wörtern wird der Wert $ng1 = -1$ zugewiesen, wodurch sich 101 Ausprägungen ergeben.

- Präfix (kurz „**pf**“): Enthält die ersten drei Zeichen des Wortes (z. B. $pf =$ Wik für `Wikipedia`). Zeichen, welche nicht zu den Bereichen a-z, A-Z oder Umlaute gehören und auch kein Punkt, Komma, Ausrufezeichen oder Fragezeichen sind, werden durch ein `#` ersetzt. Es ergeben sich maximal ca. $(26+26+4+4+1+1)^3 = 62^3 = 238328$ Ausprägungen (Kleinbuchstaben, Großbuchstaben, Umlaute, Satzzeichen, # und leerer String).

- Suffix (kurz „**sf**“): Enthält die letzten drei Zeichen des Wortes (analog zum Präfix).

- POS-Tag (kurz „**pos**“): Enthält das POS-Tag für das jeweilige Wort im Satz, bspw. $pos =$ NP. Der Stanford-Parser greift (für das Deutsche) auf den NEGRA-Korpus[10] zurück, wodurch sich 57 mögliche Ausprägungen ergeben.

[10] `http://www.coli.uni-saarland.de/projects/sfb378/negra-corpus/`

- LDA („lda“): Es wird ein Ausschnitt aus maximal 11 Wörtern aus dem Text extrahiert (das Wort selbst, sowie die fünf Wörter links und rechts davon). Über die LDA werden diesem Ausschnitt seine Themen bzw. Topics zugewiesen. Alle Themen, welche eine Wahrscheinlichkeit von weniger als 10% haben, werden verworfen. Die sich so ergebenden Themen werden als Merkmale hinzugefügt, jeweils mit der auf zwei Nachkommastellen gerundeten Wahrscheinlichkeit (bspw. lda17 = 0.25 und lda98 = 0.55 für ein Wort welches zu den Themen 17 und 98 gehört). Dieses Merkmal kann maximal $100 * 101 = 10100$ Ausprägungen annehmen (100 Themen, je 101 Werte zwischen 0,00 und 1,00).

2.5 Experimente

Auf Basis der extrahierten Wikipedia-Daten und der entworfenen Merkmale werden mehrere Experimente durchgeführt. In Kapitel 2.5.1 wird zunächst auf deren Ablauf eingegangen. Erläutert werden die Implementierung, die verwendeten Korpora und der Trainingsvorgang. Darauffolgend werden in Kapitel 2.5.2 die Ergebnisse eines Trainings auf mehreren zehntausend Beispielen beschrieben. Der genaue Einfluss einzelner Merkmale auf die Klassifikationsleistung wird in Kapitel 2.5.3 untersucht. Zuletzt geht Kapitel 2.5.4 auf den Unterschied zwischen gefiltertem und ungefiltertem Gazetteer ein.

2.5.1 Experimentablauf

Als Klassifikator wird ein Conditional Random Field verwendet. Sofern nicht anders erwähnt, wird nachfolgend für diesen eine Fenstergröße von 5 verwendet. Es werden also pro Wort zusätzlich diejenigen zehn Wörter in Merkmale umgewandelt, welche sich bis zu fünf Stellen links und rechts von diesem befinden. Die Umsetzung des Klassifikators erfolgt in Python über die Bibliothek „python-crfsuite“, welche wiederum auf die C++-Implementierung „CRFSuite“ zurückgreift.

Für die Experimente werden vier verschiedene Korpora verwendet:

1. Ein automatisiert annotierter Korpus, welcher durch die in der Vorverarbeitung beschriebenen Schritte generiert wurde. Dieser Korpus wird sowohl für das Training als auch für die Validierung der Klassifikatoren verwendet.

2. Ein kleinerer, per Hand annotierter Wikipedia-Korpus. Dieser besteht aus 5.460 Wörtern, von denen 305 Namen sind. Dieser Korpus wird ausschließlich zur Validierung verwendet.

3. Der Testdatensatz des GermEval 2014 Korpus, dessen Inhalt aus Wikipedia- und Nachrichtentexten besteht. Die darin enthaltene Sprache ist also nur in Teilen vergleichbaren mit derjenigen aus den ersten beiden Korpora. Er enthält 96.483 Wörter, von denen 2.719 Personeneigennamen sind.

Merkmal	Beispiel „John“	Beispiel „laufen“
Groß-/Klein (swu)	1	0
Länge (l)	4	6
Enthält Ziffern (cd)	0	0
Enthält Satzzeichen (cp)	0	0
Nur Ziffern (od)	0	0
Nur Satzzeichen (op)	0	0
Wortvektor-Cluster (w2v)	358 (Jerome, Jock, ...)	300 (lenken, locken, ...)
Brown-Cluster (bc)	97 (William, Charles, ...)	295 (spielen, führen, ...)
Brown-Cluster-Bits (bcb)	0001111 (von: 000111110)	0011100 (von: 0011100111110)
Gazetteer (g)	1	0
Wortmuster (wp)	Aa+	a+
Unigram-Position (ng1)	-1 (nicht Top 100)	-1 (nicht Top 100)
Präfix (pf)	Joh	lau
Suffix (sf)	ohn	fen
POS-Tag (pos)	NN (Beispiel)	VVINF (Beispiel)
LDA (lda)	Thema 47 = 0,2 (Beispiel)	Thema 80 = 0,17; Thema 12 = 0,6 (Beispiel)

Tabelle 2.4: Liste aller extrahierten Merkmale und ihre Werte für zwei beispielhafte Wörter. (POS-Tag und LDA-Wert hängen vom Kontext des Wortes ab.)

4. Ein vierter Korpus, welcher Reden aus dem Europäischen Parlament enthält[11] und sich aus 20.697 Tokens zusammensetzt, von denen wiederum 104 zu Namen gehören. Die Formulierungen unterscheiden sich deutlich von denen in Wikipedia-Texten. Dieser Korpus wird ebenfalls ausschließlich zur Validierung verwendet.

In der Trainingsphase werden die Artikel jeweils eingelesen und in Blöcke von 50 Wörtern unterteilt. Eine Unterteilung nach Sätzen wird nicht verwendet, da ein Punkt nicht immer ein Satzende signalisieren muss und insbesondere einige Personeneigennamen Punkte enthalten (beispielsweise „Ludwig XIV."). Jeder Block wird als Trainingsbeispiel verwendet, sofern mindestens eines der darin enthaltenen Wörter Teil eines Namens ist. Die Einschränkung wird gewählt, da andernfalls der Großteil aller Beispiele keine Namen enthält und damit nur wenig zusätzliche Informationen bieten würde.

Für die Validierungsphase werden ebenfalls Blöcke von jeweils 50 Wörtern verwendet – je nach Experimentablauf können diese aus einem der vier Korpora stammen. Da pro Wort das korrekte Label bekannt ist, können Precision, Recall und F1-Wert ermittelt werden.

2.5.2 Training auf einer großen Zahl an Beispielen

Es wird ein einzelner Klassifikator auf 40.000 Beispielen (je maximal 50 Wörter) mit allen zuvor beschriebenen Merkmalen trainiert. Die Anzahl von 40.000 ergibt sich in erster Linie durch RAM- und Zeitbeschränkungen. Nach dem Training wird die Leistung des Klassifikators in vier Szenarien geprüft:

- Test auf 5000 Beispielen, welche aus dem automatisiert annotierten Wikipedia-Korpus stammen.
- Test auf dem kompletten handannotierten Wikipedia-Korpus.
- Test auf dem kompletten Europaparlament-Korpus.
- Test auf dem kompletten GermEval 2014 Korpus.

Die Ergebnisse sind in Tabelle 2.5 dargestellt und liegen im Bereich zwischen 70 und 75 Punkten (F1) im Falle der Wikipedia-Daten. Auf dem GermEval 2014 Korpus werden 66 Punkte erreicht und auf dem Europaparlament-Korpus 50 Punkte. Der Klassifikator lernt demnach – trotz des Rauschens in den Beispieldaten – eine generalisierende Regel zur Erkennung von Personeneigennamen. Diese erzielt beim handannotierten Korpus sogar ein besseres Ergebnis, vermutlich gerade aufgrund des genannten Rauschens im automatisiert erstellten Korpus, welches die Testergebnisse negativ beeinträchtigt. Die niedrigeren Werte beim Test auf GermEval- und Europaparlament-Korpus lassen sich wahrscheinlich auf die darin verwendeten Sprachstile zurückführen, welche sich vom enzyklopädischen Deutsch der Wikipedia zumindest teilweise unterscheiden. Auffällig ist der große Unterschied zwischen Precision (93,9) und Recall (61,0) bei fast allen Korpora. Dieses Muster konnte auch bei den nachfolgenden Tests mit einzelnen Untermengen von Merkmalen beobachtet werden.

[11] `http://www.nlpado.de/~sebastian/software/ner_german.shtml`

Wiki (auto)	precision	recall	f1-score	support
O	99.4	99.5	99.4	243083
PER	74.1	71.6	72.8	5152
avg / total	98.9	98.9	98.9	248235

(a)

Wiki (hand)	precision	recall	f1-score	support
O	97.7	99.8	98.7	5155
PER	93.9	61.0	74.0	305
avg / total	97.5	97.6	97.4	5460

(b)

GermEval	precision	recall	f1-score	support
O	98.6	99.9	99.2	93764
PER	91.7	52.3	66.6	2719
avg / total	98.4	98.5	98.3	96483

(c)

Europaparl.	precision	recall	f1-score	support
O	99.7	100.0	99.8	20593
PER	80.9	36.5	50.3	104
avg / total	99.6	99.6	99.6	20697

(d)

Tabelle 2.5: Ergebnisse eines Klassifikators nach Training auf 40.000 Beispielen und Anwendung auf (a) den automatisiert annotierten Wikipedia-Korpus, (b) den handannotierten Wikipedia-Korpus, (c) den GermEval 2014 Korpus und (d) den Europaparlament-Korpus.

2.5.3 Untermengen von Features

Es werden nacheinander mehrere Klassifikatoren mit unterschiedlichen Teilmengen von Merkmalen trainiert. Diese Teilmengen umfassen zum einen nur einzelne Merkmale (z. B. nur Präfixe von Wörtern) und zum anderen Merkmalspaare (z. B. Präfixe und Brown-Cluster von Wörtern). Da 16 Merkmale zur Verfügung stehen und bei Paaren die Reihenfolge keine Rolle spielt, ergeben sich so $(16 + 120) = 136$ Teilmengen. Für jede Teilmenge wird eine 5-fache Kreuzvalidierung auf 4.000 Beispielen (je maximal 50 Wörter) durchgeführt. Davon werden (pro Durchlauf der Kreuzvalidierung) 3.200 Beispiele zum Training und 800 zur Validierung verwendet. Jeder so erzeugte Klassifikator wird zusätzlich auf dem handannotierten Wikipedia-Korpus getestet. Die Ergebnisse sind in Tabelle 2.6 (automatisiert erstellter Korpus) und Tabelle 2.7 (handannotierter Korpus) als Matrizen dargestellt. Die Zellen enthalten jeweils den durchschnittlichen F1-Wert aller fünf Klassifikatoren und (in Klammern) den Standardfehler. Am höchsten ist der F1-Wert für die Einzelmerkmale Brown-Cluster-Bits, Brown-Cluster, Wortvektor-Cluster und Gazetteer.

In Kombination erzielen Brown-Cluster-Bits (oder alternativ Brown-Cluster) zusammen dem Gazetteer die beste Bewertung (je 70,0 F1) auf dem automatisiert annotierten Korpus. Beim handannotierten Korpus hingegen ergeben Part-of-Speech-Tags und Gazetteer das beste Paar (77,8 F1). Die Bewertungen deuten darauf hin, dass der Gazetteer dem Klassifikator wichtige Informationen bietet und den erreichten F1-Wert um mehrere Punkte erhöhen kann. Darüber hinaus eignen sich möglicherweise Brown-Cluster besser zur Erkennung von Personeneigennamen als Wortvektor-Cluster. Dies kann allerdings auch im Zusammenhang mit den gewählten Parametern beim Erzeugen der Wortvektoren stehen. (Die Dimensionalität von 300 ist vergleichsweise niedrig. Zudem wurden Wörter, die seltener als 50 mal auftreten, ausgeschlossen, was viele seltene Namen betreffen könnte.) Ferner lässt sich erkennen, dass die einfach zu generierenden Merkmale (z. B. Wortlänge) zwar alleine keinen guten Klassifikator erzeugen können, in Kombination mit anderen Merkmalen aber dennoch das Ergebnis geringfügig verbessern.

Tatsächlich gibt es ein Plugin von RapidMiner, das automatisch nach unterschiedlichen Strategien und in effizienter Implementierung Merkmalsmengen auswählt und natürlich eine Kreuzvalidierung beinhaltet [42]. Wenn man dies nicht kennt, ist es zu aufwändig, mehr als Paare von Merkmalen zu untersuchen. Deshalb sind die folgenden Tabellen beschränkt.

2.5.4 Filterung des Gazetteers

Im Abschnitt zur Merkmalsextraktion (Kapitel 2.4.1) wurde bereits erläutert, dass der Gazetteer erzeugt wird, indem nur solche Wörter in das Verzeichnis übernommen werden, welche in Personeneigennamen häufiger vorkommen als im restlichen Korpus. An dieser Stelle soll genauer ausgewertet werden, welche Auswirkungen es hat, diese Filterung auszulassen und stattdessen *alle* Wörter hinzuzufügen, welche in Personeneigennamen vorkommen. Dazu werden zwei Merkmale, ein Gazetteer mit Filterung und eines ohne, gebildet. Analog zum vorherigen Kapitel werden beide über eine 5-fache Kreuvalidierung auf 4.000 Beispielen (je 3.200 Training, 800 Validierung) geprüft. Zusätzlich wird jeder so erzeugte Klassifikator auf den handannotierten Wikipedia-Korpus angewendet. Die Ergebnisse zeigt Tabelle 2.8. Wie deutlich zu erkennen ist, hat der ungefilterte Gazetteer fast keine Aussagekraft mehr. Seine F1-Werte liegen bei maximal knapp 1,7 Punkten, während der Gefilterte bis zu 41 Punkte erreicht.

2.6 Zusammenfassung und Ausblick

In den vorherigen Kapiteln wurde die Verwendung von Wikipedia-Artikeln im Rahmen der Named Entity Recognition (bezogen auf Personeneigennamen) untersucht. Dabei hat sich herausgestellt, dass auch eine automatisierte Annotation von Namen ausreichend ist, um einen Klassifikator mit akzeptabler Genauigkeit zu trainieren. Zudem hat sich die Auflistung der gefundenen Namen in Form eines Gazetteers als nützliches Merkmal erwiesen. Verbesserungspotenzial lässt sich im Bereich des eher niedrigen Recalls ausmachen. Ein einfacher Ansatz zur Steigerung des Wertes könnte das Training auf einer

	bcb	bc	w2v	g	sf	pf	lda	pos	wp	ng1	swu	od	l	cd	cp	op
bcb	**67.4** (2.3)	68.8 (2.2)	69.4 (1.9)	**70.0** (2.6)	66.8 (1.5)	67.2 (1.7)	67.4 (2.1)	67.8 (2.7)	68.8 (2.7)	67.6 (3.3)	68.0 (2.7)	67.8 (2.6)	68.0 (2.2)	67.6 (2.7)	68.2 (2.5)	67.6 (2.7)
bc	68.8 (2.2)	**66.6** (2.6)	68.8 (2.7)	**70.0** (2.1)	66.2 (2.3)	66.8 (1.5)	66.4 (2.3)	68.4 (2.4)	67.6 (2.3)	67.2 (2.5)	67.8 (2.2)	66.8 (2.4)	66.8 (2.5)	66.6 (2.6)	67.0 (2.4)	66.8 (2.5)
w2v	69.4 (1.9)	68.8 (2.7)	**55.8** (1.7)	67.0 (2.3)	57.6 (3.0)	56.2 (2.1)	56.6 (2.4)	63.2 (1.6)	60.2 (2.3)	57.0 (1.7)	59.6 (2.4)	55.6 (2.0)	57.6 (2.9)	56.2 (2.2)	55.8 (1.7)	55.4 (1.5)
g	**70.0** (2.6)	**70.0** (2.1)	67.0 (2.3)	**36.0** (3.0)	59.8 (2.5)	59.2 (1.3)	57.4 (3.8)	60.2 (2.0)	58.4 (1.9)	55.0 (2.4)	54.8 (1.7)	36.6 (3.3)	55.2 (2.3)	37.4 (3.0)	42.0 (2.3)	39.8 (2.2)
sf	66.8 (1.5)	66.2 (2.3)	57.6 (3.0)	59.8 (2.5)	**33.8** (2.3)	41.4 (3.9)	45.4 (3.4)	47.6 (2.2)	45.2 (2.6)	36.8 (1.7)	44.0 (2.3)	33.6 (2.1)	36.8 (1.5)	34.0 (2.1)	33.6 (2.1)	33.8 (1.9)
pf	67.2 (1.7)	66.8 (1.5)	56.2 (2.1)	59.2 (1.3)	41.4 (3.9)	**32.2** (2.8)	44.4 (3.3)	47.8 (1.6)	41.8 (2.1)	35.8 (2.6)	40.6 (1.9)	32.8 (2.3)	36.4 (3.0)	32.6 (2.5)	33.2 (2.7)	33.0 (2.4)
lda	67.4 (2.1)	66.4 (2.3)	56.6 (2.4)	57.4 (3.8)	45.4 (3.4)	44.4 (3.3)	**25.4** (1.9)	50.8 (2.6)	43.6 (3.0)	33.4 (1.0)	41.4 (2.9)	29.2 (1.5)	33.4 (1.0)	30.0 (1.8)	26.4 (1.7)	27.2 (1.7)
pos	67.8 (2.7)	68.4 (2.4)	63.2 (1.6)	60.2 (2.0)	47.6 (2.2)	47.8 (1.6)	50.8 (2.6)	**20.0** (3.6)	35.6 (1.2)	35.4 (1.4)	25.2 (2.0)	20.4 (3.1)	35.0 (2.0)	21.2 (3.5)	22.0 (2.8)	21.2 (3.2)
wp	68.8 (2.7)	67.6 (2.3)	60.2 (2.3)	58.4 (1.9)	45.2 (2.6)	41.8 (2.1)	43.6 (3.0)	35.6 (1.2)	**7.8** (2.1)	26.0 (1.9)	8.2 (1.9)	7.8 (1.7)	15.4 (2.4)	8.0 (2.1)	8.6 (2.8)	8.6 (2.2)
ng1	67.6 (3.3)	67.2 (2.5)	57.0 (1.7)	55.0 (2.4)	36.8 (1.7)	35.8 (2.6)	33.4 (1.0)	35.4 (1.4)	26.0 (1.9)	**3.8** (1.9)	19.0 (1.1)	4.2 (1.5)	9.4 (1.2)	4.2 (1.6)	3.8 (1.9)	3.8 (1.9)
swu	68.0 (2.7)	67.8 (2.2)	59.6 (2.4)	54.8 (1.7)	44.0 (2.3)	40.6 (1.9)	41.4 (2.9)	25.2 (2.0)	8.2 (1.9)	19.0 (1.1)	**1.6** (1.4)	2.2 (1.2)	9.0 (2.0)	2.0 (1.4)	1.6 (1.2)	1.4 (1.4)
od	67.8 (2.6)	66.8 (2.4)	55.6 (2.0)	36.6 (3.3)	33.6 (2.1)	32.8 (2.3)	29.2 (1.5)	20.4 (3.1)	7.8 (1.7)	4.2 (1.5)	2.2 (1.2)	**0.0** (0.0)	0.2 (0.4)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
l	68.0 (2.2)	66.8 (2.5)	57.6 (2.9)	55.2 (2.3)	36.8 (1.5)	36.4 (3.0)	33.4 (1.0)	35.0 (2.0)	15.4 (2.4)	9.4 (1.2)	9.0 (2.0)	0.2 (0.4)	**0.0** (0.0)	0.6 (0.5)	0.0 (0.0)	0.0 (0.0)
cd	67.6 (2.7)	66.6 (2.6)	56.2 (2.2)	37.4 (3.0)	34.0 (2.1)	32.6 (2.5)	30.0 (1.8)	21.2 (3.5)	8.0 (2.1)	4.2 (1.6)	2.0 (1.4)	0.0 (0.0)	0.6 (0.5)	**0.0** (0.0)	0.0 (0.0)	0.0 (0.0)
cp	68.2 (2.5)	67.0 (2.4)	55.8 (1.7)	42.0 (2.3)	33.6 (2.1)	33.2 (2.7)	26.4 (1.7)	22.0 (2.8)	8.6 (2.8)	3.8 (1.9)	1.6 (1.2)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	**0.0** (0.0)	0.0 (0.0)
op	67.6 (2.7)	66.8 (2.5)	55.4 (1.5)	39.8 (2.2)	33.8 (1.9)	33.0 (2.4)	27.2 (1.7)	21.2 (3.2)	8.6 (2.2)	3.8 (1.9)	1.4 (1.4)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	**0.0** (0.0)

Tabelle 2.6: Durchschnittliche F1-Werte (mit Standardfehler) von einzelnen Merkmalen (Mitteldiagonale) und Merkmalspaaren (sonstige Zellen) bei 5-facher Kreuzvalidierung auf dem automatisiert annotierten Wikipedia-Korpus (3200 Beispiele Training, 800 Validierung). Die Bedeutungen der Abkürzungen sind in Kapitel 2.4.2 aufgelistet.

größeren Stichprobe als den gewählten 40.000 Beispielen darstellen, eventuell ergänzt durch eine Erhöhung der Fenstergröße. Sinnvoll erscheint auch die Verwendung nichtlokaler Merkmale, wodurch jedoch die Komplexität des Klassifikators deutlich ansteigen würde. Darüber hinaus bietet die Vorverarbeitung noch Raum zur Verbesserung bzw. zur genaueren automatisierten Annotierung von Personeneigennamen.

	bcb	bc	w2v	g	pos	pf	lda	sf	wp	swu	ng1	od	l	cd	cp	op
bcb	**71.2** (2.1)	70.0 (1.1)	65.8 (2.6)	75.2 (1.0)	74.2 (0.4)	67.0 (1.1)	72.6 (1.9)	67.0 (2.8)	69.6 (1.4)	70.6 (1.6)	72.4 (1.6)	70.8 (1.5)	70.0 (1.9)	71.4 (1.9)	70.8 (1.6)	72.0 (2.4)
bc	70.0 (1.1)	**62.4** (2.7)	59.0 (4.7)	73.0 (1.3)	68.8 (1.2)	59.2 (0.7)	64.6 (0.8)	57.2 (1.7)	65.4 (2.0)	63.8 (1.3)	62.8 (2.0)	62.6 (2.2)	60.2 (2.0)	63.0 (2.0)	62.6 (2.6)	62.0 (2.3)
w2v	65.8 (2.6)	59.0 (4.7)	**49.2** (1.9)	65.2 (1.6)	61.6 (1.4)	45.0 (2.9)	49.6 (1.5)	45.0 (2.5)	48.6 (1.6)	52.2 (1.5)	51.2 (1.7)	50.2 (1.8)	46.2 (3.7)	50.4 (1.6)	45.8 (1.5)	49.6 (2.4)
g	75.2 (1.0)	73.0 (1.3)	65.2 (1.6)	**43.4** (0.5)	**77.8** (1.7)	65.8 (1.5)	64.8 (1.0)	58.4 (2.1)	65.4 (1.9)	62.8 (1.2)	68.6 (2.1)	45.2 (3.2)	64.4 (1.6)	44.0 (3.3)	41.2 (1.2)	41.4 (1.5)
pos	74.2 (0.4)	68.8 (1.2)	61.6 (1.4)	**77.8** (1.7)	**28.2** (3.1)	48.4 (2.4)	56.6 (2.7)	36.6 (1.9)	36.2 (3.3)	29.0 (1.1)	43.2 (1.0)	27.0 (3.0)	43.6 (2.3)	28.2 (1.9)	29.6 (2.1)	29.4 (2.7)
pf	67.0 (1.1)	59.2 (0.7)	45.0 (2.9)	65.8 (1.5)	48.4 (2.4)	**18.8** (2.0)	36.2 (2.5)	17.6 (2.4)	32.2 (1.8)	32.6 (1.5)	21.0 (2.0)	18.8 (1.7)	23.8 (2.5)	19.6 (1.9)	16.8 (1.9)	19.0 (1.7)
lda	72.6 (1.9)	64.6 (0.8)	49.6 (1.5)	64.8 (1.0)	56.6 (2.7)	36.2 (2.5)	**17.4** (0.5)	31.2 (1.5)	37.8 (2.2)	33.8 (1.9)	16.8 (1.7)	18.4 (2.1)	23.4 (1.9)	20.2 (1.9)	15.4 (1.6)	17.4 (1.5)
sf	67.0 (2.8)	57.2 (1.7)	45.0 (2.5)	58.4 (2.1)	36.6 (1.9)	17.6 (2.4)	31.2 (1.5)	**12.4** (1.7)	29.6 (1.2)	24.4 (2.3)	14.0 (1.8)	12.4 (1.2)	16.8 (1.3)	13.6 (2.5)	10.6 (1.6)	10.8 (1.8)
wp	69.6 (1.4)	65.4 (2.0)	48.6 (1.6)	65.4 (1.9)	36.2 (3.3)	32.2 (1.8)	37.8 (2.2)	29.6 (1.2)	**2.8** (1.0)	2.2 (1.0)	13.0 (0.9)	2.8 (1.0)	9.0 (3.5)	2.8 (1.0)	2.4 (1.2)	1.8 (1.0)
swu	70.6 (1.6)	63.8 (1.3)	52.2 (1.5)	62.8 (1.2)	29.0 (1.1)	32.6 (1.5)	33.8 (1.9)	24.4 (2.3)	2.2 (1.0)	**1.0** (0.0)	15.6 (1.5)	1.0 (0.0)	7.0 (1.3)	1.0 (0.0)	1.0 (0.0)	0.8 (0.4)
ng1	72.4 (1.6)	62.8 (2.0)	51.2 (1.7)	68.6 (2.1)	43.2 (1.0)	21.0 (2.0)	16.8 (1.7)	14.0 (1.8)	13.0 (0.9)	15.6 (1.5)	**0.6** (0.5)	0.6 (1.2)	4.2 (0.7)	1.0 (1.1)	0.0 (0.0)	0.6 (0.5)
od	70.8 (1.5)	62.6 (2.2)	50.2 (1.8)	45.2 (3.2)	27.0 (3.0)	18.8 (1.7)	18.4 (2.1)	12.4 (1.2)	2.8 (1.0)	1.0 (0.0)	0.6 (1.2)	**0.0** (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
l	70.0 (1.9)	60.2 (2.0)	46.2 (3.7)	64.4 (1.6)	43.6 (2.3)	23.8 (2.5)	23.4 (1.9)	16.8 (1.3)	9.0 (3.5)	7.0 (1.3)	4.2 (0.7)	0.0 (0.0)	**0.0** (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)
cd	71.4 (1.9)	63.0 (2.0)	50.4 (1.6)	44.0 (3.3)	28.2 (1.9)	19.6 (1.9)	20.2 (1.9)	13.6 (2.5)	2.8 (1.0)	1.0 (0.0)	1.0 (1.1)	0.0 (0.0)	0.0 (0.0)	**0.0** (0.0)	0.0 (0.0)	0.0 (0.0)
cp	70.8 (1.6)	62.6 (2.6)	45.8 (1.5)	41.2 (1.2)	29.6 (2.1)	16.8 (1.9)	15.4 (1.6)	10.6 (1.6)	2.4 (1.2)	1.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	**0.0** (0.0)	0.0 (0.0)
op	72.0 (2.4)	62.0 (2.3)	49.6 (2.4)	41.4 (1.5)	29.4 (2.7)	19.0 (1.7)	17.4 (1.5)	10.8 (1.8)	1.8 (1.0)	0.8 (0.4)	0.6 (0.5)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	0.0 (0.0)	**0.0** (0.0)

Tabelle 2.7: Durchschnittliche F1-Werte (mit Standardfehler) von einzelnen Merkmalen (Mitteldiagonale) und Merkmalspaaren (sonstige Zellen) nach Training von jeweils 5 Klassifikatoren auf wechselnden 3.200 Beispielen (5-fache Kreuzvalidierung) und Anwendung auf den handannotierten Korpus. Die Bedeutungen der Abkürzungen sind in Kapitel 2.4.2 aufgelistet.

gefiltert, automatisiert annotiert

Fold	precision	recall	f1-score	support
1	57.6	28.6	38.2	928
2	62.4	28.9	39.5	871
3	56.6	26.4	36.0	962
4	55.9	25.6	35.2	893
5	50.0	22.1	30.7	986

ungefiltert, automatisiert annotiert

Fold	precision	recall	f1-score	support
1	31.0	1.0	1.9	928
2	100.0	2.1	4.0	871
3	75.0	0.9	1.9	962
4	0.0	0.0	0.0	893
5	45.5	0.5	1.0	986

gefiltert, handannotiert

Fold	precision	recall	f1-score	support
1	86.2	26.6	40.6	305
2	86.2	26.6	40.6	305
3	86.7	27.9	42.2	305
4	86.2	26.6	40.6	305
5	86.5	27.2	41.4	305

ungefiltert, handannotiert

Fold	precision	recall	f1-score	support
1	0.0	0.0	0.0	305
2	0.0	0.0	0.0	305
3	0.0	0.0	0.0	305
4	0.0	0.0	0.0	305
5	0.0	0.0	0.0	305

Tabelle 2.8: Gegenüberstellung von gefiltertem (linke Tabellen) und ungefiltertem (rechte Tabellen) Gazetteer. Es wurden anhand von 5-facher Kreuzvalidierung Klassifikatoren auf jeweils 3.200 wechselnden Beispielen aus dem automatisiert annotierten Wikipedia-Korpus trainiert und auf 800 Beispielen getestet (obere Tabellen). Ebenso wurde auf dem handannotierten Wikipedia-Korpus getestet (untere Tabellen).

Kapitel 3

Internetsprache vs. Nachrichtensprache: Ist das Genre des verwendeten Korpus bestimmend für die Qualität eines POS Taggers?

von Jan Weckwerth

3.1 Einführung

Hier untersuchen wir die Qualität eines Part-Of-Speech (POS) Taggers unter dem Gesichtspunkt des Genres eines zum Anlernen verwendeten Korpus. Dazu wurden drei verschiedene Korpora verglichen. Zwei davon (Negra und *TUDZ*) basieren auf Nachrichtentexten, einer davon „*EmpiriST2015*“ auf Internetquellen. Die Korpora wurden dabei unter der Verwendung des *POS Tagger* untersucht.

Ein Verfahren des maschinellen Lernens, welches in direkter Abhängigkeit zum Verarbeiten natürlicher Sprache steht, ist das Annotieren von Textquellen mit so genannten POS-Tags. Diese Tags weisen jedem Wort eines Satzes die grammatikalische Bedeutung zu.

Auch	Philip	Glass	wurde	auf	seinen	weltweiten	Tourneen	mit	Kassetten	und	Tonbändern	überschüttet	.
ADV	NE	NE	VAFIN	APPR	PPOSAT	ADJA	NN	APPR	NN	KON	NN	VVPP	$.

Tabelle 3.1: Beispiel für POS annotierten Text.

Ein POS Tagger übernimmt die Aufgabe, einen eingegebenen Text mit solchen Tags zu versehen. Dazu muss dieser Tagger jedoch zunächst angelernt werden. Der Tagger benötigt dafür eine Beispielmenge mit vorher annotierten Texten. Eine solche Menge annotierter Texte wird annotierter Korpus genannt und steht für verschiedene Sprachen zur Verfügung. Das Lernen eines Taggers ist mit dem POS Tagger [78] über das Max-Entropy-Verfahren möglich, welcher im Rahmen dieser Arbeit verwendet wird. Ebenfalls im Umfang des POS Taggers ist ein auf dem Negra-Korpus erlernter Tagger. Der Negra-Korpus beinhaltet 355096 annotierte Tokens aus 20602 Sätzen, welche aus den Zeitungstexten

der *Frankfurter Rundschau* stammen, einer deutschsprachigen Zeitung.

Die Arbeit beschäftigt sich nun mit der Frage, ob das korrekte Annotieren von deutschsprachigen Texten von dem Genre des zum Anlernen verwendeten Korpus abhängt. Dazu werden insgesamt drei angelernte Tagger verglichen. Die ersten beiden Tagger werden dabei mit dem Negra-Korpus beziehungsweise mit dem TUDZ angelernt. Der dritte Tagger basiert auf 5000 von Hand annotierten Tokens, des vorläufigen EmpiriST2015-Trainingsdatensets. Dieser entsteht in der *Shared Task* des Empirikom-Netzwerks zur automatischen linguistischen Annotation deutschsprachiger internetbasierter Kommunikation im Jahr 2015 und ist eine Gemeinschaftsarbeit zwischen der Technischen Universität Dortmund, der Friedrich-Alexander-Universität Erlangen-Nürnberg und der Berlin Brandenburgischen Akademie der Wissenschaften.

3.2 Hintergrund

Die Hypothese dieser Arbeit beruht auf den Ergebnissen der Arbeit von Dima und Hinrichs [21]. Dort wurde das Phänomen untersucht, dass die Penn Treebank, ein englischer annotierter Korpus, beim Taggen von Fragen schlechtere Ergebnisse liefert, als wenn diese auf den eigenen Daten getestet wird. Eine Erweiterung der Penn Treebank um eine Reihe von Fragen, unter anderem auch aus „Internet“-Medien hat die Leistung erhöht.

Die Penn Treebank basiert auf Nachrichtentexten. Die Vermutung, welche im Rahmen der Arbeit bestätigt wurde ist, dass der syntaktische Aufbau von Fragen in den Nachrichtentexten der Penn-Treebank zu einheitlich ist. Das Hinzufügen von Fragen aus anderen Quellen verbessert demnach die korrekte Erkennung von spezifischen Strukturen in Fragen.

Der Transfer zu der Hypothese dieser Arbeit besteht nun darin, dass nicht die Frage als syntaktisches Novum betrachtet wird, sondern das Genre „Internet“, als ein Medium, welches nicht die syntaktische und auch semantische Klarheit eines Nachrichtentextes aufweist. Abweichend von der Arbeit aus [21] jedoch werden die grundlegenden Korpora nicht um Daten erweitert, sondern in Kontrast dazu gestellt.

3.3 Korpora

Die verwendeten Korpora sind der Negra-Korpus, der TUDZ und das EmpiriST2015. In diesem Abschnitt sollen diese etwas genauer vorgestellt werden.

Der Negra-Korpus [39] basiert auf 60.000 bereits annotierten Tokens, welche um 295.096 Token aus deutschen Zeitungstexten der *Frankfurter Rundschau*, einer deutschen Tageszeitung, erweitert wurden. Der Korpus wurde mit Part-of-Speech Tags und mit syntakti-

schen Strukturen annotiert. Das verwendete Tag-Set ist das STTS. Für die ersten 60.000 bereits annotierten Token wurde eine morphologische Analyse vorgenommen. Die daraus entnommenen Informationen sind über einen erweiterten STTS annotiert. Weiter Informationen sind die grammatikalische Funktion eines Tokens und die Annotation von Phrasen.

Der TUDZ [75] basiert auf 1.569.916 Tokens, welche auf der Grundlage der Zeitung „die tageszeitung“ (taz) aufgebaut wurden. Ein Großteil dieser Tokens stammen aus einem Zeitraum zwischen 1989 und 1999. Auch der TUDZ ist mit Part-of-Speech Tags und syntaktischen Strukturen annotiert. Bei den verwendeten Tags handelt es sich ebenfalls um das STTS.

Der letzte Korpus, welcher repräsentativ für das Genre „Internet“ steht, besteht aus 5000 von Hand annotierten Tokens. Die Grundlage dazu bilden Chat-Mitschnitte aus verschiedenen Quellen. Die Herausforderung für diesen Korpus ist die Annotation von unbekannten Wortkonstrukten, wie zum Beispiel Emoticons und das Vergeben von Tags für Phänomene der konzeptionellen Mündlichkeit [7]. Die dazu benötigten Tags sind als Erweiterungen des STTS abgebildet.

Für diese Arbeit werden spezifische Erweiterungen des STTS entfernt. Dazu gehören vor allem Annotationen an der Baumdarstellungen der hier vorgestellten Korpora. Diese drücken grammatikalische Besonderheiten aus, welche vom POS Tagger nicht genutzt werden. Zudem macht es den Vergleich der Korpora möglich. Die im POS Tagger verwendeten Normalisierungsprozeduren sind bereits darauf ausgelegt diese Gleichschaltung der Notation vorzunehmen.

3.4 Versuchsaufbau

Der Versuchsaufbau vergleicht drei unterschiedliche Korpora. Dazu wird das Mittel der 4-fachen Kreuzvalidierung genutzt. Die vorhandenen Korpora sind in jeweils 4 gleich große Teile, gemessen an der Zeilenanzahl der vorhandenen Teile aufgeteilt. Der Versuch ist in drei Schritte unterteilt. Im ersten Schritt wird die Performanz eines Korpus auf sich selbst untersucht.

Aus den vier Teilen des Korpus wird dabei ein Teil als Testmenge und drei Teile zum Lernen des POS Taggers verwendet. Für das Lernen und Testen bietet der POS Tagger das Verwenden von .props Dateien an. Diese enthalten alle zu machenden Einstellungen, welche vom POS Tagger erwartet werden. In dem Versuchsaufbau wird die von Stanford mitgelieferte Datei verwendet, welche für den Negra-Korpus optimiert ist und schnelles Parsen verspricht. Neben der Angabe der Trainingsmenge muss hier das Format angegeben werden. Unterstützt werden drei Dateiformate, wobei in diesem Versuch der Negra- und TUDZ im Penn-Treebank Format vorliegen. Der dritte Korpus, welcher stellvertre-

tend für das Internet-Genre steht, ist im TSV (Tab-Seperated Values) Format vorhanden.

Bei dem Negra- und TUDZ kommt eine transformierende Klasse zum Einsatz, welche dafür sorgt, dass die POS-Tags aus den Baumbanken extrahiert werden. Bei beiden Korpora sind Implementierungen von Stanford vorhanden. Der TUDZ benutzt für die Beschriftung der Kanten spezielle Tags. Auch diese können mit Hilfe der transformierenden Klasse entfernt werden, da sie dem Training über dem POS Tagger keine weiteren Informationen hinzufügen und die Vergleichbarkeit der Korpora verhindert. Für den dritten Korpus wurden die passenden Spalten für Token und Tag angegeben.

Über eine zweite .props Datei wird das Testen gesteuert. Die Änderungen zu der beim Training verwendeten Datei bestehen in der Angabe einer passenden Testmenge und dem erlernten Tagger.

Über ein Shell-Skript wird der POS Tagger aufgerufen. Die Bedienung erfolgt über die Klasse „edu.stanford.nlp.tagger.maxent.MaxentTagger“. Dieser wird die .props- Datei zum Lernen als Parameter übergeben. Nachdem die Lernphase beendet ist, ruft das Skript die gleiche Klasse erneut auf und verwendet die .props-Datei zum Testen. Der erste Aufruf erzeugt einen Tagger als Datei. Das Ergebnis des zweiten Aufrufs ist eine Aussage über die Anzahl der insgesamt getagten Sätze, der Anzahl an unbekannten Tokens, der Anzahl der korrekt und inkorrekt getaggten Sätze, korrekt und inkorrekt getaggten Tokens und den korrekt beziehungsweise inkorrekt getagten unbekannten Wörter.

Der zweite Schritt untersucht die Abhängigkeit eines Korpus im selben Genre von seiner Quelle. Damit soll gemessen werden, ob ein Tagger im eigenen Genre unabhängig von seiner Quelle genau so gut funktioniert wie in seiner eigenen. Da nur eine Quelle für den internetbasierten Text vorliegt kann nur der Unterschied zwischen dem Negra-Korpus und dem TUDZ untersucht werden. Der Ablauf unterscheidet sich vom ersten Schritt darin, dass nun zum Testen jeweils der vierte Teil des anderen Korpus genutzt wird. Der weitere Ablauf entspricht dem ersten Schritt.

Im letzten Schritt wird die Leistung eines Taggers außerhalb seines Genres ermittelt. Der Tagger des Nachrichten-Genres, also des Negra- und TUDZ wird mit jeder Testmenge des Internet-Genres getestet. Ebenso wird mit dem Internet-Genre basiertem Korpus verfahren. Der weitere Ablauf entspricht ebenfalls dem ersten Schritt.

3.5 Erwartete Ergebnisse

Erwartet wird, dass die Korpora in ihrem eigenen Genre bessere Performanz zeigen, als in dem jeweils anderem Genre. Ebenfalls wird jedoch erwartet, dass ein Tagger innerhalb des selben Genres schlechter abschneiden wird, wenn eine andere Quelle genutzt wird. Diese Erwartung basiert auf dem Prinzip des Probably Approximately Correct Learning

[86]. Dieses postuliert, dass eine Beispielmenge eine bestimmte Größe braucht, um effektiv gelernt zu werden. Außerdem muss die Verteilung der Lerndaten und der Testdaten gleich sein.

Ebenfalls wird erwartet, dass die geringe Größe des internetbasierten-Korpus schlechtere Ergebnisse in jedem Schritt des Versuchsablaufs bedingen könnte. In diesem Fall ist die vorliegende Arbeit als Modell für einen größeren Korpus zu betrachten, welches erneut angewandt werden kann, sobald ein größerer Korpus aufgebaut wurde.

3.6 Ergebnisse

Die Ergebnisse aus Schritt 1 zeigen in Tabelle 3.2 die Leistung der einzelnen Tagger auf ihren eigenen Genres. Der TUDZ und der Negra Korpus erreichen 95% korrekte Tags bei ihrem Durchlauf. Der EmpiriST2015 wurde mit den selben Parametern gelernt und erreichte 78%. Alle Korpora verlieren beim korrekten Taggen von ganzen Sätzen. Die beiden etablierten Korpora um die Hälfte und der EmpiriST20155 um ungefähr ein Vierfaches. Der Anteil an unbekannten Wörtern liegt prozentual bei 30%. Beim Negra-Korpus liegt dieser bei 13% und beim TUDZ bei 8%.

Korpus	∅ Getagte Wörter			∅ Getagte Sätze			∅ Unbekannte getaggte Wörter		
	Korrekt	Inkorrekt	Σ	Korrekt	Inkorrekt	Σ	Korrekt	Inkorrekt	Σ
sTUDZ	326669 (**95,68%**)	14741 (4,31%)	341410	10026 (**53,18%**)	8825 (46,81%)	18851	23279 (**82,8%**)	4834 (17,19%)	28113
Negra	76268 (95,05%)	3964 (4,94%)	80232	2368 (50,93%)	2281 (49,06%)	4649	8770 (82,5%)	1859 (17,49%)	10629
EmpiriST2015	1131 (78,01%)	318 (21,98%)	1449	23 (20,75%)	88 (79,24%)	111	209 (48,97%)	218 (51,02%)	427

Tabelle 3.2: Ergebnisse aus Schritt 1

Der zweite Schritt des Versuchsablaufs ergab die folgenden Ergebnisse in Tabelle 3.3. Beide Tagger erarbeiten dabei weniger korrekt getaggte Wörter, als bei ihren eigenen Testläufen aus Tabelle 3.2. Insgesamt ist der Wert für das korrekte Erkennen von Wörtern des TüBa-/DZ Taggers um 17% Punkte gefallen. Im selben Rahmen ist der Wert des Negra Taggers um 6% Punkte gefallen.

Lerndaten	Testdaten	∅ Getagte Wörter			∅ Getagte Sätze			∅ Unbekannte getaggte Wörter		
		Korrekt	Inkorrekt	Σ	Korrekt	Inkorrekt	Σ	Korrekt	Inkorrekt	Σ
sTUDZ	Negra	64972 (78,89%)	8948 (12,10%)	73920[1]	1238 (26,63%)	3411 (73,36%)	4694	5440 (67,35%)	2636 (32,64%)	8076
Negra	sTUDZ	303785 (**88,97%**)	37625 (11,02%)	341410	5357 (**28,41%**)	13494 (71,58%)	18852	41195 (**75,96%**)	13031 (24,03%)	54226

Tabelle 3.3: Ergebnisse aus Schritt 2

Tabelle 3.4 zeigt die Ergebnisse aus Schritt 3 des Versuchsablaufs. TUDZ und Negra bearbeiten 78% beziehungsweise 72% der Testdaten aus dem EmpiriST2015 korrekt. Das ist im Vergleich zu dem Tagger, welcher direkt aus EmpiriST2015 erstellt wurde, ein Unterschied von 0% Punkten beziehungsweise -6% Punkte. Das macht einen relativen Unterschied von 8% zwischen Negra und EmpiriST2015. TUDZ taggt im Vergleich zu den Tests auf seinen eigenen Daten -17% Punkte weniger korrekte Daten. Negra taggt -23% Punkte weniger.

Lerndaten	Testdaten	∅ Getagte Wörter			∅ Getagte Sätze			∅ Unbekannte getaggte Wörter		
		Korrekt	Inkorrekt	Σ	Korrekt	Inkorrekt	Σ	Korrekt	Inkorrekt	Σ
sTUDZ	EmpiriST2015	1137 (78,44%)	312 (21,55%)	1449	17 (15,84%)	94 (84,15%)	111	103 (46,24%)	119 (53,75%)	222
Negra	EmpiriST2015	1053 (72,63%)	396 (27,36%)	1449	10 (9,59%)	101 (90,4%)	111	148 (41,07%)	213 (58,92%)	361
EmpiriST2015	sTUDZ	233854 (68,49%)	107556 (31,50%)	341410	770 (4,08%)	81081 (95,91%)	18852	79366 (50,98%)	76291 (49,01%)	155657
EmpiriST2015	Negra	55030 (68,58%)	25202 (31,41%)	80232	168 (3,62%)	4481 (96,37%)	4694	19626 (51,99%)	18119 (48,00%)	37745

Tabelle 3.4: Ergebnisse aus Schritt 3

Der EmpiriST2015 Tagger hingegen hat 68% der Wörter der beiden anderen Korpora korrekt verarbeitet. Das macht einen Unterschied von -27% Punkten zu den Ergebnissen von Negra und TUDZ auf ihren eigenen Testdaten. Es macht zudem einen Unterschied von -10% zwischen den Ergebnissen auf den fremden Testdaten und den eigenen Testdaten von EmpiriST2015.

3.7 Zusammenfassung & Ausblick

Zusammenfassend lässt sich kein klares Bild aus den erarbeiteten Testergebnissen ziehen. Die ursprüngliche Annahme, dass ein Tagger in seinem Genre, hier der EmpiriST2015, besser arbeitet als ein anderer Tagger, konnte nicht bestätigt werden. Für die These sprechen die Ergebnisse aus dem dritten Schritt des Versuchsablaufs. Hier hat sich gezeigt, dass durchgehend die Tagger schlechter auf den Testdaten fremder Korpora arbeiten, als auf den eigenen. Im Fall von EmpiriST2015 konnte der TüBa-D/Z Tagger jedoch genau so gute Ergebnisse erzielen, wie der originale EmpiriST2015 Tagger. Vor dem Hintergrund muss jedoch beachtet werden, dass der TüBa-D/Z Tagger auf einen deutlich größeren Datenbestand zurückgegriffen hat und das Lernen dramatisch länger dauerte, als das Lernen des EmpiriST2015.

Die Erwartung, dass die Negra beziehungsweise TüBa-D/Z Tagger mit jeweils den gegenseitigen Testdaten schlechtere Ergebnisse erzielen als auf ihren eigenen wurde bestätigt. Dies spiegelt ein bekanntes Problem, das Tagger oft zu korpusspezifisch gelernt werden.

Bemerkenswert sind die schlechten Ergebnisse beim Taggen vollständiger Sätze. Kaum einer der Tagger konnte mehr als 50% aller Sätze vollständig korrekt verarbeiten. Dazu kommt, dass gerade bei den Ergebnissen aus Tabelle 3.4 besonders schlechte Ergebnisse bei allen Taggern auftraten. Im Rahmen von EmpiriST2015 ließe sich dies an dem kleinen Lerndatensatz erklären. Bei den etablierten Korpora lässt dies die Vermutung zu, dass gerade bei Chat basierten Korpora das Erkennen von Satzgrenzen und das Verarbeiteten von nachlässig verfassten Beiträgen zu diesen Ergebnissen führen könnten.

Ein klareres Bild mit eindeutigem Ergebnis ließe sich in Zukunft vielleicht durch einen größeren Datensatz für das Erlernen des Taggers für das Genre „Internet“ erreichen. Ein Ansatz, welcher im Rahmen des Kobra-Projekts [72] erarbeitet wird, ist die Schaffung eines deutschen Referenzkorpus zur internetbasierten Kommunikation [8], genannt DeRiK.

[1] Die Anzahl an Wörtern ist auf Grund eines noch nicht geklärten Fehlers des POS Tagger entstanden.

Dieser könnte mit vergleichbarer Anzahl von Tokens die Qualität des daraus abgeleiteten POS Taggers deutlich erhöhen und, wie auch die beiden etablierten Korpora Negra und TüBa-D/Z, ein besseres Ergebnis erzielen.

Ebenfalls müsste der Schritt 2 des Versuchsaufbaus um den Vergleich von zwei Taggern erweitert werden, welche auf annotierten Lerndaten aus der internetbasierter Kommunikation beruhen. Dadurch ließe sich erkennen, ob das Prinzip der Korpusspezifizität auch auf diese zutrifft und wenn ja, in welchem Maße.

Kapitel 4

Wörterbuchbasierte Stimmungserkennung zur Extraktion politischer Standpunkte aus Zeitungsartikeln

von Stefan Rötner

4.1 Motivation

Zeitungen sind ein wichtiges Medium zur Verbreitung aktueller Nachrichten und zwar im Idealfall in Form von objektiven Berichten. Trotzdem sind sie durch subjektive Artikel wie Kommentare und Glossen ein wichtiges Instrument der politischen Meinungsbildung. Aufgrund der politischen Ansichten und Ziele der Besitzer und Redakteure ist es naheliegend zu vermuten, dass Zeitungen auch stets eine gewisse politische Position zugeordnet werden kann. In zahlreichen Ländern kann man (zum Teil über staatliche Zeitungen und Fernsehsender) eine enge Verflechtung von Presse und Politik beobachten. Im Zuge der Pegida-Bewegung wird unter dem Schlagwort „Lügenpresse“ immer wieder die Anschuldigung erhoben, dass auch die *großen* deutschen Zeitungen durch gezielte Falschinformation und einseitige Berichterstattung die Leser manipulieren. Die Extraktion politischer Standpunkte mit Methoden des *Natural Language Processing* kann zum Beispiel in der Überprüfung dieser Hypothese oder in der generellen Analyse der Medienlandschaft Anwendung finden.

Frei verfügbare Ressourcen und Implementierungen für deutschsprachige Stimmungserkennung sind rar gesät und bezüglich Haltung annotierte deutsche Zeitungsartikel, die als Trainingsmenge für einen überwachten Lerner dienen könnten, sind nur in geringer Zahl verfügbar. Daher vergleiche ich zunächst verschiedene Methoden der Stimmungsanalyse bezüglich ihrer Anwendbarkeit auf deutschsprachige Zeitungstexte, um anschließend eine geeignete Implementierung für die Stimmungserkennung verwenden zu können.

Zum Schluss entwickele ich ein neues Verfahren zur Bestimmung der politischen Positionen auf Grundlage der Haltung zu bestimmten Streitthemen und führe ein konkretes

Experiment durch.

4.2 Stimmungserkennung

Das Ziel der Stimmungserkennung (engl. *Sentiment Analysis, Opinion Mining*) ist es, die Haltung (Polarität) eines Textes als positiv oder negativ (manchmal auch als neutral) zu erkennen. Oft wird zur Lösung dieses dreiklassigen Problems ein zweischrittiger Ansatz verwendet, bei dem zunächst überprüft wird, ob es sich um einen objektiven oder subjektiven Text handelt, und ein subjektiver Text anschließend als positiv oder negativ klassifiziert. In einigen Anwendungen, wie Produktrezensionen nach einem 5-Sterne-System, werden außerdem verschiedene Haltungsstärken (z.B. *positiv* und *sehr positiv*) unterschieden. Die beiden denkbaren Ansätze - überwachte Stimmungserkennung auf Basis annotierter Trainingsdaten und unüberwachte Methoden unter Verwendung regelbasierter Systeme und Stimmungswörterbücher - evaluiere ich im Folgenden.

4.2.1 Daten

Folgende Datensätze von Texten mit händisch vergebenener Polarität-Annotation werden in den Experimenten zur Evaluation verwendet:

MLSA-Korpus [1] Der Korpus[19] besteht aus 270 Sätzen mit 3 Ebenen von für die Stimmungserkennung relevanten Annotationen. Für meine Experimente habe ich die erste Ebene, bestehend aus ganzen Sätzen mit händischer Annotation für Subjektivität/Objektivität und Polarität, verwendet.

PressRelations DataSet [2] Der Datensatz besteht aus 1521 Aussagen aus 617 Nachrichtendokumenten, die mit der Polarität gegenüber jeweils einer oder beiden Parteien SPD und CDU annotiert sind.

Annotierte Artikel Um die Verfahren bezüglich der Eignung für das geplante Experiment zu evaluieren, habe ich 33 zufällige Artikel zum Thema *Mindestlohn* aus der Artikelsuche auf *TAZ.de* und *Welt.de* von Hand annotiert.

4.2.2 Überwachte Methoden

Da keine große Trainingsmenge aus annotierten Zeitungsartikeln zur Verfügung steht und die Annotation sehr aufwendig ist, kann nur entweder auf einem sehr kleinen Datensatz gelernt oder Produktbewertungen zum Training verwendet werden. Dabei steht zu erwarten, dass Modelle, die auf Produktrezensionen gelernt wurden, schlecht auf andere Domänen generalisieren.

[1] https://sites.google.com/site/iggsahome/downloads
[2] http://www.pressrelations.de/research/

Als Referenz zu den im Folgenden beschriebenen unüberwachten Ansätzen, sollen die Resultate des folgenden Rapidminer[3]-Experiments dienen:

Experiment 1: Die Beispieltexte werden nach einer Vorverarbeitung aus *Tokenizer*, *StopWordFilter*, *Stemmer* und *CaseTransformer* durch Wortvektoren mit TF-IDF Werten repräsentiert. Als Lernverfahren wird die LibSVM[16] mit einer One-Against-All Strategie verwendet. Falls in der Ergebnistabelle (Tabelle 4.1) Trainings- und Testdatensatz übereinstimmen, wurde eine 5-fache Kreuzvalidierung durchgeführt, ansonsten wurden jeweils die gesamten Datensätze für Training und Test verwendet.

	Accuracy	**Positiv**		**Neutral**		**Negativ**	
		Presicion	Recall	Precision	Recall	Precision	Recall
Train PR Test PR	69.03%	69,79%	58.52%	72,79%	60,37%	66,58%	84,39%
Train MLSA Test MLSA	45,93%	44,44%	11,59%	47,03%	86,36%	42,00%	23,08%
Train PR Test MLSA	40,74%	32,91%	37,68%	43,43%	69,09%	50,00%	8,79%

Tabelle 4.1: Ergebnisse der One-Against-All LibSVM

Auch in diesem Experiment lässt sich bestätigen, dass das verhältnismäßig gute Modell auf dem PressRelations-Datensatz nur schlecht auf den MLSA-Testsatz generalisiert. Auch der PressRelations-Datensatz ist ein relativ kleiner Trainingsdatensatz, sodass sich mit einem größeren Trainingsdatensatz oder einem aufwendigeren überwachten Verfahren sicherlich bessere Ergebnisse erzielen lassen. Trotzdem gehe ich (insbesondere solange keine bedeutend größeren deutschen Trainingsmengen verfügbar sind) davon aus, mit einem unüberwachten Verfahren stabilere Vorhersagen treffen zu können.

4.2.3 Unüberwachte Methoden

Unüberwachte Methoden basieren auf der Idee, dass bestimmten Worten unabhängig vom Kontext eine a Priori-Stimmung zugeordnet werden kann und die Gesamtstimmung eines Textes durch Aggregation der a Priori-Stimmungen der Wörter des Textes ermittelt werden kann. Im einfachsten Fall erfolgt die Aggregation durch Mittelwertbildung.

Im Folgenden betrachte ich eine Stimmung daher nicht als polynomiale Klasse, sondern als Zahl x, wobei $signum(x)$ der Polarität und $|x|$ der Stärke entspricht. Durch die Verwendung der harten *signum*-Regel genügt bereits ein einziges Stimmung ausdrückendes Wort, damit ein Text nicht als neutral eingestuft wird. Trotzdem erscheint dies aufgrund folgender Beobachtung eine gute Lösung zu sein: Obwohl insbesondere Berichte in Zeitungen objektiv sein sollten, taucht die Klasse *neutral* in der von mir per Hand annotierten Stichprobe der zu untersuchenden Artikel nur sehr selten auf. Auch in den Positionen der

[3] https://rapidminer.com/

Parteien sind neutrale Positionen sehr selten und daher für die Ermittlung einer politischen Position weniger interessant.

Beispielsweise mit für den *GErman SenTiment AnaLysis shared Task* entwickelten Methoden zur Subjektivitätserkennnung [83, 22, 68] oder auch der naiven in Abschnitt 4.2.6 beschriebenen Methode lassen sich neutrale Artikel im Vorhinein herausfiltern. Daher betrachtete ich bei den Experimenten jeweils die Vorhersagegenauigkeit für das dreiklassige Problem und andererseits bezüglich des binären Problems auf einem Testdatensatz, aus dem alle neutralen Beispiele entfernt wurden. Binär bedeutet dabei, dass die Vorhersagegenauigkeit bezüglich positiver und negativer Vorhersagen bestimmt wird, während neutrale Vorhersagen (also objektive Artikel) ignoriert werden.

4.2.4 Stimmungswörterbücher

Stimmungswörterbücher ordnen in der Regel einer Kombination aus Wort (bzw. Stammform) und *Part-of-speech (POS) Tag* die a Priori-Stimmung zu. Stimmung ausdrückende Worte (im Folgenden *Stimmungsworte*) und ihre a Priori-Stimmung lassen sich entweder von Hand oder auf Grundlage der Anzahl gemeinsamer Vorkommen mit einer bestimmten Polarität oder bereits identifizierten Stimmungsworten in großen Textkorpora bestimmen. Auf annotierten Korpora können außerdem die Gewichte, die ein gutes Lernverfahren für die einzelnen Einträge des Wortvektors berechnet, als Anhaltspunkte dienen.

PMI-IR Ein statistisches Maß für die Abhängigkeit zweier Wörter auf Grundlage der Wahrscheinlichkeit für ein gemeinsames Auftreten, ist die *Pointwise Mutual Information (PMI)* :

$$PMI(\text{Wort1}, \text{Wort2}) = log_2 \left[\frac{p(\text{Wort1} \wedge \text{Wort2})}{p(\text{Wort1}) \cdot p(\text{Wort2})} \right]$$

Turney[79] schlägt vor die Stimmung eines Wortes (*Semantic Orientation*) aus den PMI-Werten des Wortes mit einem bekannten positiven und einem bekannten negativen Wort zu berechnen:

$$SO(\text{Wort}) = PMI(\text{Wort}, \text{„excellent“}) - PMI(\text{Wort}, \text{„poor“})$$

Weiterhin schlägt Turney vor die Wahrscheinlichkeiten für das Vorkommen der Worte anstatt durch Zählen auf einem riesigen Korpus durch die Anzahl der Treffer einer Suchmaschine zu approximieren:

$$\begin{aligned} SO(\text{Wort}) &= log_2 \left[\frac{hits(\text{Wort} \wedge \text{„excellent“})}{hits(\text{Wort}) \cdot hits(\text{„excellent“})} \right] - log_2 \left[\frac{hits(\text{Wort} \wedge \text{„poor“})}{hits(\text{Wort}) \cdot hits(\text{„poor“})} \right] \\ &= log_2 \left[\frac{hits(\text{Wort} \wedge \text{„excellent“}) \cdot hits(\text{„poor“})}{hits(\text{Wort} \wedge \text{„poor“}) \cdot hits(\text{„excellent“})} \right] \end{aligned}$$

SentiWS SentiWS[64] ist eines der wenigen frei verfügbaren[4] Wörterbücher für deutschsprachige Stimmungserkennung und enthält 1.650 positive und 1.818 negativen Stamm-

[4] http://asv.informatik.uni-leipzig.de/download/sentiws.html

formen. Einer Kombination aus Stammform und POS-Tag wird jeweils eine Stimmung aus [-1,1] zugeordnet. Das Wörterbuch wurde unter Verwendung verschiedener Methoden und Quellen zusammengestellt (z.B. Übersetzung englischer Wörterbücher und Verwendung der PMI mit mehreren positiven und negativen deutschen Stimmungswörtern).

Zur Evaluation des puren PMI-IR Ansatzes ohne menschliches Expertenwissen, habe ich in einem Experiment auf dem MLSA-Datensatz die Stimmungen der einzelnen Worte anhand der Trefferzahlen der Suchmaschine *Bing*[5] über die Bing-Websearcb-API verwendet. Die große Schwierigkeit liegt hier in der Wahl geeigneter positiver und negativer Referenzen. Zur Bestimmung geeigneter Referenzen, habe ich verschiedene Mengen von Referenzwörtern auf einigen Testbegriffen evaluiert. Bei allen Referenzmengen bevorzugt das Verfahren eine Polarität. Obwohl sich auf den Testbegriffen durch Addition eines entsprechenden Offset plausible Ergebnisse erreichen lassen, konnte keine Referenzmenge (auch nicht die zur Bestimmung der Polaritäten von SentiWS verwendete [64]) auf den Testdatensätzen konkurrenzfähige Ergebnisse liefern.
Die Vorhersage der Stimmung durch Mittelwertbildung der Priori-Stimmungen erreicht mit SentiWS dagegen die in Tabelle 4.2 beschriebene Genauigkeit.

	SentiWS
MLSA	70% 47%
PressRelations	65% 51%
Annotierte Artikel	56% 42%

Tabelle 4.2: Ergebnisse der Stimmungserkennung mit einfacher Mittelwertbildung jeweils bezüglich des *vollständigen* Datensatzes und der binären Stimmungserkennung auf dem Datensatz *ohne neutrale* Beispiele.

4.2.5 Stimmungsberechnung

Oft stehen Stimmungsworte in Verbindung mit *Relativierungen* (z.B. *weniger* gut, *sehr* schlecht) und Negationen (z.B. *nicht* gut) welche die Stimmung verändern, obwohl sie selbst nicht mit einer a Priori-Stimmung im Wörterbuch zu finden sind. Um die Vorhersagegenauigkeit gegenüber der Aggregation der a Priori-Stimmungen durch Mittelwertbildung zu verbessern, sollen nun Negationen und Relativierungen berücksichtigt werden. In Analogie zu den beiden relevanten Arbeiten, gehe ich nun davon aus, dass Priori-Stimmungen durch ganzzahlige Werte aus [-5,5] ausgedrückt werden.
Eine einfache Auflösung von Negation lässt sich durch Multiplikation der Stimmung mit -1 erreichen. Für eine Relativierung lässt sich annehmen, dass sich die Stimmungsstärke um 1 erhöht oder erniedrigt [63]. Die Berechnung erfolgt dabei rekursiv von der Wurzel ausgehend:

- sehr gut$\{3\} \hat{=} 1 + 3 = 4$
- nicht gut$\{3\} \hat{=} (-1) * 3 = -3$

[5] http://www.bing.com/

- nicht sehr gut$\{3\} \hat{=} (-1) * (1 + 3) = -4$
- sehr schlecht$\{-3\} \hat{=} -1 - 3 = -4$

Insbesondere gilt mit dieser Form der Negationsauflösung also, dass die Stimmung von *nicht sehr gut* gleich der Stimmung von *sehr schlecht* ist. Die Arbeit zum *Semantic Orientation Calculator (SO-CAL)* [74] schlägt daher vor, die Negation stattdessen durch Veränderung um einen bestimmten Offset entgegen des Vorzeichens der Stimmung zu verändern (*Shift Negation*). Dabei wurde ein Offset von 4 vorgeschlagen:

- nicht gut$\{3\} \hat{=} 3 - 4 = -1$
- nicht sehr gut$\{3\} \hat{=} (1 + 3) - 4 = 0$

Für die Implementierung(SO-CAL2) verwende ich die deutsche Relativierungen- und Negationenliste, die von *Senti Strength*[6] bereitgestellt werden. Zur Negationsauflösung verwende ich die Shift-Negation mit Offset 4. Berücksichtigt werden die Relativierungen, die unmittelbar vor dem Wort stehen. Negationen können (bedingt durch die flexible deutsche Satzstellung noch häufiger als im Englischen) jedoch sehr weit von dem betroffenen Stimmungswort entfernt stehen. In der Ausarbeitung [74] wird ein komplexes regelbasiertes System auf Basis der Interaktion verschiedener POS-Tags mit Negation angesprochen, aber nicht beschrieben. Ich verwende daher zunächst eine sehr einfache Heuristik, indem ich vor dem Stimmungswort bis zum nächsten Nomen oder Verb nach einer Negation suche.

Um die Stimmungswerte aus SentiWS (aus dem Intervall $[-1; 1]$) mit SO-CAL2 verwenden zu können, wähle ich zwei verschiedene Ansätze. Zum einen **(A)** habe ich SentiWS zusätzlich in eine Darstellung mit ganzzahligen Stimmungswerten aus dem Intervall $[-5; 5]$ transformiert. An einem Histogramm der in SentiWS gespeicherten Stimmungswerte sieht man schnell, dass nur einige Werte zwischen $[-1; 1]$ angenommen werden, sodass die Transformation mittels fester Klassengrenzen zwischen den Häufungsgebieten der Stimmungsstärken möglich ist. Der Durchschnitt der Stimmungsstärken im transformierten Lexikon liegt bei -0,394; positive und negative Stimmungen sind also zu gleichen Teilen vertreten. Zum anderen **(B)** habe ich die für SO-CAL vorgeschlagenen Konstanten proportional in in das Intervall $[-1; 1]$ abgebildet (Negations-Offset $-0,8$ und Änderungsrate für einen Relativierer $0,2$).

Tabelle 4.3 zeigt die Ergebnisse von SO-CAL2 und dem naiven Ansatz mit Durchschnittsbildung im Vergleich. Zwecks Vergleichbarkeit wurde für den naiven Ansatz die transformierte Version von SentiWS verwendet. Die SO-CAL2-Variante mit dem transformierten Wörterbuch erreicht auf den Testdatensätzen bessere Ergebnisse als die Varianten mit transformierten Konstanten. Im Vergleich mit der Durchschnittsbildung sind jedoch keine deutlichen Verbesserungen zu beobachten. Mögliche Ursachen sind die sehr einfache Heuristik zur Ermittlung zugehöriger Negationen und die verhältnismäßig kurze Relativiererliste.

[6] http://sentistrength.wlv.ac.uk/

	Average	SO-CAL2 (A)	SO-CAL2 (B)
MLSA	75% 50%	74% 49%	67% 46%
PressRelations	65% 50%	63% 49%	63% 50%
Annotierte Artikel	50% 36%	54% 39%	48% 36%

Tabelle 4.3: Ergebnisse der Stimmungserkennung mit einfacher Mittelwertbildung jeweils bezüglich des *vollständigen* Datensatzes und der binären Stimmungserkennung auf dem Datensatz *ohne neutrale* Beispiele.

4.2.6 Subjektivtätserkennung

Bisher wurden die verschiedenen Methoden jeweils auch in Bezug auf binäre Stimmungserkennung auf einem Datensatz ohne neutrale Beispiele analysiert, um einen Anhaltspunkt zu liefern, wie gut die die Verfahren mit einem vorgeschalteten Subjektivitätsfilter arbeiten. Eine einfaches Kriterium für die Subjektivität eines Textes ist der Anteil der Stimmungswörter an der Gesamtzahl. Ein Schwellwert zur Unterscheidung subjektiver und objektiver Artikel lässt sich mit Hilfe der entsprechend annotierten Datensätze experimentell bestimmen.
In Bezug auf den konkreten Anwendungsfall der Extraktion politischer Meinungen aus Zeitungsartikeln gehe ich davon aus, dass bessere Ergebnisse erzielt werden, wenn nur stark subjektive Texte betrachtet werden (je höher der Schwellwert gewählt wird, desto mehr Stimmungsworte stehen zur Bestimmung der Stimmung zur Verfügung). Mit anderen Worten, solange genügend subjektive Artikel zur Verfügung stehen, darf der Subjektivitätsfilter auch subjektive Artikel als objektiv klassifizieren. Mit einem Schwellwert von $0,07$ lässt sich auf den annotierten Artikeln ein Recall von 1 für die Klasse *objektiv* und eine Precision von 1 für die Klasse *subjektiv* erreichen.
Tabelle 4.4 zeigt die Erkennungsrate mit Schwellwert $0,07$ auf den Testdatensätzen und wie sich die Verwendung des Filters auf die Stimmungserkennung auswirkt. Die vorgeschaltete Subjektivitätserkennung führt dabei wie erwartet zu einer deutlichen Verbesserung der Stimmungserkennung. Die Werte, die sich auf dem Datensatz ohne neutrale Beispiele (also mit optimaler Subjektivitätserkennung) ergaben, werden für die ersten beiden Testdatensätze jedoch nicht erreicht. Hier sind die Ergebnisse für die einfache Durchschnittsbildung deutlicher besser als die für SO-CAL2.

	Erkennung	Average ohne Filter	Average	SO-CAL2
MLSA	61%	70% 47%	54%	52%
PR	55%	65% 51%	59%	57%
Artikel	45%	56% 42%	57%	43%

Tabelle 4.4: Ergebnisse der Subjektivitätserkennung und Vergleich der binären Stimmungserkennung mit Subjektivitätsfilter mit der einfachen Mittelwertbildung ohne Filter (jeweils bezüglich des *vollständigen* Datensatzes und der binären Stimmungserkennung auf dem Datensatz *ohne neutrale* Beispiele).

4.2.7 Themenbezogene Stimmungserkennung

Häufig drückt ein einzelner Satz gegenüber verschiedenen Aspekten unterschiedliche Stimmungen aus, sodass sich die Stimmung bezüglich eines Themas von der Gesamtstimmung des Satzes unterscheidet. In langen Zeitungsartikeln ist dieses Problem offensichtlich noch häufiger anzutreffen als auf Satzebene oder in Rezensionen zu einem einzelnen Produkt. Als einfachen Ansatz zur Berücksichtigung dieser Problematik habe ich die beiden folgenden Verfahren evaluiert:

Satzselektion Für die Berechnung der Stimmung bezüglich eines Aspekts wird ein Stimmungswort nur berücksichtigt, wenn im gleichen Satz auch der Aspekt erwähnt wird.

Dependenzstrukturselektion Um der aspektbezogenen Stimmungsanalyse auf Satzebene gerecht zu werden, werden nur Stimmungsworte berücksichtigt, die im Strukturbaum bezüglich einer Dependenzgrammatik Nachfahren des zu bewertenden Aspekts sind.

Um mehr relevante Stimmungsworte zu finden, suche ich nicht nur direkt nach dem Aspekt sondern auch nach den im OpenThesaurus[58] hinterlegten Synonymen.

Für einen experimentellen Vergleich der Methoden verwende ich den entsprechend annotierten PressRelations-Datensatz bezüglich der Themen *CDU* und *SPD* und die annotierten Artikel bezügliche des Aspektes *Mindestlohn*. Dennoch sind beide Testdatensätze in diesem Zusammenhang nicht besonders repräsentativ, da die Texte des PressRelations-Datensatzes nur zwei bis vier Sätze enthalten und der Artikel-Datensatz aus nur 33 Artikel besteht). Zur Berechnung der Stimmung der selektierten Stimmungsworte verwende ich die Durchschnittsbildung mit SentiWS und SO-CAL2 in Kombination mit der transformierten Version von SentiWS.
Bei der Evaluierung auf den verfügbaren Testdatensätzen (Tabelle 4.5) liefert die Satzselektion leicht schlechtere Vorhersagen als das naive Verfahren ohne aspektorientierte Selektion. Die binären Vorhersagen der Dependenzselektion erreichen auf dem PressRelations-Datensatz mit 91% die höchste Genauigkeit. Der geringe Wert von 33% bezüglich der dreiklassigen Vorhersage ist darauf zurückzuführen, dass nur in seltenen Fällen die präzisen im Dependenzbaum unterhalb liegenden Stimmungsworte zur Verfügung stehen und entsprechend viele Artikel mangels Stimmungswörter als objektiv klassifiziert werden. Auf dem sehr kleinen Testdatensatz der annotierten Artikel werden sogar alle Artikel als objektiv eingestuft. Die Dependenzselektion liefert also, falls Stimmungsworte selektiert werden, sehr präzise Vorhersagen, ist aber allgemein nicht dazu geeignet relevante Stimmungswörter zu identifizieren.

4.2.8 Implementierung

Die Vorverarbeitungsschritte Satz-Tokenisierung, Wort-Tokenisierung und POS-Tagging habe ich mit openNLP[7] unter Verwendung der bereitgestellten deutschsprachigen Model-

[7] https://opennlp.apache.org/

	Keine Selektion	Satzselektion		Dependenzselektion	
	Average	Average	SO-CAL2	Average	SO-CAL2
PR	65% 51%	65% 42%	64% 41%	91% 33%	91% 33%
Artikel	56% 42%	52% 42%	48% 39%	- -	- -

Tabelle 4.5: Ergebnisse der aspektbezogenen Stimmungserkennung jeweils bezüglich des *vollständigen* Datensatzes und der binären Stimmungserkennung auf dem Datensatz *ohne neutrale* Beispiele.

le[8], die jeweils auf dem Tiger-Korpus trainiert wurden, durchgeführt. Die Ermittlung der Stammformen der Wörter und das Parsen der Dependenzstruktur habe ich mit MateTools [9][12, 11] mit den bereitgestellten deutschen Modellen [70] durchgeführt.

4.3 Extraktion politischer Positionen

Als einfache Methode zur Extraktion politischer Haltungen aus Texten schlage ich vor, eine politische Haltung durch einen Vektor von Stimmungen zu bestimmten Streitthemen auszudrücken, an denen sich Haltungen üblicherweise unterscheiden. Die Ähnlichkeit zweier Haltungen lässt sich dann mit den üblichen Vektormetriken wie Betragsnorm, euklidischer Norm und Hamming-Distanz, vor allem aber über die Kosinus-Ähnlichkeit beschreiben. Damit lässt sich eine politische Haltung unter Verwendung eines beliebigen Verfahrens zu Stimmungsanalyse ermitteln. Im Folgenden zeige ich in einem kleinen Experiment, wie das Verfahren mit der oben entwickelten Methode verwendet werden kann, um politische Haltungen von Zeitungen zu vergleichen und die Korrelation mit politischen Parteien zu überprüfen.

4.3.1 Konkretes Experiment

In diesem konkreten Experiment vergleiche ich die Haltung der beiden großen deutschen Tageszeitungen *TAZ* und *WELT* mit den Haltungen der 5 größten deutschen Parteien *CDU*, *SPD*, *FDP*, *Die Linke* und *Bündnis90/Die Grünen*.

Bestimmung der Positionen politischer Parteien Die Formulierung der Haltungen politischer Parteien ist ein sehr subjektiver Vorgang. Für mein Experiment verwende ich daher die Streitthemen, die der Wahl-O-Mat[10] zur Bundestagswahl 2013 verwendet hat, und extrahiere die Haltung der Parteien aus den Antworten, welche die Parteien damals in Form von (Zustimmung +, Enthaltung 0, und Ablehnung -) selbst zu diesen Fragen gegeben haben. Leider lassen sich nicht alle Fragen sinnvoll mit einem einzelnen Wort als Streitthema formulieren, sodass nur 17 Streitthemen extrahiert wurden (siehe Tabelle 4.6). Zusätzlich habe ich aus diesen 17 Themen noch einmal diese ausgewählt, die ich

[8] http://opennlp.sourceforge.net/models-1.5/
[9] https://code.google.com/p/mate-tools/
[10] https://www.wahl-o-mat.de/bundestagswahl2013/

intuitiv für besonders geeignet halte, politische Haltungen voneinander zu trennen.

Artikel-Crawling Für das Experiment habe ich für jedes Streitthema einen Crawler für die Treffer der Artikelsuche zu diesem Thema auf der jeweiligen Website[11] verwendet und die extrahierten Artikel zur Bestimmung der Position verwendet. Da die Positionen der Parteien zu einigen Themen sich z.B. aufgrund von Koalitionsverträgen im Laufe der Zeit verändern, verwende ich für das Experiment nur die Artikel, die vor der Bundestagswahl 2013 geschrieben wurden.

Zur Stimmungserkennung verwende ich für das Experiment die oben beschriebene Methode zur Subjektivtitätserkennung mit Schwellwert $0,07$ in Kombination mit der Durchschnittsmethode mit dem SentiWS-Wörterbuch jeweils mit Satzselektion, Dependenzselektion und ohne Selektion.

4.3.2 Ergebnisse

Für jeden Artikel einer Zeitung zu einem Thema habe ich eine Polarität (1,0,-1) ermittelt und als Gesamtposition der Zeitung die Polarität über alle subjektiven Artikel zu diesem Thema gemittelt. Mit Dependenzselektion wurden wie in den obigen Experimenten nicht genügend subjektive Artikel ermittelt. Die Ergebnisse der Satzselektion und des Verfahrens ohne Selektion finden sich in Tabelle 4.6. Eine Bewertung kann anhand Positionen der Parteien zu diesen Thema und der intuitiven Partei-Zeitungs-Ähnlichkeit durchgeführt werden. Nach subjektiver Einschätzung des Autors sollte die WELT eher der CDU nahestehen und die TAZ im linken Bereich des politischen Spektrums verortet sein.

	Satz-Selektion		**Keine Selektion**		**CDU**	**SPD**	**FDP**	**LINKE**	**GRÜNE**
	TAZ	WELT	TAZ	WELT					
Mindestlohn	-0,04	-0,13	-0,57	-0,49	-	+	-	+	+
PKW Maut	-0,44	-0,34	-0,63	-0,50	0	-	-	-	0
Tempolimit	-0,08	-0,13	-0,44	-0,29	-	-	-	+	+
Videoüberwachung	-0,05	0,13	-0,72	-0,51	+	0	-	-	-
Betreuungsgeld	-0,32	-0,49	-0,60	-0,64	+	-	0	-	-
NATO	-0,24	-0,27	-0,79	-0,64	+	+	+	-	+
Kohlekraft	-0,05	-0,02	-0,46	-0,52	+	+	+	-	-
Verstaatlichung	-0,10	-0,19	-0,77	-0,66	-	-	-	+	-
Flüchtlinge	-0,58	-0,51	-0,73	-0,64	-	+	0	+	+
Parteiverbot	-0,36	-0,47	-0,74	-0,71	+	+	+	+	+
Frauenquote	0,23	0,01	-0,32	-0,08	0	+	-	+	+
Länderfinanzausgleich	-0,11	-0,11	-0,57	-0,48	0	+	+	+	+
Rüstungsexporte	-0,34	0,00	-0,54	-0,65	+	+	+	-	0
Ehegattensplitting	0,13	0,04	-0,42	-0,52	+	0	+	-	-
Eurobonds	-0,30	-0,13	-0,87	-0,77	-	0	-	+	+
Vorratsdatensp.	-0,24	-0,26	-0,74	-0,81	+	+	-	-	-
Mietpreisbremse	-0,14	-0,27	-0,50	-0,67	+	+	-	+	+

Tabelle 4.6: Übersicht über die ermittelte Stimmung der Zeitungen zu einigen Streitthemen und die Positionen der Parteien zu diesen Streitthemen.

[11] http://www.taz.de/ und http://www.welt.de/

Zunächst einmal fällt auf, dass nahezu alle ermittelten Stimmungen negativ sind, obwohl die beiden Verfahren auf den Testdatensätzen ausgewogene Vorhersagen geliefert haben. Eine mögliche Erklärung dafür ist, dass Meinungen in Zeitungen überwiegend in Form von Kritik an anderen Positionen und Äußerungen geäußert werden. Wenn man aber die Stärken der Stimmungen vergleicht entsprechend vor allem bei Verwendung der Satz-Selektion die Ergebnisse überwiegend den Erwartungen. So ist beispielsweise die Position der TAZ zur *PKW Maut* negativer, als die der WELT. Ein von mir im Vorhinein als relativ gut geeignet eingestuftes Thema zur Unterscheidung, das bei beiden Verfahren nicht die erwarteten Ergebnisse liefert, ist das Thema *Betreuungsgeld*.
Die überwiegend negative Stimmung in Zeitungsartikeln durch einen festen Offset auszugleichen ist schwierig, da die Stärken der Stimmungen zwischen den Themen stark variieren. Ich betrachte daher gemäß der Vorstellung, dass zumindest der Grad der Ablehnung oder Zustimmung korrekt ist, eine Verschiebung der Stimmungen pro Thema um die gemittelte Stimmung der beiden Zeitungen zu diesem Thema. Mit anderen Worten wird die positivere Stimmung als positiv und die negativere als negativ interpretiert.

Der Vergleich mit einer anderen Position erfolgt mittels der Kosinus-Ähnlichkeit, also anhand des Winkels zwischen den beiden Vektoren. Ein Wert von 1 (Winkel 0°) entspricht dabei einer vollen Übereinstimmung, und -1 beschreibt genau entgegengesetzte Positionen.

Unter Verwendung der von mir im Vorhinein als besonders geeignet eingestuften Themen (*Mindestlohn, Frauenquote, PKW Maut, Tempolimit, Videoüberwachung, Verstaatlichung, NATO* und *Kohlekraftwerk*) ergeben sich für die Kosinus-Ähnlichkeit der berechneten Zeitungspositionen mit den Parteipositionen die Werte in Tabelle 4.7. Die Themen können erneut nur mit Hilfe des eigenen subjektiven Empfinden bewertet werden. Dabei entspricht die resultierende Ordnung genau den Erwartungen des Autors. Bei Verwendung des Kritik-Offsets entsprechen außerdem die Vorzeichen der Kosinus-Ähnlichkeit den Erwartungen (siehe zusätzlich Abbildung 4.1).

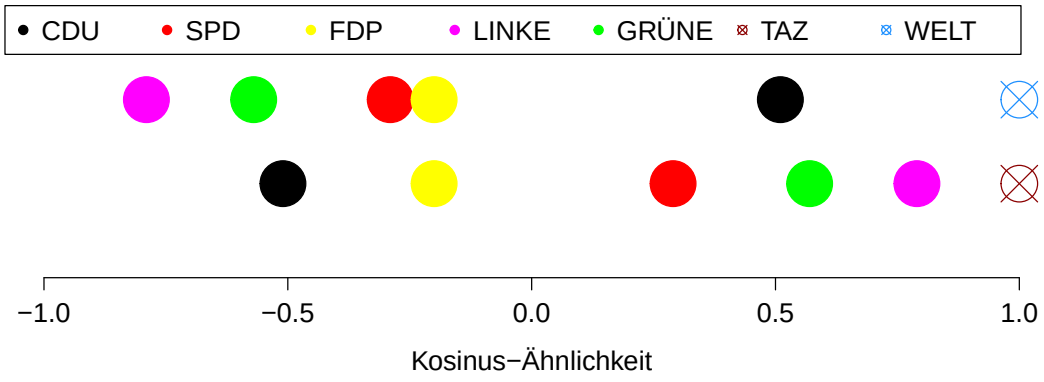


Abbildung 4.1: Grafische Darstellung der mit als wichtig eingestuften Themen und Kritik-Offset berechneten Ähnlichkeit der Parteimeinungen zu den Haltungen von WELT (oben) und TAZ (unten).

	Original		Mit Kritik-Offset	
	TAZ	WELT	TAZ	WELT
CDU	-0,08	0,24	**-0,51**	**0,51**
SPD	0,35	0,19	**0,29**	**-0,29**
FDP	0,12	0,26	**-0,20**	**0,20**
Die Linke	0,48	0,03	**0,79**	**-0,79**
Bündnis 90 / Die Grünen	0,05	-0,31	**0,57**	**-0,57**

Tabelle 4.7: Vergleich der Distanz der Parteipositionen zu den verschiedenen Zeitungen anhand der ausgewählten Streitthemen.

Vergleicht man jedoch die Zeitungshaltungen bezüglich aller extrahierten Streitthemen mit den Parteipositionen (Tabelle 4.8), entsprechen die Ergebnisse nicht mehr ganz den Erwartungen. Insbesondere die Ähnlichkeit zur den Positionen der CDU erscheint zweifelhaft.

	Original		Mit Kritik-Offset	
	TAZ	WELT	TAZ	WELT
CDU	-0,05	-0,13	0,18	-0,18
SPD	-0,16	-0,21	0,15	-0,15
FDP	0,12	0,12	0,00	0,00
Die Linke	-0,05	-0,16	0,26	-0,26
Bündnis 90 / Die Grünen	-0,24	-0,29	0,12	-0,12

Tabelle 4.8: Vergleich der Distanz der Parteipositionen zu den verschiedenen Zeitungen anhand aller geeigneten Streitthemen.

Die geringere Übereinstimmung bei Verwendung aller Streitthemen, resultiert daher, dass viele Streitthemen, die nicht Teil meiner subjektiven Auswahl für geeignete Themen waren, nicht die erwarteten Zeitungspositionen liefern (z.B. *Betreuungsgeld*, *Eurobonds*). Eine offensichtliche mögliche Ursache ist natürlich ungenaue Stimmungserkennung; andererseits ist nicht klar, dass die von mir erwartete Korrelation von Zeitungsmeinungen und Parteipositionen tatsächlich existiert. Sollten tatsächlich einige Themen besser geeignet sein, als andere, wäre ein Kriterium zu Identifikation dieser Themen wünschenswert. Tabelle 4.9 zeigt die Ergebnisse eines Vergleichs anhand der beiden Themen mit der größten Differenz zwischen den Zeitungsmeinungen (*Frauenquote* und *Videoüberwachung*). Bereits das Thema mit der drittgrößten Differenz ist jedoch *Betreuungsgeld*, sodass ein Vergleich anhand dieser Themen keine plausiblen Ergebnisse liefert.

4.4 Fazit und Ausblick

Mit dem Stimmungswörterbuch SentiWS und der Stimmungsaggregation über Mittelwertbildung existiert ein Verfahren für unüberwachte Stimmungserkennung als Alter-

	Original		Mit Kritik-Offset	
	TAZ	WELT	TAZ	WELT
CDU	-0,21	1,00	-0,62	0,62
SPD	0,98	0,07	0,78	-0,78
FDP	-0,54	-0,75	-0,11	0,11
Die Linke	0,84	-0,66	0,99	-0,99
Bündnis 90 / Die Grünen	0,84	-0,66	0,99	-0,99

Tabelle 4.9: Vergleich der Distanz der Parteipositionen zu den verschiedenen Zeitungen anhand der beiden Streitthemen mit der größten Differenz zwischen den Zeitungsmeinungen (*Frauenquote* und *Videoüberwachung*).

native zu überwachten Verfahren, die schlecht auf andere Domänen generalisieren. Die Berechnung der Stimmungen aus PMI-Werten gemäß der Trefferzahlen einer Suchmaschine, lieferte keine zuverlässigen Vorhersagen. Auch die Berücksichtigung von Negationen und Relativierern gemäß SO-CAL, brachte mit der einfachen Heuristik für das Auffinden relevanter Negationen und der kurzen Relativierer-Liste nicht die erhofften Verbesserungen.

Das größte Verbesserungspotential sehe ich in der aspektbezogenen Stimmungsanalyse, z.B. durch Erweiterungen wie Koreferenzauflösung oder die Identifizierung von Quelle und Ziel für Stimmungsworte oder der Verwendung von *Topic Models* anstelle von einfachen Synonymlisten. Auch der naive Ansatz der Satz-Selektion liefert jedoch für die Extraktion der Zeitungspositionen deutlich plausiblere Ergebnisse.

Die Arbeit zeigt, wie Stimmungserkennung zur Extraktion politischer Meinungen eingesetzt werden kann. Im konkreten Experiment mit einer naiven Methode zur Stimmungserkennung entsprechen zumindest für eine Menge von als besonders relevant eingestuften Themen die Ergebnisse genau den Erwartungen. Die Identifikation für die Stimmungserkennung geeigneter Themen ist jedoch ein schwierige Problem und auch die Verbesserung der generellen Stimmungserkennung bietet viel Potential die Stimmungsvorhersagen unter Verwendung aller Themen zu verbessern. Die Übersetzung der Wahl-O-Mat-Fragen in einzelne Streitthemen war deutlich schwieriger als erwartet. Eine sinnvolle Alternative könnten hier *Topic Models* und die berechneten Stimmungen aus Politiker-Blogs oder Bundestagsreden sein.

Kapitel 5

Untersuchung eines politisch inkorrekten Begriffs mittels Latent Dirichlet Allocation

von Sibylle Hess

5.1 Einführung

Antidiskriminierende Bezeichnungen von Menschen mit spezifischen Merkmalen, betreffe es nun das Geschlecht, die Abstammung oder körperliche bzw. geistige Defizite, unterliegen in ihrer Bedeutung häufig einem Wandel der Zeit. Ehemals aufwertend gemeinte Bezeichnungen wie *Behinderte* zum zuvor verbreiteten Begriff *Krüppel* werden mittlerweile als denunzierend empfunden und die Benutzung eines anderen Begriffs wie *Menschen mit Behinderung* empfohlen[1]. Begriffe zur Bezeichnung von Menschen fremder Herkunft unterliegen einer besonders starken Bedeutungsverschiebung, wie an den zahlreichen Bezeichnungen wie *Asylant*, *Schwarzer*, *Zigeuner* oder *Ausländer*, die zumindest vom Duden als abwertend gekennzeichnet wurden, aber im täglichen Sprachgebrauch durchaus ohne wertende bzw. diskriminierende Bedeutung benutzt werden, deutlich wird.

Unabhängig von der Debatte über die angemessene Wortwahl und ob normierende Spracheingriffe vorgenommen werden sollten, ist anzunehmen, dass sich politisch inkorrekte Begriffe durch Nutzung in negativen oder problematischen Kontexten über einen gewissen Zeitraum ergeben. Um Stimmungen oder geteilte Ansichten einer Bevölkerung zu diesem Thema aufzunehmen, ist eine Analyse über die Benutzung von Begrifflichkeiten in Abhängigkeit von den gegebenen Kontexten von Interesse. Ein Versuch zu einer solchen Analyse soll im Folgenden anhand der exemplarischen aber vielschichtigen Begriffen zum Asylerwerb unternommen werden.

5.2 Methodik

Als Methode wird hierzu die *Latent Dirichlet Allocation* (LDA) [10] gewählt, ein generatives probabilistisches Topic Modell, welches von Blei et al. zur Identifikation zugrunde

[1] http://www.duden.de/rechtschreibung/Behinderter (aufgerufen am 21.2.15)

liegender Themen einer Sammlung von Dokumenten vorgeschlagen wurde. Brody et al. zeigten 2009 die Anwendbarkeit dieses Verfahrens auf den Bereich der *Word Sense Disambiguation* [14], der automatischen Extraktion von verschiedenen Bedeutungen eines Wortes. Der Begriff *Topic Modelling* umfasst i.A. unüberwachte Lernverfahren, welche zur Annotation und Extraktion der immanenten Thematik, der Erkennung von Abhängigkeiten verschiedener Themen und ihrer Veränderung im Laufe der Zeit entwickelt wurden [9].

Das erste Topic Modelling Verfahren wurde 1990 mit der auf *Singular Value Decomposition* basierenden Methode des *Latent Semantic Indexing* (LSI) [20] entwickelt, dessen Modell einigen Restriktionen unterliegt, wie der zwangsläufig eindeutigen Zuordnung von Wörtern zu Topics. Ein erstes probabilistisches Modell wurde 1999 von Hofmann mit dem *Probabilistic Latent Semantic Indexing* entwickelt [32], welches die gemeinsamen Vorkommen von Wörtern in Dokumenten gemäß einer zusammengesetzten, bedingt unabhängigen Multinomialverteilung modelliert. Diese Vorgehensweise hat jedoch eine Tendenz zum Overfitting und ermöglicht keine Schlussfolgerung auf nicht im Trainingsdatensatz betrachtete Dokumente. LSI erfuhr kürzlich eine Erweiterung über die *Non Negative Matrix Factorization* (NMF) [4], welches jedoch nur einen ersten Versuch zur Modellierung von Themen mittels NMF darstellt und eine weitergehende Forschung in dieser Richtung motiviert.

Großer Beliebtheit erfreut sich das generative Verfahren der LDA. Für eine gegebene Anzahl an Topics K, einen Parametervektor $\alpha \in \mathbb{R}^K_{>0}$, ein Vokabular V und eine Matrix $\beta \in [0,1]^{K\times|V|}$, werden die N_d Wörter $\{w_{d1}, \dots w_{dN_d}\} \subseteq V$ des d-ten Dokuments gemäß des folgenden statistischen Prozesses generiert:

1. Wähle einen Themenverteilung $\theta_d \in [0,1]^K$ gemäß der durch α parametrisierten Dirichlet-Verteilung $\mathrm{Dir}(\alpha)$.

2. Wähle für $1 \leq n \leq N_d$ das Wort w_{dn} auf die folgende Art und Weise

   - Ziehe ein Thema $z_{dn} \in \{1, \dots, K\}$ multinomialverteilt gemäß θ_d.
   - Ziehe ein Wort $w_{dn} \in \{1, \dots, |V|\}$ multinomialverteilt gemäß $\beta_{z_{dn}}$.

Die im ersten Schritt angesprochene Dirichlet-Verteilung weist einem Wahrscheinlichkeitsvektor $\theta_d = (\theta_{d1}, \dots, \theta_{dK}) \in [0,1]^K$, $\sum_{k=1}^K \theta_{dk} = 1$ für einen Parametervektor α eine Dichte von

$$p(\theta_d|\alpha) = \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \theta_{dk}^{\alpha_k - 1}$$

zu, wobei Γ die Gammafunktion bezeichnet. Der Vektor θ_d beschreibt die Verhältnisse, in welchen die einzelnen Themen in Dokument d vorkommen. D.h. ein Wort des Dokumentes d wird mit Wahrscheinlichkeit θ_{dk} dem Thema k entstammen. Die Wahl des Themas aus welchem das n-te Wort hervorgeht, wird im 2. Schritt mit der Variable z_{dn} getroffen. Dementsprechend gilt für die Wahrscheinlichkeit mit der Thema k in Dokument d für das n-te Wort gewählt wird

$$p(z_{dn} = k|\theta_d) = \theta_{dk}.$$

Themen definieren Wahrscheinlichkeiten über dem Vokabular, repräsentiert über die Vektoren β_k. So ist die Wahrscheinlichkeit mit der Wort v in Thema k benutzt wird

$$p(w_{dn} = v|z = k) = \beta_{kv}.$$

Es wird angenommen, dass die K themenbezogenen Wortverteilungen β_k für jede Dokumentensammlung Dir(η)-verteilt gezogen werden für einen festgelegten Vektor $\eta \in \mathbb{R}^K_{>0}$. Die resultierende multivariante Verteilung mit der eine Sammlung von Dokumenten D, repräsentiert über die Wörter w_{dn} mit $1 \leq d \leq |D|$ und $1 \leq n \leq N_D$ nach dem beschriebenen Schema generiert wurde, ist gegeben durch

$$p(\theta, \beta, z, w|\alpha, \eta) = \prod_{k=1}^{K} p(\beta_k|\eta) \prod_{d=1}^{|D|} p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} p(z_{dn}|\theta_d) p(w_{dn}|\beta, z_{dn}) \right).$$

Da die einzige Beobachtung die Wörter w_{dn} sind, gilt es die a-posteriori Wahrscheinlichkeit

$$p(\theta, \beta, z|w, \alpha, \eta) = \frac{p(\beta, \theta, z, w|\alpha, \eta)}{p(w|\alpha, \eta)}$$

zu bestimmen, welche aufgrund der marginalisierten Wahrscheinlichkeit im Nenner

$$p(w|\alpha, \eta) = \int \int \sum_z p(\theta, \beta, z, w|\alpha, \eta) d\theta d\beta$$

praktisch nicht zu berechnen ist. Um diese Wahrscheinlichkeit zu approximieren gibt es verschiedene Möglichkeiten. Die hier verwendete Variante ist *Gibbs Sampling* [25]. Gibbs Sampling ist ein einfach zu implementierender Algorithmus, der die unbekannte Verteilung über iterativ alternierend gezogene Stichproben approximiert, welche für jede der Zufallsvariablen bedingt unter den anderen Zufallsvariablen gezogen werden. Die vom Nutzer zu bestimmenden Parameter sind die Anzahl der Themen K, sowie die Parameter $\alpha > 0$ und $\eta > 0$. Zur Vereinfachung wird angenommen, dass die Dirichlet Verteilung nach der die Themenverteilung θ und die Wort-Topic Verhältnisse β gezogen werden, symmetrisch ist und deshalb $\alpha = \alpha_1 = \ldots = \alpha_K$ gilt, ebenso für η. Ist $\alpha < 1$, so werden mit größerer Wahrscheinlichkeit nur wenige Themen für ein Dokument ausgewählt, wohingegen ein großes $\alpha > 0$ die Themenproportionen in Richtung Gleichverteilung verschiebt. Ähnliche Überlegungen gelten für η in Hinsicht auf die Verteilung von β. Steyvers et al. empfehlen einen Wert von $\eta = 0.01$ und $\alpha = 50/K$, d.h. Wörter werden mit höherer Wahrscheinlichkeit eindeutig den Themen zugeordnet und je mehr Themen ausgewählt werden, desto heterogener wird die Zuordnung von Dokumenten zu Themen.

5.3 Topic Extraktion

Unter Benutzung der Software RapidMiner[2] wird eine Themenanalyse auf Ausschnitten von Zeitungsartikeln vorgenommen, welche den Ausdruck `Asyl*` enthalten. Die Abfrage erfolgt mittels des Digitalen Wörterbuchs der deutschen Sprache[3] (DWDS), einem

[2] https://rapidminer.com/
[3] http://www.dwds.de/

Abbildung 5.1: Wordclouds zum Thema *Asylbewerber* (grün, oben) und *Asylant* (rot, unten)

digitalen lexikalischen System, welches u.a. Text- und Zeitungskorpora zur Recherche bereitstellt. Das genutzte Zeitungskorpus ist das umfangreichste vom DWDS bereitgestellte Korpus der Zeitschrift *Die Zeit*, welches Artikel von 1946 bis heute in einem Gesamtumfang von ca. 225 Mio. Textwörtern enthält. Aus diesem Korpus werden 5000 Artikelausschnitte mit dem Ausdruck `Asyl*`, die den Umfang eines Satzes haben, extrahiert. D.h. ein Dokument der LDA entspricht einem Satz aus einem Artikel, welcher die Form `Asyl*` enthält. Dieser Umfang wird gewählt um die Benutzung verschiedener *Asyl*-beschreibenden Wörter in ihrem direkten Kontext und Wortgebrauch unterscheiden zu können. Die Eingabe zur LDA sind die Sätze in lemmatisierter Form, mit entfernten Stoppwörtern, sowie die Anzahl der erwarteten Themen $K = 6$. Diese Anzahl ist heuristisch bestimmt, da sie am besten zu interpretierende Ergebnisse liefert und alle stets präsenten Themen für verschiedene Variationen von K beinhaltet. Die Parameter α und η betragen wie bereits besprochen $\alpha = 8.3$ und $\eta = 0.01$. Die resultierende Themenverteilung β wird mittels Wordclouds visualisiert. Hierzu sind die Top-70 Wörter, die in einem Themenbereich am wahrscheinlichsten vorkommen, mit einer Schriftgröße proportional zu ihrer Wahrscheinlichkeit und einer Farbstärke entsprechend der Spezifität des Ausdrucks zu diesem Thema, d.h. je spezifischer das Wort, desto kräftiger die Farbe, mittels *Wordle*[4] dargestellt. Die Spezifität $\text{spec}(v, \kappa)$ eines Wortes v bzgl. des Themas κ wird hierbei mit der Formel

$$\text{spec}(v, \kappa) = \frac{\beta_{\kappa v}}{\sum_{k=1}^{K} \beta_{kv}}$$

berechnet. Die gefundenen Themen lassen sich unter den Oberbegriffen *Asylbewerber*, *Asylant*, *Flüchtling*, *politisches Asyl*, *Asylbewerberheim* und *Asylrecht* zusammenfassen.

[4] http://www.wordle.net/advanced

5.3.1 Themenbeschreibung

Für alle betrachteten Parametereinstellungen ergibt sich jeweils ein Thema in dem der Begriff *Asylbewerber* und eines in dem der Begriff *Asylant* besonders häufig benutzt wird sowie spezifisch für dieses ist. Abbildung 5.1 zeigt die entsprechenden Wordclouds zu diesen beiden Themen. Zwei weitere Themen in denen die um Asyl bittende Person beschrieben wird, sind die Themen *Flüchtling* und *Politisches Asyl*, wie in Abbildung 5.2 dargestellt.

Es ist zu bemerken, dass dem Thema *Asylant* auch andere vom Duden als unpassend deklarierte Wörter entspringen, wie *Ausländer* oder *Wirtschaftsflüchtling*. Es ist aber keine Tendenz zur Benutzung des Terminus *Asylant* in Hinblick auf brisante Themen wie *Sozialhilfe* oder *Obdachlosigkeit* zu erkennen, Schlagwörter die in der Wordcloud *Asylbewerber* durchaus vorkommen. Um einen Eindruck zu gewinnen, in welchem Kontext diese Wörter nach der vorliegenden Gruppierung genannt werden, sind die Top-3 Textausschnitte, sortiert nach der Wahrscheinlichkeit $\theta_{dAsylbew}$ in Tabelle E.2 dargestellt. Die entsprechenden a-posteriori Wahrscheinlichkeiten sind Tabelle E.1 zu entnehmen. Bei diesen Textabschnitten liegt ein überraschend deutlicher Fokus auf den finanziellen Aspekten zur Aufnahme und Unterstützung von Emigranten. Die drei Textausschnitte in welchen das Thema *Asylant* mit höchster Wahrscheinlichkeit aufgegriffen wird, weisen demgegenüber eine hohe Heterogenität auf (siehe Tabellen E.3 und E.4). Den Texten ist jedoch gemein, dass das Ansehen der Asylanten in der Bevölkerung (Dokument 6) oder von einzelnen Personen (Dokumente 4 & 5) aufgegriffen wird. Die Interpretation der Themen ist jedoch schwerlich genau zu treffen und die vorgeschlagenen Arten der Darlegung stellen nur einen ersten Versuch zur Unterscheidung der beiden Arten in der Ausdrucksform da.

Etwas eindeutiger scheinen die Themen *Flüchtling* und *Politisches Asyl* besetzt zu sein. Spezifische Schlagwörter wie *Asylantrag*, *Grenze*, *ablehnen* und *Gesetz* legen beim Thema *Flüchtling* nahe, dass die Diskussion zur Aufnahme von Flüchtlingen innerhalb Europas, insbesondere in Deutschland, sich besonders mit Asylanträgen und dem geltenden Recht zur Aufenthaltsgenehmigung beschäftigt. Interessant ist die Verschiebung des Vokabulars wenn es um die Diskussion zur Aufnahme von asylsuchenden Individuen wie Edward Snowden geht, welches der Gegenstand des Themas *Politisches Asyl* zu sein scheint. Hier wird von *gewähren*, *bitten*, *beantragen* und *genießen* des Asyls gesprochen.

Schlagwörter des fünften Themas *Asylbewerberheim* wie *Öffentlichkeit*, *jung*, *Anschlag* und *Rostock* scheinen auf eine Darlegung der Gewalt gegenüber Emigranten in Deutschland zu schließen, wie bei dem Vorfall in Rostock-Lichtenhagen wo 1992 ein Asylbewerberheim in Brand gesetzt wurde. Das sechste Thema *Asylrecht* scheint relativ eindeutig die Politik und die momentane Gesetzeslage zu diesem Thema zu beschreiben.

5.3.2 Entwicklung der Themen im Laufe der Zeit

Um Veränderungen des benutzten Vokabulars über der Zeit wahrnehmen zu können, sind die relativen Anzahlen der Dokumente in denen die Themen zur Beschreibung des Asylsuchenden mit höchster Wahrscheinlichkeit vorkommen in Abbildung 5.4 dargestellt. Es

Abbildung 5.2: Wordclouds zum Thema *Flüchtling* (oben, braun) und *politisches Asyl* (unten, blau)

Abbildung 5.3: Wordclouds zum Thema *Asylbewerberheim* (oben, orange) und *Asylrecht* (unten, türkis)

ist eine einheitliche Entwicklung der Verhältnisse in welchen auf die drei Themen eingegangen wird zu erkennen. Die Interpretation der Zeitreihen in Hinblick auf einhergehende Ereignisse erweist sich jedoch als schwierig. Insbesondere ist die hohe Varianz der Themenproportionen in dem Zeitraum bis 1980 irreführend, da zu dieser Spanne weniger als 350 Artikel existieren in welchen die Buchstabenfolge *Asyl* vorkommt. Eine vergleichsweise geringe Anzahl gegenüber mehr als 4600 betrachteten Artikeln in der ähnlich großen Zeitspanne ab 1980. Unter Berücksichtigung dieser Umstände ist keine allgemeine Tendenz zur Favorisierung des einen oder anderen Vokabulars bis heute zu erkennen. Der Inhalt des Artikels scheint eher darüber zu bestimmen ob man von einem Asylbewerber, Asylanten oder Flüchtling spricht. Da diese Inhalte jedoch immer wieder von den Medien aufgegriffen werden, ist auch das entsprechende Vokabular immer wieder präsent.

Zur detaillierteren Betrachtung sind noch einmal die Zeitreihen für alle Themen ab 1980 in Abbildung 5.5 dargestellt. Große Unterschiede in den Erwähnungen der einzelnen Themen ergeben sich nicht und etwaige Peaks in den Zeitreihen sind nur schwer zu erklären. Betrachtet man die Artikel mit hoher Zuordnung zu dem Thema *Asylbewerberheim* zwischen 2001 und 2008, so finden sich unter den ersten 10 Dokumenten Thematisierungen der Vorfälle in Rostock-Lichtenhagen 1992, des Falls von Oury Jalloh, der in Dessau 2005 durch einen Brand in Polizeigewahrsam umkam und der Taten im brandenburgischen Dolgenbrodt, wo 1992 junge Männer ein Asylbewerberheim anzündeten. Eine Reflexion dieser Daten in der zugehörigen Zeitreihe ist jedoch nicht zu erkennen. 1992 ist das Thema *Asylbewerberheim* eher geringfügig vertreten und 2005 gibt es kein einziges Dokument, welches diesem Thema zugeordnet wird.

Ähnlich verhält es sich für die anderen Zeitreihen. Das Thema *politisches Asyl* erfährt keine besonders hohe Zunahme im Jahr 2013 in welchem Edward Snowden PRISM offenlegte. In den Dokumenten über *Asylrecht* wird ab 1980 besonders häufig auf den Petersberger Asylkompromiss von 1992 eingegangen, eine Gegebenheit die sich in der Zeitreihe kaum widerspiegelt. Die Dokumente zum Thema *Flüchtling* lassen kein spezifisches Ereignis im Jahr 2002 erkennen wohingegen die Zeitreihe dort einen aussergewöhnlichen Peak hat. Die eher heterogene Dokumentensammlung zum Thema *Asylant* weist keinen spezifischen Inhalt in der Zeitspanne von 2004 bis 2006 auf und der Verlauf des Themas *Asylbewerber* hat generell keine besonderen Ausprägungen.

Es ist anzumerken, dass die betrachtete Kontextgröße von einem Satz ggf. zu klein ist um verständliche Resultate für eine Betrachtung der Zuordnung von Artikeln zu Themen in Abhängigkeit von der Zeit zu erhalten, da in Nebensätzen erwähnte Gegebenheiten aus der Vergangenheit zu einem anderen Datum ins Gewicht fallen. Eine umfangreichere Analyse der Zeitreihen unter Berücksichtigung verschiedener Kontextgrößen und den demnach entstehenden Themen, wäre hier von weitergehendem Interesse.

5.4 Schlussfolgerungen

In der vorliegenden Arbeit wurde untersucht, inwiefern das Verfahren der LDA eine Darlegung über Verschiebungen des genutzten Vokabulars in Hinblick auf das prekäre Thema *Asyl* möglich macht. Die Analyse erfolgte exemplarisch auf Grundlage von Artikelaus-

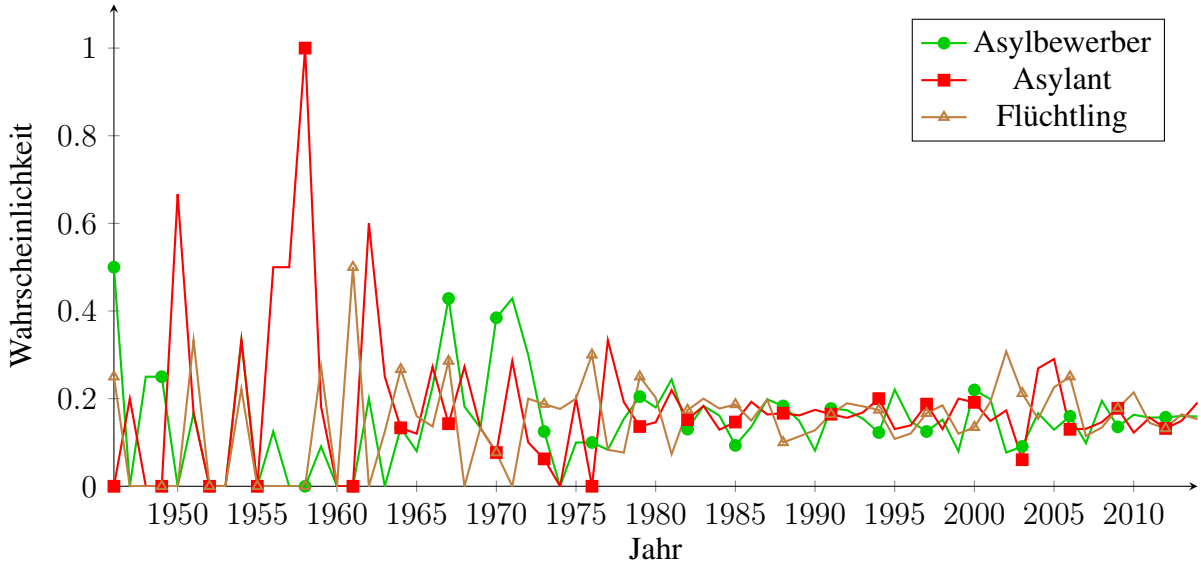


Abbildung 5.4: Darstellung der Thematisierung des Asylsuchenden in Artikeln der Zeitschrift *Die Zeit* seit 1946. Die auf der y-Achse aufgetragenen Werte beschreiben die Proportionen der Dokumentenabschnitte, die den Ausdruck `Asyl*` enthalten und sich auf das jeweilige Thema beziehen.

schnitten der Zeitung *Die Zeit*, welche die Buchstabenfolge *Asyl* enthalten. Die Themen wurden anhand von Wordclouds visualisiert und ihre Präsenz im Laufe der Zeit dargestellt. Anhand dieser Visualisierung wurde festgestellt, dass der als politisch inkorrekt empfundene Begriff des *Asylanten* den politisch korrekten Bezeichnungen *Asylbewerber* und *Flüchtling* je nach Kontext vorgezogen wird. Dieser Kontext muss jedoch nicht notwendigerweise ein negativer sein. Dementsprechend scheint das Wort *Asylant* keiner stark negativen Bedeutungsnuance zu unterliegen und hat nicht zwingendermaßen einen denunzierenden Kontext.

Die Darstellung der Themen-Proportionen im Laufe der Zeit konnte keine weiteren Erkenntnisse liefern. Es gab nur wenige ausschlaggebende Charakteristika der Zeitkurven und diese konnten nicht mit Ereignissen in Verbindung gebracht werden. Hier wäre eine genauere Untersuchung von Nöten.

Des Weiteren sind die vorliegenden Schlussfolgerungen unter dem Aspekt zu betrachten, dass hier nur ein Zeitungskorpus von vielen zugrunde lag. Eine Analyse die sich auf verschiedene Zeitungskorpora, insbesondere Boulevardzeitungen stützt, würde womöglich ein umfassenderes Bild in Bezug auf Events liefern. Interessant wäre zudem eine Analyse der vorliegenden Begrifflichkeiten und ihrem Gebrauch in Kommentarspalten von Zeitungen um einen bildungssprachlichen Umgang mit den vorliegenden Themen erfassen zu können.

Zusammenfassend ist zu bemerken, dass das Verfahren der LDA geeignet war um Unterschiede im Vokabular und Kontext ähnlicher Begriffe wie *Asylant* und *Asylbewerber* herauszustellen. In der langfristigen Themenbetrachtung ergaben sich jedoch Deutungsschwierigkeiten, dessen Ursprung einer näheren Analyse bedürfen.

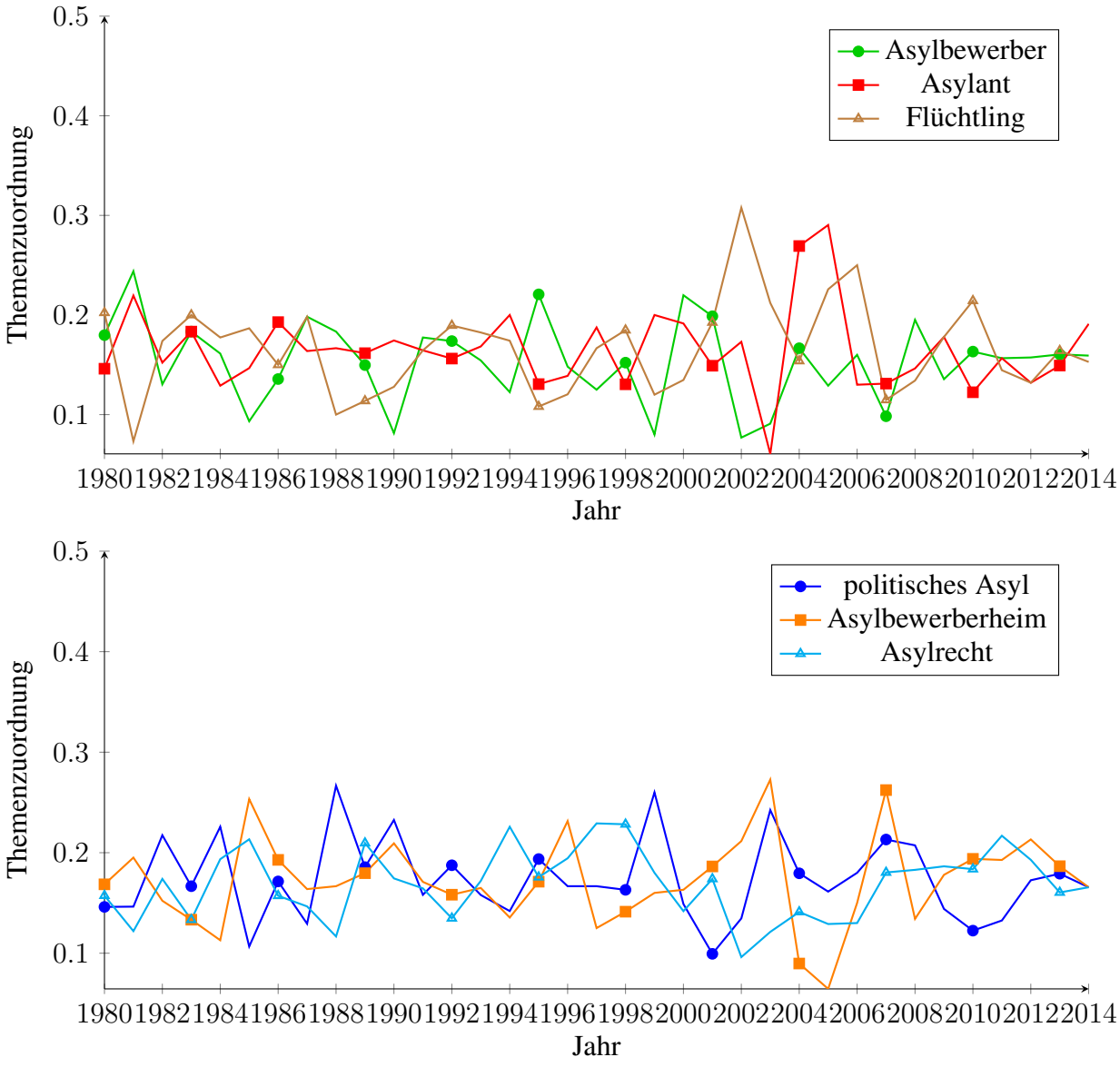


Abbildung 5.5: Entwicklung der Thematisierung des Asylsuchenden (oben) und der anderen Asyl-bezogenen Themen (unten) in Artikeln der Zeitung *Die Zeit* ab 1980. Die auf der y-Achse aufgetragenen Werte beschreiben die Proportionen der Dokumentenabschnitte, die den Ausdruck `Asyl*` enthalten und sich auf das jeweilige Thema beziehen.

5.5 Dokumentenausschnitte

d	$\theta_{dAsylbew}$	$\theta_{dAsylant}$	$\theta_{dFlucht}$	$\theta_{dPolitisch}$	θ_{dHeim}	θ_{dRecht}
1	0.47635	0.09346	0.09346	0.13216	0.16401	0.13216
2	0.31327	0.14351	0.12808	0.17224	0.13879	0.22240
3	0.27811	0.15653	0.12613	0.11802	0.19416	0.11802

Tabelle E.1: A-posteriori Wahrscheinlichkeiten in denen die Themen zu den Textausschnitten in Tabelle E.2 ausgewählt wurden.

d	Datum	Textausschnitt d
1	1996-11-04	Vor allem, wenn man an Kühns Harburger Parteifreund denkt, Verteidigungsminister Volker Rühe, wie der einst, als Generalsekretär seiner Partei, konspirative Briefe verschickt hat, in denen er den Vorsitzenden aller CDU-Fraktionen in Stadt und Land Anweisung gab, einmal ganz spontan danach zu fragen, wie lange sich Asylbewerber durchschnittlich in der jeweiligen Gemeinde aufhalten oder 'wie hoch die monatlichen Kosten für Unterbringung und Versorgung der Asylbewerber sind' oder 'ob Fälle bekannt geworden sind, in denen Asylbewerber staatliche Leistungen unberechtigterweise mehrfach in Anspruch genommen haben'.
2	1989-10-06	Zusätzlich erhalten Aussiedler und anerkannte Asylberechtigte während dieser Zeit anstelle von Sozial- oder Arbeitslosenhilfe ein Unterhaltsgeld: Aussiedler 352 Mark pro Woche, Asylberechtigte 324 Mark.
3	2012-10-13	Sie verlangten eine Erhöhung der staatlichen Hilfen für Asylbewerber auf das Niveau der Sozialhilfe und von Hartz IV. 'Das wird dazu führen, dass die Asylbewerber-Zahlen noch weiter steigen, denn es wird für Wirtschaftsflüchtlinge noch attraktiver zu uns zu kommen und mit Bargeld wieder abzureisen', sagte Friedrich.

Tabelle E.2: Exemplarische Auswahl von drei Textausschnitten in denen das Thema *Asylbewerber* eine hohe Proportion annimmt.

d	$\theta_{dAsylbew}$	$\theta_{dAsylant}$	$\theta_{dFlucht}$	$\theta_{dPolitisch}$	θ_{dHeim}	θ_{dRecht}
4	0.11096	0.35160	0.12433	0.15106	0.13770	0.12433
5	0.13411	0.34244	0.12109	0.13411	0.13411	0.13411
6	0.18355	0.34121	0.11599	0.11599	0.10472	0.13851

Tabelle E.3: A-posteriori Wahrscheinlichkeiten in denen die Themen zu den Textausschnitten in Tabelle E.4 ausgewählt wurden.

d	Datum	Textausschnitt d
4	1966-04-01	Als das deutsche Blut ganz laut und ganz mörderisch geschützt wurde, fiel mir plötzlich auf: in unserem kleinen Asyl in Sanary sur mer tauchten immer mehr deutsch-jüdische Mischehen auf: Toller und Christiane, der Verleger Landshoff und Reni, Meier-Graefe und seine jüdische Frau... unser Asyl war voll von Rassenschande..
5	1992-11-06	Das wär' ja noch schöner, hodagsaggt, daß wir da mitmarschieren, hodagsaggt, da droben in Berlin, in diesem Kreuzberg, hodagsaggt, bei dieser Schaufensterveranstaltung, hodagsaggt, für die ganzen Asylanten, hodagsaggt, die uns das Geld wegnehmen, hodagsaggt, das wir für unsere Kinder zum Beispiel brauchen, hodagsaggt, und nicht für eine multikriminelle Gesellschaft, hodagsaggt, und Wurscht ist es ihm sowieso, hodagsaggt, wer ihn lobt dafür, hodagsaggt.
6	2009-06-08	Unter anderem ergab dieser Bericht, dass Asylwerber in Österreich nicht ordnungsgemäß über ihre Rechte unterrichtet werden, dass die Unterstützung zur Deckung eines minimalen Lebensunterhaltes nicht ausreicht und oft unrechtmäßig entzogen wird, dass Minderjährigen, die in Gewahrsam genommen wurden, der Zugang zum Bildungssystem verwehrt wird, dass die Arbeitsmöglichkeiten der Asylwerber sehr stark beschränkt werden, dass geltendes Recht in Transitzonen nicht angewandt wird, dass bei der Einschränkung der Bewegungsfreiheit kein Rechtsmittel möglich ist oder dass Bestimmungen für »besonders schutzwürdige Personen« völlig unbeachtet bleiben.

Tabelle E.4: Exemplarische Auswahl von drei Textausschnitten in denen das Thema *Asylant* eine hohe Proportion annimmt.

Kapitel 6

Untersuchung zur Anwendbarkeit von automatisch berechneten Worteinbettungen in der deutschen Sprache

von Sebastian Buschjäger

6.1 Einleitung

Die Darstellung von natürlicher Sprache im Computer bildet die essentielle Grundlage für alle natürlichsprachlichen Systeme und hat damit kritischen Einfuss auf die Performanz eines Systems. Eine gängige Kodierung von Texten bildet das Bag-Of-Words (BOW) Modell [35], in welchem Texte als Binärvektoren dargestellt werden, die für jedes Wort der Sprache einen Eintrag haben. Entsprechend dem Vorkommen eines Wortes im Text wird der passende Eintrag im Vektor entweder auf Null oder auf Eins gesetzt. Alternativ können anstelle dieser Binärwerte auch absolute Termhäufigkeiten bzw. Termhäufigkeit/inverse Dokumentenhäufigkeit (TF-IDF)-Werte genutzt werden [65].

Das klassische Bag-Of-Words Modell erzeugt hochdimensionale, dünn besetzte Vektoren. Diese eignen sich gut für Klassifikationsaufgaben, in welchen einzelne Wörter ohne Kontext bereits Hinweise auf die Klasse liefern [35]. Unglücklicherweise gehen Information über die Reihenfolge der Wörter in einem Text im BOW Modell verloren, sodass sehr ähnliche Vektoren für vollständig unterschiedliche Sätze entstehen können. Dieser Effekt wird insbesondere im Bereich der Sentimentanalyse und des *Opinion Mining* deutlich, wenn bereits kleinste Änderungen in einem Satz die zugehörige Klasse ändern [62, 81, 57]. Aus diesem Grund ist es wünschenswert, eine Wortrepräsentation zu finden, welche bereits zu einem gewissen Grad semantische Relationen abbilden kann.

Mikolov et al. haben vor Kurzem ein Verfahren zur effizienten Berechnung von niedrig dimensionalen Wortvektoren vorgestellt. Sie konnten zeigen, dass die durch ihr Verfahren berechneten Wortvektoren gewisse syntaktische und semantische Ähnlichkeiten aus dem

Trainingskorpus übernehmen, so dass die so entstehenden Vektoren sich z.B. für die Sentimentanalyse besser eignen als klassische BOW Vektoren [50, 51]. Mikolov et. al. nennen ihre Methode Word2vec und haben unter [49] eine Implementierung öffentlich zugänglich gemacht.

Die syntaktischen und semantischen Beziehungen von Word2vec sind bisher nur für die englische Sprache untersucht worden. Es gibt einige Hinweise in der Literatur, dass die deutsche Sprache im Vergleich zur englischen Sprache durchaus andere Anforderungen an ein natürlichsprachliches System stellt. Allen voran scheint die Wortstellung im deutschen zweitrangig gegenüber dem Englischen zu sein, da Bezüge mittels Kasus besser hergestellt werden können [44]. Dieses Problem der freien Wortstellung bereitet maschinellen Lernverfahren üblicherweise Schwierigkeiten, da Bezüge zwischen den Wörter weit über den Satz verteilt sein können [41, 40].

Die vorliegende Arbeit vergleicht den Einfluss des verwendeten Trainingskorpus auf die berechneten Word2vec Vektoren und untersucht, inwiefern sich bei unterschiedlichen Textkorpora syntaktische und semantische Relationen in den Vektoren ergeben. Insbesondere wird ein Vergleich zwischen einem deutschen und einem englischen Textkorpus durchgeführt. Hierzu werden zunächst Wortvektoren mittels Word2vec auf Basis der englischen und deutschen Wikipedia trainiert. Anschließend wird der in [50] vorgestellte Testdatensatz ins Deutsche übersetzt und die dort vorgestellten Experimente wiederholt. Zusätzlich wird eine Untersuchung zwischen der Ähnlichkeiten von Wortvektoren und ihren Synonymen vorgenommen um weitere Problemeinsicht zu gewinnen.

Diese Arbeit ist wie folgt strukturiert: Abschnitt 6.2 stellt das Word2vec Modell kurz vor. Abschnitt 6.3 beschreibt die durchgeführten Experimente, d.h. das Training der Wortvektoren und die Aufbereitung der Testdaten im Detail. In Abschnitt 6.4 sind schließlich die Ergebnisse dieser Experimente zu finden. Abschnitt 6.5 versucht diese Ergebnisse zu interpretieren und mit dem Wissenstand aktueller Literatur zu vergleichen. Schließlich fasst der letzte Abschnitt 6.6 die Ergebnisse dieser Arbeit zusammen.

6.2 Das Word2vec Modell

Word2Vec besteht im Wesentlichen aus einem Netz mit drei Schichten, welches trainiert wird, um eine Wahrscheinlichkeitsverteilung über Wörtern zu finden. Hierbei ergibt sich implizit eine Einbettung der Wörter des Vokabulars $\mathcal{V}$ in den euklidischen Raum $\mathbb{R}^D$. Hierzu definieren die Autoren in [50] zwei einfache strukturierte neuronale Netze, die effizient trainierbar sind. Anschließend werden zwei Lernaufgaben für diese Netze definiert. Bei der ersten Lernaufgabe wird versucht ein Wort anhand seines Kontext vorherzusagen (vgl. Abbildung 6.1 (a)). Einzelne Wörter werden als 1-of-V Kodierung kodiert, wobei V hier der Anzahl der Wörter im Vokabular $\mathcal{V}$ entspricht. Die Eingabeschicht bei einem Kontext der Größe N hat also insgesamt $V \cdot N$ Eingabeneuronen. Die lineare Projektionsschicht U ist für alle Eingabeneuronen dieselbe und bildet die N verschie-

denen V-dimensionalen Eingabevektoren auf einen D-dimensionalen Vektor ab. Durch die Linearität der Projektionsschicht ergibt sich der resultierende Wortvektor also im Wesentlichen aus einer gewichteten Summe der umliegenden Wörter $w_1, \ldots, w_N$:

$$\tilde{w} = \sum_{i=1}^{N} U \cdot w_i \tag{6.1}$$

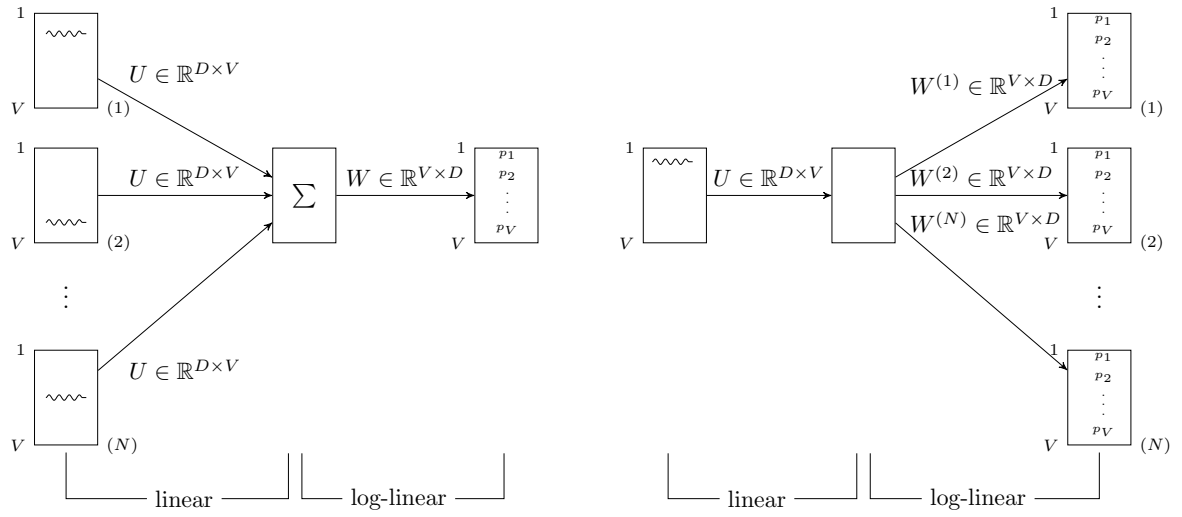
Bezeichne $W_{j\bullet}$ die j-te Zeile der Matrix W, so kann mit Hilfe des „Eingabevektors“ $\tilde{w}$ die Wahrscheinlichkeit für jedes Wort w_j des Vokabulars mit einer Softmax-Funktion geschätzt werden:

$$p(w_j|w_1, \ldots, w_N) = \frac{e^{W_{j\bullet}^T \cdot \tilde{w}}}{\sum_{i=1}^{V} e^{W_{j\bullet}^T \cdot \tilde{w}}} \tag{6.2}$$

In diesem Modell müssen lediglich die Matrizen U und W bestimmt werden, sodass sich eine Trainingskomplexität von $Q = N \cdot D + D \cdot V$ ergibt. Die Autoren nennen dieses Modell Continous Bag-Of-Words (CBOW), da für den gewichteten Durchschnitt die Reihenfolge der Wörter unerheblich ist und sich der Kontext eines Wortes nicht nur auf vergangene, sondern auch zukünftige Wörter im Satz bezieht.
Die zweite Lernaufgabe ist so formuliert, dass das neuronale Netz den Kontext, d.h. die umliegenden Wörter eines Wortes vorhersagt (vgl. Abbildung 6.1 (b)) und bildet damit sozusagen das Gegenstück zum CBOW Modell. Auch in diesem Modell werden Wörter zunächst mit einer 1-of-V Kodierung abgebildet . Ein einzelnes Wort wird linear in der Projektionsschicht U zu einem D-dimensionalen Vektor transformiert. Dieser wird anschließend auf N log-lineare Klassifizierer $W^{(1)}, \ldots, W^{(N)}$ angewandt (vgl. Formel 6.1 und Formel 6.2). In diesem Modell müssen die Matrizen $U, W^{(1)}, \ldots, W^{(N)}$ bestimmt werden, sodass sich eine Trainingskomplexität von $Q = N \cdot (D + D \cdot V)$ ergibt. Die Autoren nennen dieses Modell (Continous) Skip-Gram Modell.
Alle Modelle werden mit Hilfe des stochastischen Gradientenverfahrens [13] und Backpropagation [67] trainiert. Die Größe des Vokabulars ist üblicherweise zwischen $10^5 - 10^7$, sodass diese die Berechnungskomplexität dominiert. Um dieses Problem in den Griff zu bekommen, führen Mikolov et. al zwei Verbesserungen ein: Zum Einen kann mittels einer Huffmann-Kodierung die Vokabulargröße von V auf $log_2(V)$ verringert werden wodurch sich eine hierachische Softmax-Funktion ergibt [55]; Andererseits ergibt sich mittels Noise Contrastive Estimation, von Mikolov et. al. Negative Sampling genannt, eine Möglichkeit zur direkten Approximation von Formel 6.2 [27, 53].

6.3 Experimenteller Aufbau

Word2Vec trainiert ein neuronales Netz um eine Wortverteilung für einen gegebenen Textkorpus zu finden. Hierbei wird zugunsten eines schnelleren Trainings ein vergleichsweise einfaches Modell angenommen, um so die Verarbeitung großer Datenmengen zu erlauben. Das neuronale Netz führt eine Transformation von hochdimensionalen, dünnbesetzten 1-of-V Vektoren $w \in \mathcal{V}$ zu niedrig dimensionalen, dichten Wortvektoren $\tilde{w} \in \mathbb{R}^D$ durch. Diese implizite Abbildung $u \colon \mathcal{V} \to \mathbb{R}^D$ scheint die Semantik der Wörter im gewissen Sinne zu erhalten, sodass ähnlichkeitserhaltende Vektoroperationen ermöglicht werden, z.B:



(a) CBOW Modell

(b) Skip-Gram Modell

Abbildung 6.1: Schematische Darstellung von Word2vec.

$u(Paris) - u(France) + u(Italy) \approx u(Rome)$.
Um diese semantischen Beziehungen weiter zu untersuchen, definieren die Autoren in [51] Testfragen der Art „w_1 verhält sich zu w_2, wie w_3 zu w_4“. Eine Frage gilt genau dann als korrekt beantwortet, wenn gilt:

$$w_4 = \underset{w \in \mathcal{V}}{\operatorname{argmin}} \{ \| u(w_2) - u(w_1) + u(w_3) - u(w) \|_2^2 \} \tag{6.3}$$

Einen Testdatensatz mit knapp 20000 vordefinierten Wortbeziehungen in insgesamt 14 verschiedenen Kategorien (vgl. Tabelle D.1) und eine Implementierung dieses Ähnlichkeitstests ist unter [49] zu finden.
Mikolov et. al. trainieren in [51] das neuronale Netz mit einem Google-News Korpus der insgesamt 6 Milliarden Wörtern beinhaltet. Sie konnten damit sehr gute Ergebnisse im Semantiktest erzielen. Unglücklicherweise ist der verwendete Google-News Korpus nicht verfügbar, jedoch haben die Autoren vorher trainierte Wortvektoren veröffentlicht [49]. Diese Vektoren wurden auf 100 Milliarden Wörtern des Google-News Textkorpus mit Skip-Gram, Negative Sampling und $D = 300$ trainiert. Diese vortrainierten Vektoren umfassen ein Vokabular von ca. 3 Millionen Wörtern und dienen als Referenzvektoren.
Mit Hilfe von [54] erfolgt eine teilautomatisierte Übersetzung des Testdatensatzes ins Deutsche. Da Word2vec nur einzelne Wörter abbilden kann, sind alle längeren Phrasen nach Möglichkeit durch Synonyme ersetzt oder entfernt worden. Dies erwies sich gerade für die vierte Grammatikkategorie (vgl. Tabelle D.1) als schwierig, da im Deutschen der Superlativ für Adjektive mit dem Wort „am“ gebildet wird und sich so immer zusammengesetze Phrasen ergeben. Aus diesem Grund wird auf die Evaluation dieser Kategorie im Deutschen verzichtet, sodass der deutsche Testdatensatz insgesamt knapp 18000 vordefinierte Beziehungen enthält.
Die Grundlage des deutschen Trainingsdatensatzes bilden die Artikel der deutschen Wikipedia vom 19.12.2014 [84]. Zur Extraktion der Artikel aus den XML Archiven ist unter

[5] ein Skript zu finden, welches für Word2vec leicht angepasst wurde: Die Implementierung von Mikolov et. al. unterscheidet nicht zwischen Groß- und Kleinschreibung und kann lediglich mit ASCII konformen Zeichen umgehen. Daher werden alle Wörter der extrahierten Artikel zunächst in Kleinbuchstaben transformiert und anschließend Umlaute, sowie ß in eine entsprechende ASCII Darstellung umgewandelt. Kommata und Tabs werden in Leerzeichen umgewandelt; Punkte werden durch Zeilenumbrüchen ersetzt. Alle weiteren nicht-ASCII konformen Zeichen werden gelöscht. Dieser so entstandene Datensatz enthält insgesamt ca. 591 Millionen Wörtern aufgeteilt in 43 Millionen Zeilen und hat ein Vokabular von knapp $1,6$ Millionen Wörtern. Mit Hilfe dieses Datensatzes werden Vektoren nach dem Skip-Gram und nach dem CBOW Modell trainiert. Die Anzahl der Zeilen gibt mit Hilfe der Wortanzahl eine grobe Abschätzung der duchschnittlichen Satzlänge, sodass für den Kontext $N = 13$ gewählt wird. Um Vergleichbarkeit zu gewährleisten wird wie für die Referenzvektoren $D = 300$ gewählt. Für die Anzahl der Trainingsepochen von Backpropagation wurde wie in der Implementierung unter [49] 15 gewählt.

Damit die Anwendbarkeit von Word2vec für die deutsche Sprache untersucht werden kann, sollten sich die zugrundeliegende Trainingskopora möglichst ähnlich sein. Aus diesem Grund wird die englische Wikipedia vom 08.02.2015 [85] als Trainingsdatensatz für die englischen Wortvektoren genutzt. Nach der bereits beschriebenen Aufbereitung der Wikipedia-Daten entsteht so ein Datensatz mit insgesamt ca. $1,6$ Milliarden Wörtern und einem Vokabular von knapp $1,7$ Millionen Wörtern. Für diesen Datensatz wird mit Hilfe von Negative Sampling ein neuronales Netz für CBOW mit $N = 13$ und $D = 300$ trainiert[1]. Auch hier werden 15 Trainingsepochen durchgeführt.
Da das Vokabular mit $1,6$ bzw $1,7$ Millionen Wörtern in allen Experimenten sehr groß ist, ist die Auswertung von Formeln 6.3 kaum für das Gesamtvokabular möglich. Daher wurde Formel 6.3 wie in [50] auf einem zufälligen Ausschnitt von 200000 Einträgen ausgewertet.

Word2vec nutzt Regelmäßigkeiten im Textkorpus aus, um ähnliche Wörter auf ähnliche Vektoren abzubilden. Daher wird zusätzlich untersucht, inwiefern sich die Wortvektoren von Substantiven zu den Wortvektoren ihrer Synonyme ähneln. Hierzu wurde je eine Liste von 1000 Substantiven für die deutsche und englische Sprache zusammengestellt. Für die englische Sprache bietet OpenOffice unter [24] ein Synonymwörterbuch; für die deutsche Sprache ist der OpenThesaurus unter [59] herunterladbar. Für jedes Substantiv w wurde die durchschnittliche Kosinusähnlichkeit zu seinen Synonymen $s_1, \ldots, s_m$ berechnet:

$$\frac{1}{m}\sum_{i=1}^{m} cos(\angle(w, s_i)) = \frac{1}{m}\sum_{i=1}^{m} \frac{w^T \cdot s_i}{\|w\|_2 \cdot \|s_i\|_2} \tag{6.4}$$

Da Word2vec nur Wörter kennt die es zuvor gesehen hat, werden umgangssprachliche Synonyme, sowie nicht im Vokabular von Word2vec enthaltende Wörter ignoriert. Um die Kosinusähnlichkeit in Relation setzen zu können, wurde für jede Sprache zusätzlich die Ähnlichkeit von zufällig gewählten Substantiven zueinander bestimmt.

[1] Das Training für Wortvektoren nach dem Skip-Gram Modell für diesen Datensatz hätte etwas über einer Woche gedauert.

6.4 Ergebnisse

Tabelle D.1 zeigt die Genauigkeit des semantischen Wortbeziehungstests für die 14 verschiedenen Kategorien in Prozent. Als Referenz dient die Performanz der Google-News Wortvektoren, in Klammern ist jeweils der Unterschied zu dieser Referenz angegeben. Die letzte Zeile gibt die durchschnittliche Differenz eines Modells zur Referenz an.
Zunächst zeigt sich, dass Wortbeziehungen zwischen Hauptstädten und ihren Ländern sowohl im Deutschen als auch im Englisch mit bis zu 9.88% höherer Genauigkeit erkannt werden können, falls die Wikipedia als Trainingskorpus genutzt wird.
Ansonsten ergibt sich für die englischen Wortvektoren ein eher gemischtes Bild. Im Falle der vierten Grammatikkategorie ist eine Verschlechterung von 30.7% zu erkennen. Des Weiteren kommt es zu sowohl moderaten Verschlechterungen als auch zu moderaten Verbesserungen in den übrigen Kategorien. Alles in Allem schneiden die mit der englischen Wikipedia trainierten Wortvektoren um fast 5% schlechter ab als die mit dem Google-News Korpus trainierten Vektoren.
Im Falle der deutschen Vektoren ergibt sich ein deutlicheres Bild. Abgesehen von den ersten beiden Kategorien scheitern die Wortvektoren für das Deutsche in den übrigen Kategorien. Gerade der Vergleich von Partizipien im Deutschen ist mit einer Genauigkeit von lediglich 3.59% bzw 6.54% um fast 75% schlechter als im Englischen. Des Weiteren sind Adjektivformen von Nationlitäten um knapp 45% bzw. knapp 48% schlechter. Alle übrigen Kategorien verschlechtern ihre Genauigkeit im Deutschen gegenenüber dem Englischen um $2.7\% - 41\%$. Sowohl das CBOW Modell, als auch das Skip-Gram Modell liefern im Deutschen unterschiedliche Genauigkeiten in verschiedenen Kategorien, sind jedoch beide im Durchschnitt knapp 25% schlechter als die Referenz.
Tabelle D.2 zeigt die Kosinusähnlichkeit für den Synonym- Test. Das englische Synonymwörterbuch enthält durchschnittlich 13 Synonyme pro Wort, der deutsche OpenThesaurus enthält durchschnittlich 12 Synonyme pro Wort, sodass beide Wörterbücher annähernd vergleichbar sind. Zunächst zeigt sich, dass in beiden Sprachen für alle Trainingsdatensätze die Kosinusdistanz zufälliger Wörter erwartungsgemäß annähernd bei Null liegt. Ansonsten liegt die durchschnittliche Kosinusdistanz für die englische Sprache unabhängig vom gewählten Modell und Trainingskorpus bei knapp 0.25. Für die deutsche Sprache zeigt sich eine ähnliche Modellunabhängigkeit. Hier liegt die durchschnittliche Kosinusdistanz jeweils bei 0.56.

6.5 Evaluierung

Die Datenlage zeigt sich zunächst uneindeutig. Einerseits scheitern die Wortvektoren im Deutschen in fast allen Kategorien des Semantik-Tests, andererseits scheinen sie die Beziehungen zwischen Synonymen deutlich besser abzubilden.
Eine erste Erklärung liefern die zugrundeliegenden Datensätze. Es wird allgemein davon ausgegangen, dass die Güte der Wortvektoren direkt mit der Größe des Trainingsdatensatzes korreliert [50]. Der Trainingskorpus für die Google-News Vektoren enthielt knapp 6 Milliarden Wörtern bei einem Vokabular von ca. 3 Millionen Wörtern, sodass das neuronale Netz jedes Wort im Schnitt 2000 mal pro Epoche behandeln konnte. Die englische

Kategorie	Referenz	Englisch	Deutsch	
		CBOW	Skip-Gram	CBOW
capital-common-countries	81,60	86,96 (+5,36)	**93,48** (+9,88)	91,70 (+8,1)
capital-world	83,30	**91,29** (+7,99)	82,55 (+1,25)	84,88 (+1,58)
currency	30,95	19,06 (-11,89)	**11,55** (-19,4)	15,54 (-15,41)
city-in-state	72,15	66,84 (-5,31)	44,08 (-28,07)	**43,09** (-29,06)
family	85,18	**87,94** (+2,76)	52,96 (-32,22)	**50,99** (-34,19)
gram1-adjective-to-adverb	29,13	**31,55** (+2,42)	11,39 (-17,74)	**11,28** (-17,85)
gram2-opposite	44,97	29,43 (-15,54)	**15,61** (-29,36)	15,81 (-29,16)
gram3-comparative	91,14	86,71 (-4,43)	**57,81** (-33,33)	60,32 (-30,82)
gram4-superlative	87,61	56,91 (-30,7)	- -	- -
gram5-present-participle	78,22	64,58 (-13,64)	**3,59** (-74,63)	6,54 (-71,68)
gram6-nationality-adjective	92,37	88,81 (-3,56)	46,52 (-45,85)	**43,87** (-48,5)
gram7-past-tense	64,49	**65,26** (+0,8)	**42,11** (-22,38)	42,17 (-22,32)
gram8-plural	86,64	**84,01** (+2,63)	**45,16** (-41,48)	48,02 (-38,62)
gram9-plural-verbs	67,93	**62,07** (-5,86)	62,83 (-5,1)	65,15 (-2,78)
Durchschnittliche Differenz zur Referenz	-	-4,93	-26,03	-25.44

Tabelle D.1: Anzahl der korrekt beantworteten Testfragen in Prozent für die verschiedenen Textkorpora und Modelle. In Klammern ist jeweils der relative Unterschied zur Referenz angegeben.

	Referenz	Englisch	Deutsch	
		CBOW	Skip-Gram	CBOW
Synonyme	0,25	0,26	0,56	0,56
Zufällige Substantive	0,08	0,04	0,06	0.05

Tabelle D.2: Duchschnittliche Kosinusähnlichkeit für Synonyme im Deutschen und Englischen.

Wikipedia enthält $1,6$ Milliarden Wörter mit einem Vokabular von ca. $1,7$ Millionen Wörtern. Hier konnte das neuronale Netz die Darstellung jedes Wortes ca. 940 mal pro Epoche anpassen. Diese Diskrepanz erklärt zunächst das um etwa 5% schlechtere Abschneiden der englischen Wikipedia-Vektoren im Vergleich zu den Google-News Vektoren und stützt damit die generelle Aussage von Mikolov. et. al. in [50].
Die deutsche Wikipedia enthält lediglich 591 Millionen Wörter bei einem Vokabular von ca. $1,6$ Millionen Wörtern, sodass hier das neuronale Netz jedes Wort ungefähr 370 mal pro Epoche behandeln konnte. Eine höhere Genauigkeit der Google-News Wortvektoren war aufgrund des 200 mal größeren Trainingskorpus demnach zu erwarten, überrascht jedoch in ihrer Deutlichkeit von teilweise bis zu 75% Unterschied in der Klassifikationsgüte. Andererseits scheinen die deutschen Wortvektoren Synonymähnlichkeiten deutlich besser abzubilden, als dies die englischen Vektoren tun, sodass die strukturellen Unterschiede beider Sprache offenkundig einen deutlich Einfluss auf das Word2vec Modell haben. Es fällt auf, dass die deutsche Wikipedia ein annähernd so großes Vokabular wie die englische Wikipedia hat, obwohl die englische Wikipedia fast drei mal so groß ist wie die Deutsche. Hierfür gibt es zwei Erklärungen: Zum Einen führt Word2vec keinerlei Vorverarbeitung oder Ähnliches durch, sodass Wörter je nach Kasus, bzw. Konjugation auf unterschiedliche Wortvektoren abgebildet werden. Zum Anderen ist es allgemein bekannt, dass die deutsche Sprache ein größeres Vokabular hat als die englische Sprache (vgl. [37] und [60]).

Anscheinend ist es kritisch für die Performanz des neuronalen Netzes, wie häufig es ein Wort pro Epoche bearbeiten kann. Ist das Vokabular im Vergleich zum Trainingskorpus klein, wie dies bei den Google-News Vektoren der Fall ist, so kann das neuronale Netz die Darstellung der Wörter pro Epoche häufiger anpassen, was zu besseren Ergebnissen führt. Im Falle der deutschen Wikipedia ist dieses Verhältnis von Wörtern im Trainingskorpus zur Vokabulargröße jedoch zugunsten des Vokabulars verschoben, sodass es zu schlechteren Ergebnissen kommt.
Des Weiteren kommt es im Deutschen weiterhin zum Problem der freien Wortstellung. Um dieses Problem annähernd in den Griff zu bekommen, wurde der Kontext eines jeden Wortes mit $N = 13$ auf die durchschnittliche Satzlänge gesetzt, sodass dieses Problem weniger Einfluss haben sollten. Dennoch passt das Word2vec Modell schlichtweg nicht perfekt zur Syntax der deutschen Sprache: Im Englischen werden Objektbeziehungen vor Allem durch die Satzstellung deutlich, wohingegen diese im Deutschen fast beliebig ist und Objektbeziehungen durch den Kasus klar gemacht werden. Word2vec bildet diejenigen Wörter auf ähnliche Vektoren ab, die an ähnlichen Stelle und im Satz und damit

in ähnlichen Kontexten vorkommen. Diese Annahme erweist sich im Englischen als passend, da hier ähnliche Wörter in ähnlichen Kontexten immer an einer ähnlichen Stelle im Satz stehen. Gerade die Wortreihenfolge in semantisch ähnlichen Sätzen ist immer annähernd gleich. Im Deutschen ist diese Annahme aufgrund der freien Wortstellung nicht mehr haltbar, da hier der Kasus und nicht die Position eines Wortes entscheidend ist. Da Word2vec keinerlei Syntax beherrscht bzw. keinerlei Informationen über Kasus hat, passt das Word2vec Modell damit nur eingeschränkt zur deutschen Sprache.

Unglücklicherweise ist der Google-News Datensatz nicht öffentlich zugänglich, sodass eine genauere Untersuchung der verwendeten Texte nicht möglich ist. Dennoch ist es Fakt, dass sich jede Sprache in weitere Subsprachen wie z.B. Fachsprache oder Umgangssprache aufteilen lässt. Durch den lexikalischen Charakter der Wikipedia treten gewisse Wortkategorien wie z.B. dem Partizip Präsens nur sehr selten auf. Ähnliches gilt für den Superlativ oder spezielle Währungsnamen. Des Weiteren ist es offensichtlich, dass ein statistisches Verfahren lediglich diejenige Verteilung finden kann, welche den Trainingsdaten zugrunde liegt. Der von Mikolov et. al. konzipierte Testdatensatz macht jedoch keinerlei Annahmen über die Verteilung der Wörter im Text, sodass dieser Testdatensatz unter Umständen gar nicht zu den Trainingsdaten passt. Im Vergleich zu den Wikipedia-Korpora lässt sich annehmen, dass der Google-News Korpus durch seine Größe und den abgedeckten Themenbereich einen deutlich heterogeneren Wortgebrauch zeigt, wodurch dieser besser zu den Testdaten passt und es so zu besseren Ergebnissen kommt.

6.6 Zusammenfassung

In dieser Arbeit wurde die Anwendbarkeit des Word2vec Modelles für verschiedene Textkorpora untersucht. Hierzu wurde zunächst der Testdatensatz von Mikolov et. al. ins Deutsche übersetzt und anschließend verschiedene Vektorkonfigurationen auf verschiedenen Textkorpora trainiert.
Es zeigt sich, dass Word2vec zunächst grundsätzlich auch für die deutsche Sprache anwendbar ist und in einigen Testkategorien sogar bessere Ergebnisse erzielt werden können als dies in der englischen Sprache möglich ist. Alles in Allem schneiden die deutschen Vektoren jedoch deutlich schlechter ab als ihr englisches Pendant. Die Gründe hierfür liegen zum Einen im Problem der freien Wortstellung, welches von Word2vec nur unzureichend modelliert wird und zum Anderen an dem großen Vokabular der deutschen Sprache im Vergleich zu den verfügbaren Trainingsdaten.

An dieser Stelle zeigen sich auch Diskrepanzen zwischen den Vektoren für die englische Sprache die auf verschiedenen Trainingsdaten trainiert worden sind. Die Wortvektoren bilden vor Allem die semantischen Eigenarten der Sprache des Trainingskorpus ab und generalisieren nur bedingt darüber hinaus. Diese Subsprache entspricht trotz Einsatz großer Datenmengen nicht notwendigerweise dem üblichen Bild der Gesamtsprache. Anders ausgedrückt: Word2Vec kann nur diejenige Verteilung approximieren, die es auch in den Trainingsdaten finden kann. Diese zunächst triviale Aussage hat jedoch weitreichende Folgen. Häufig wird in der Sprachverarbeitung angenommen, dass die verwendeten

Trainingsdaten die Gesamtverteilung einer Sprache hinreichend widerspiegeln, d.h. es wird weder direkt zwischen Zeitungssprache, Fachsprache oder Umgangssprache unterschieden, noch sind die Testdaten bzw. Modelle auf diese Subsprachen angepasst. Diese Unterscheidung scheint aber wichtigen Einfluss auf die Performanz eines Systems zu haben, da die Sprache sich je nach Kontext mehr oder weniger stark verändert.

Abschließend sei noch anzumerken, dass die Kosinusähnlichkeit für Synonyme trotz der Schwäche von Word2vec im Deutschen deutlich bessere Ergebnisse erzielte als dies im Englischen der Fall war. Da die Trainingskorpora in beiden Sprachen hinreichend ähnlichen sind, sind die Gründe hierfür in den grundsätzlichen Unterschieden der deutschen und englischen Sprache zu suchen.

Kapitel 7

On Using Deep Neural Word Embeddings for German Dependency Parsing

by Lukas Pfahler

7.1 Motivation

When processing natural language data on computers, the way we represent text is crucial. Different levels of abstraction are possible - we can represent text as a sequence of characters, a sequence of words or even sequences of larger text snippets. When representing text as words, traditionally we represent words as sparse 1-of-n vectors $\vec{w} \in \{0,1\}^V$, where we know the vocabulary of size V and each word vector has exactly one non-zero entry corresponding to the word[35].

Recently, Mikolov et al. introduced Deep Word Embeddings[52], where each word is represented using a dense, low-dimensional vector $\vec{w} \in \mathbb{R}^D$, $D \ll V$ called word embedding. These representations can be efficiently trained using simple neural networks[50]. They exhibit somewhat surprising properties: Experiments for the English language show that two words a and b whose word embeddings w_a and w_b have a great cosine similarity $sim(\vec{w}_a, \vec{w}_b) = \langle \vec{w}_a, \vec{w}_b \rangle \cdot (||\vec{w}_a|| \cdot ||\vec{w}_b||)^{-1}$ have a similar meaning or function. Furthermore the difference of embeddings $\Delta = \vec{w}_a - \vec{w}_b$ is related to the relationship of the two words – syntactical relationships like singular/plural or adjective/superlative and even semantic relations like country/capital or man/woman are identified with high accuracy.

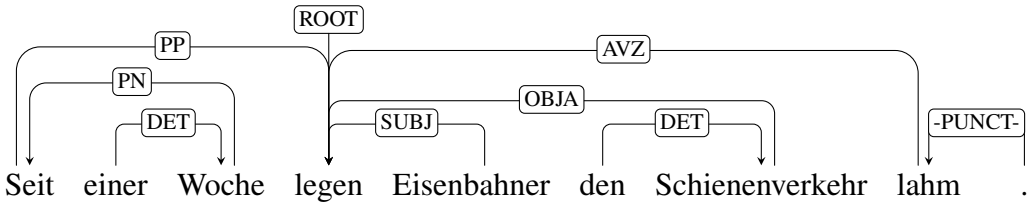


Abbildung 7.1: Example dependency tree from the TüBa-D/Z tree bank.

I want to investigate if these regularities can be used in dependency parsing. In dependency parsing, we model the syntactical dependencies between words within a sentence

by directed arcs in a graph. Usually, we restrict ourselves to trees rather than general graphs, i.e., each word has exactly one parent. For an example parse tree see figure 7.1. One essential step of dependency parsing is identifying the grammatical relationship between two words in a sentence[47]. Since the difference of two word embeddings seems to contain information about the relationship between the two corresponding words, it seems worthwhile to investigate whether we can use these differences to determine the grammatical relationship of two words in a sentence. This could lead to new, more accurate dependency grammars. Recently, there have already been attempts to improve existing parsers by incorporating new features based on word embeddings. Jacob and Klein investigated their usefulness for constituency parsing[3], Hisamoto et al.[31] and Bansal et al.[6] for dependency parsing. However there have not been attempts to use the difference of word embeddings as features for identifying the type of dependency.

The rest of this paper is organized as follows: In section 2, I explain how the word embeddings used in all of my experiments have been trained. Then, in section 3, I present the dataset I created for my experiments and explain a learning task relevant to dependency parsing that I want to solve using this data. In section 4, I experimentally compare three different approaches to solve the task, first without using word embeddings, then using only word embeddings and finally using a mixture of traditional word representation and word embeddings. In the last section I summarize my results and present further research endeavors to follow up on my results.

7.2 Learning Word Embeddings

Good word embeddings are substantial for all following experiments. Mikolov et al. show that for English language training a skip-gram neural network[1] with dimensionality of at least 300 is the best way to obtain word embeddings[50, 51]. Furthermore, the authors show that word embedding quality increases with increasing amounts of training data. Thus, when training word embeddings for the German language, we have to obtain a large training corpus.

A convenient source for German text data is the German Wikipedia[2], the data is licensed under Creative Commons and a snapshot of the current version of all Wikipedia pages is available for download[3]. Additionally I use 8.2MB of text data from the TüBaD/Z tree bank[75]. This way I obtained a 3.9GB training corpus of more than 591 million words with a vocabulary of more than 1.6 million different words. Note that this amount of training data is small in comparison to the data used for training by Mikolov et al. – text data from Google News – which consists of more than 6 billion words. Furthermore, with Wikipedia being a collection of encyclopedia documents, it only covers a limited subset of the German language with respect to vocabulary and syntactic structures. This might have negative effects on the quality of word embeddings.

[1] for a detailed review of skip-gram, see Chapter 6.

[2] `http://de.wikipedia.org`

[3] `http://dumps.wikimedia.org/dewiki/20140813/`

Training of the word embeddings is done using the freely available software Word2Vec[4] that implements the skip gram model training. In order to apply Word2Vec, some preprocessing to the text data had to be done: All non-ASCII characters were either replaced by close ASCII characters (ä ↦ae, ß ↦ss, etc.) or removed, all characters were converted to lowercase, sentences were separated by line breaks and all punctuation and unnecessary whitespace were removed.

7.3 Experimental Analysis

The main contribution of this paper are the following experiments: First I present the learning task closely related to dependency parsing that I want to solve and the corresponding data set. Then I describe experimental results for solving the learning problem with different approaches.

7.3.1 Preparation of Training Data

The following experiments are based on the TüBa-D/Z treebank for the German language[75] that is available in the `.conll` dependency treebank format. TüBa-D/Z contains 1,165,305 hand annotated dependencies, there are 34 different classes of dependencies[23], the biggest class is `DET` - determiners - with 13.97% of the occurrences, see Figure 7.2 for all classes. The TüBa treebank is most commonly used to train both constituency and dependency parsers. However I am going to analyze closely related proxy problems.

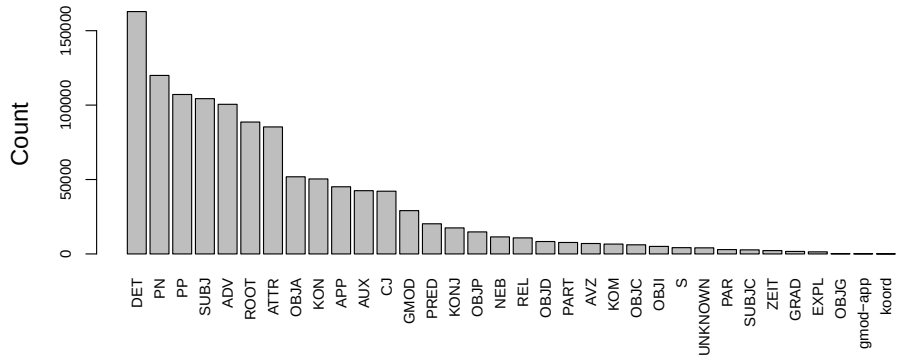


Abbildung 7.2: The classes of dependencies in the TüBa-D/Z treebank with their respective frequencies. See [23] for a detailed description of the classes.

I prepare one dataset for all the following experiments: I take every dependency $a \overset{t}{\rightarrow} b$ of class t in the tree bank and translate it into a new example (a, b, Δ) with $\Delta = \vec{w}_a - \vec{w}_b$. Each of these examples is labeled by the corresponding dependency class t. The class `ROOT` has a unique role in dependency parsing, as it is not a binary but a unary relation.

[4] `https://code.google.com/p/word2vec/`

Thus I exclude it from the test. But, according to Moschitti et al. the task of identifying the predicate of a sentence, which will be the root of a dependency tree, is *quite easy*[56].

The learning task for this dataset is as follows: Train a classifier that, given two words that are in a binary dependency relation, predicts the type of the dependency. This task is very relevant to dependency parsing, take for instance the MSTParser dependency parser[47], which works in two steps: First the set of arcs that build the parse tree is identified, then the arcs are labeled with the dependency classes. The second labeling step is closely related to my classification problem, however I admit that my problem is a simplification: In a parser, the output of the classifier is further constrained, e.g. a sentence can only have one subject. However, I believe that I have identified an interesting, relaxed proxy problem.

7.3.2 Baseline with Traditional Word Representations

First I want to investigate how well the classification task works without applying word embeddings, i.e. given two words in dependence I want to predict the type of dependency. A simple baseline is using Naive Bayes classification.

Using Bayes rule, the probability that a dependency between words a and b is of type l is

$$P(Label = l \mid Start = a, End = b) = \frac{P(Start = a, End = b \mid Label = l)P(Label = l)}{P(Start = a, End = b)}.$$

We classify by assigning the type with the highest probability, thus only the numerator is important. Naive Bayes assumes conditional independence of the two words, i.e.,

$$P(Start = a, End = b \mid Label = l) = P(Start = a \mid Label = l)P(End = b \mid Label = l).$$

These probabilities can be estimated by counting occurrences in a single pass over the tree bank. When predicting labels for previously unseen word-label combination this formulation would assign zero probability. This problem will be treated by applying Laplace smoothing, a common technique in Natural Language Processing[36].

I test the accuracy of Naive Bayes classification using disjunct training and testing data sets. Through all the experiments in this paper the data is sampled linearly, i.e., not at random, because the data is in chronological order and thus the sets are almost independent and identically distributed.

The performance of this simple model is surprisingly good. Measured using a ten-fold cross-validation, it achieves an accuracy of $80.79 \pm 0.65\%$ on predicting the dependency classes. Given that there are 34 different classes, I believe that this value is high; in fact it is substantially higher than the accuracy achieved by the default-learner. To test how the amount of training data influences the prediction quality, I run an experiment using different train/test split ratios. As we can see in figure 7.3, the accuracy increases monotonically but shows signs of convergence at around 80%.

Why does this work so good? Words can only have certain grammatical functions in a text. For example determiners are very easily identified – an arc starting at 'der','die' or

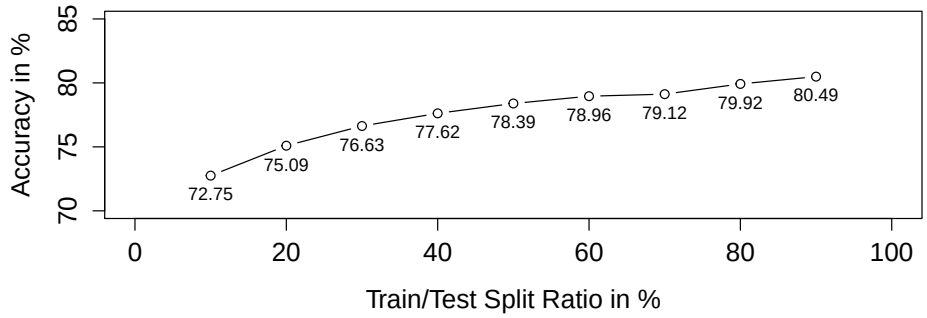


Abbildung 7.3: The accuracy using a Naive Bayes classifier predicting the dependency relationships in the test dataset using different amounts of training data. Note that for large amounts of training data the variance of the accuracy is higher, which might explains the small knee at 70%.

'das' will almost certainly have the label `DET`. Some other words like 'ich' or 'du' are almost always the subject and thus the corresponding arcs will be labeled `SUBJ`. Even in more complex cases Naive Bayes is effective: Words pointing at the predicate of the sentence will most likely be subject, object or adverbial phrase. The type of word already contains a lot of information to distinguish between those. Of course some words, especially names, can be either subject or object in a sentence. However, choosing the one observed more frequently seems to be a good strategy.

7.3.3 Prediction Using Only Word Embeddings

I try to solve a task closely related to the word relation task described by Mikolov et al.[52]: Given two word embeddings for words in a specific relation, they calculate the difference Δ. Then, for a new word embedding w_c they find the embedding closest to $w_c + \Delta$ to find the word that has the same relation to c. I have a given set of differences with relation labels (Δ_i, y_i) and take two word embeddings $\vec{w}_a$ and $\vec{w}_b$, calculate the difference $\Delta = \vec{w}_a - \vec{w}_b$ and predict the relationship $\widehat{y} = y_j$ with

$$j := \arg\max_i \{sim(\Delta_i, \Delta)\} .$$

Thus I use a k-NN classifier[28] with Cosine similarity.

In order to make k-NN efficient for the big amount of data, an indexing scheme for cosine similarity is necessary. Various exact and approximate indexing schemes have been proposed, I choose Locality Sensitive Hashing[34, 17], a simple approximate indexing structure that partitions the search space using a set of random hyperplanes and only calculates exact similarities for the set of examples lying in the same partition. Using one set of ten random hyperplanes, I am able to train a nearest neighbor model for about 1.1 million examples and split-validate the model on about 116,000 examples.

First, I want to investigate the influence of the choice of k. Using a 90% training set and

a 10% test split I test the prediction accuracy for all odd k between 1 and 19. As we can see in figure 7.4, the prediction quality seems to work best for $k = 1$. This suggests that the training dataset does not cover the high-dimensional space of word embedding differences sufficiently good. It may be worthwhile to use more annotated examples.

To get more reliable results, I cross-validate the nearest neighbor model for $k = 1$ using 10 linearly split test folds and achieve an accuracy of $70.14 \pm 0.72\%$, which is about 10% lower than the performance achieved using Naive Bayes. Please note that the implemented inexact indexing scheme might have a small negative effect on the overall performance, since in some cases the nearest neighbor computation returns the wrong example.

7.3.4 Improving Naive Bayes with Word Embeddings

We have seen that Naive Bayes is superior to nearest neighbor classification with word embedding distances. But maybe we can use word embeddings to further improve the Naive Bayes classification? One weakness of Naive Bayes might be the independence assumption. However, when we abandon the assumption, we have to count over pairs of words instead of single words. Our dataset contains 1.6 million words, thus we count over a space of 2.56 trillion possible pairs. We do not have nearly the amount of training data needed to accurately estimate probabilities for this large space of events.

However, word embeddings could provide a solution for this problem. Similar words should have similar word embeddings. Clustering the embeddings with k-Means thus gives me k partitions of similar words. Now I can estimate probabilities for pairs of embedding clusters instead of pairs of words. When I set $k \approx \sqrt{V}$, the space of events I want to estimate probabilities for, is about the same size as in the previous Naive Bayes Experiment. This method relates to two possible advantages of using word embeddings for parsing that Jacob and Klein identified: First, I can handle unknown words the same way as words with similar embeddings. Second, I can pool related words and get more reliable probability estimates for rare words[3]. Hisamoto et al. also tried to improve parsers by annotating words with embedding cluster features[31]. It also relates to the features Bansal et al. propose to enrich existing dependency parsers: They use agglomerative clustering and encode the cluster assignment in bit string features[6].

I use the embedding cluster assignments as new features for the Naive Bayes classifier, these two features will be called *Clusters*. I also introduce a feature for the pair of both cluster assignments. I call this feature *Full Bayes*, since I want to use it to approximately abandon the independence assumption of Naive Bayes as described above.

I experimentally test the performance achieved by Naive Bayes using different subsets of these features by applying 10-fold cross-validation. Using only words in section 3.2, I obtained an accuracy of $80.79 \pm 0.65\%$. See the results for the different subsets of features below:

features:	words	no words
using Clusters	**82.07 ± 0.58%**	$58.34 \pm 0.78\%$
using Full Bayes	$81.41 \pm 0.69\%$	$55.92 \pm 0.96\%$
using all features	$81.10 \pm 0.66\%$	$57.07 \pm 0.93\%$

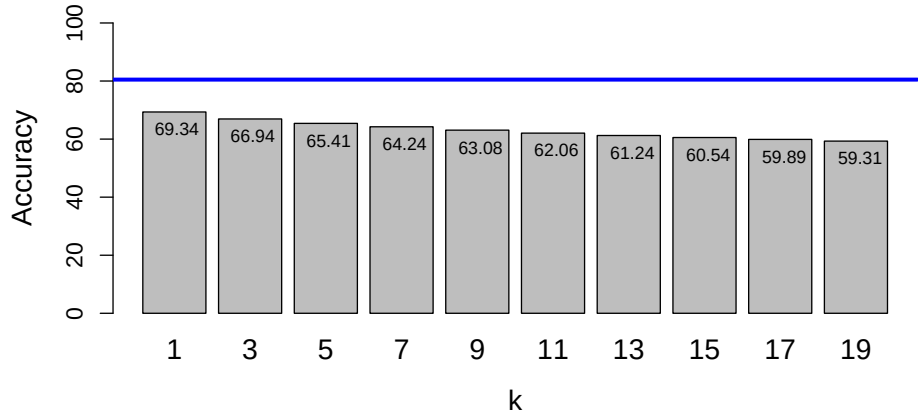


Abbildung 7.4: The overall accuracy for k-NN classification of the differences of word embeddings for predicting dependency classes. The blue line marks the performance achieved by Naive Bayes on the same train-test split.

Obviously, the words are the most important features, because the accuracies obtained by only using word embedding clusters are significantly worse. However, using words and features based on embeddings improves performance by around 1.0%, which is a significant, but not overly impressive increase.

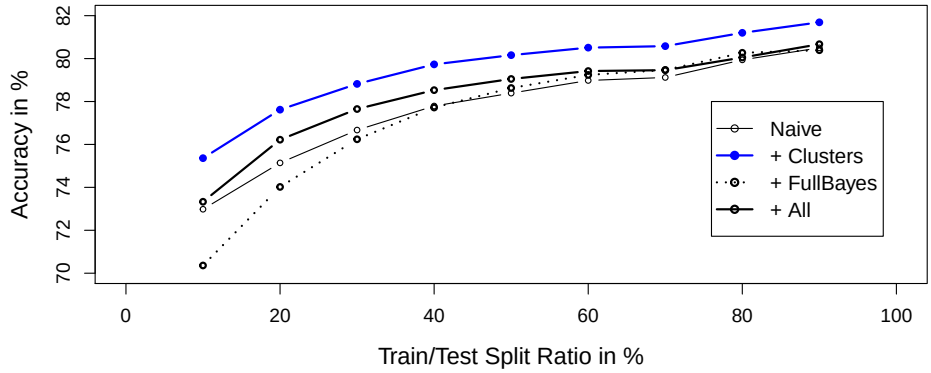


Abbildung 7.5: Progress of Naive Bayes classification accuracy for the different subsets of features.

I also want to investigate what influence the amount of training data has on the classification performance. As you can see in figure 7.5 for little training data using all features is superior to using only the words, but when using 90% of the data for training, this advantage is gone. However, using the cluster features in addition to the words is superior to every other subset of features on every train/test split.

Overall, my results align well with the results presented by Bansal et al. as well as Jacob

and Klein, where the incorporation of word embedding based features into existing parsers only yielded small improvements over the baseline[6, 3] and the effect was bigger for small training sets[3].

7.3.5 Final Comparison of Classification Results

Finally, I want to once more analyze my results and compare the errors made by my best Naive Bayes model and the 1-NN model. I focus on the 12 biggest dependency classes, i.e. all the classes that account for more than 2% of the example instances. See figure 7.6 for a comparison of precision and recall for the best Naive Bayes and best k-NN model.

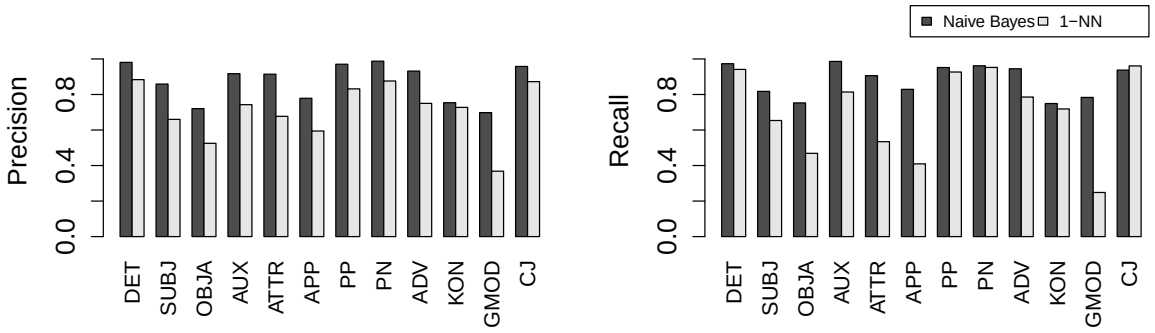


Abbildung 7.6: A comparison of precision and recall for the biggest 12 classes of dependencies for Naive Bayes and 1-NN classification.

As you can see, Bayes is superior in both precision and recall on all but one class. 1-NN is worse in distinguishing between subjects and accusative objects than Naive Bayes, but it is particularly bad at detecting appositions(`APP`), attributes(`ATTR`) and genitive attributes(`GMOD`). Looking at the contingency table in table C.1, we see that 1-NN does not confuse two particular classes, but almost evenly spreads its errors over all labels.

	DET	SUBJ	OBJA	AUX	ATTR	APP	PP	PN	ADV	KON	GMOD	CJ
DET	**151973**	3404	2367	128	5549	2605	481	459	1618	963	2308	127
SUBJ	3032	**60923**	6645	4370	4375	3176	956	699	3639	1833	2499	182
OBJA	1498	5849	**21692**	515	2963	1915	723	532	2708	1079	1714	121
AUX	141	5738	945	**32250**	1072	682	241	124	886	827	467	55
ATTR	1328	2390	2191	207	**42336**	4655	697	552	2357	1712	3947	156
APP	558	1408	1241	111	3661	**16980**	428	368	1111	867	1693	138
PP	491	2797	2344	417	3892	2173	**90469**	503	2660	1117	1727	189
PN	540	2072	2083	147	3221	2335	839	**113396**	1745	1048	1841	201
ADV	881	4803	3684	741	4782	2778	1316	1349	**74212**	2256	1953	203
KON	333	1797	1157	579	2390	1431	951	534	1893	**32741**	1018	189
GMOD	508	1382	1292	72	3748	1917	293	320	902	676	**6517**	69
CJ	187	655	642	87	1242	825	259	234	726	445	550	**39971**

Tabelle C.1: Contingency Table for 1-NN: True dependencies are printed horizontally and predicted dependencies vertically.

7.4 Conclusion

I have shown that parsing using traditional word representations is more accurate than using differences of word embeddings. But word embeddings can still be used to generate additional features to improve classifiers. Possible future research could analyze the usefulness of using the difference of word embeddings discretized by k-Means as features.

The next step would be to incorporate this new set of features into an existing dependency parser as Bansal et al. did in [6]. However, at this point, I do not believe that the word embedding features will improve the parser substantially, since my experiments yielded only small improvements and the related literature also did not achieve substantial enhancements.

Following the results of Buschjäger (c.f. Chapter 6) that word embeddings do not capture linguistic regularities for the German language as well as for the English language, it might be worthwhile to repeat the experiments for English dependency parsing. Maybe for English, using 1-NN classification and word embedding differences achieves competitive accuracies.

Throughout the course of this project, it became increasingly clear that further theoretical understanding of word embeddings is necessary, particularly with respect to the structure or topology of the embedding space and the influence of the embedding dimension. Like Bansal et al. I believe that the word embedding should be tailored to the task we want to solve and incorporate exactly those aspects of language we need to solve that task[6]. Only when we have identified and defined what properties of word embeddings we need, can we find an efficient way to train them and then apply them – with theoretical understanding – to the NLP task we want to solve.

7.5 Acknowledgment

All experiments were performed using RapidMiner[5]. RapidMiner was extended by a new operator that implements Locality Sensitive Hashing for Cosine Similarity.

[5] `https://rapidminer.com/`

Literaturverzeichnis

[1] Statistiken zur Wikipedia. `http://de.wikipedia.org/wiki/Wikipedia:Statistik`. [Online; zugegriffen am 18.02.2015].

[2] Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Polyglot-NER: Massive multilingual named entity recognition. *CoRR*, abs/1410.3791, 2014.

[3] Jacob Andreas and Dan Klein. How much do word embeddings encode about syntax. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014.

[4] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models - going beyond SVD. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 1–10, 2012.

[5] Giuseppe Attardi and Antonio Fuschetto. Wikipedia Extractor. `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`, 2015. [Online; zugegriffen am 18.02.2015].

[6] Mohit Bansal, Kevin Gimpel, and Karen Livescu. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2014.

[7] Michael Beißwenger, Thomas Bartz, Angelika Storrer, and Swantje Westpfahl. Richtlinie für das manuelle PoS-Tagging. EmpiriST2015 : Shared Task des Empirikom-Netzwerks zur automatischen linguistischen Annotation deutschsprachiger internetbasierter Kommunikation, June 2014.

[8] Michael Beißwenger, Maria Ermakova, Alexander Geyken, Lothar Lemnitzer, and Angelika Storrer. DeRiK: A German reference corpus of computer-mediated communication. *Oxford Journal*, 2013.

[9] David M. Blei, Lawrence Carin, and David Dunson. Probabilistic topic models. *IEEE Signal Processing Magazine*, 27(6):55–65, 2010.

[10] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[11] Bernd Bohnet and Jonas Kuhn. The best of bothworlds – a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the*

European Chapter of the Association for Computational Linguistics, pages 77–87, Avignon, France, April 2012. Association for Computational Linguistics.

[12] Bernd Bohnet and Joakim Nivre. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[13] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8), 1991.

[14] Samuel Brody and Mirella Lapata. Bayesian word sense induction. *Computational Linguistics*, (April):103–111, 2009.

[15] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. Class-based n-gram models of natural language. *Comput. Linguist.*, 18(4):467–479, December 1992.

[16] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[17] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing - STOC '02*, pages 380–388, 2002.

[18] Grzegorz Chrupała and Dietrich Klakow. A named entity labeler for German: Exploiting Wikipedia and distributional clusters. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, Mai 2010.

[19] Simon Clematide, Stefan Gindl, Manfred Klenner, Stefanos Petrakis, Robert Remus, Josef Ruppenhofer, Ulli Waltinger, and Michael Wiegand. MLSA – a multi-layered reference corpus for German sentiment analysis. pages 3551–3556. European Language Resources Association (ELRA), 2012.

[20] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.

[21] Corina Dima and Erhard W. Hinrichs. A semi-automatic, iterative method for creating a domain-specific treebank. In Galia Angelova, Kalina Bontcheva, Ruslan Mitkov, and Nicolas Nicolov, editors, *Proceedings of Recent Advances in Natural Language Processing, RANLP 2011*, pages 413–419. RANLP 2011 Organising Committee, 2011.

[22] Markus Dollmann and Michaela Geierhos. Sentiba: Lexicon-based sentiment analysis on German product reviews. In Gertrud Faaßand Josef Ruppenhofer, editors, *Workshop Proceedings of the 12th Edition of the KONVENS Conference*, pages 185–191, Hildesheim, Germany, October 2014. Universität Heidelberg.

[23] Kilian Foth. *Eine umfassende Dependenzgrammatik des Deutschen*, 2006.

[24] Apache Foundation. Lingucomponent Sub-Project: Thesaurus Development. `http://www.openoffice.org/lingucomponent/thesaurus.html`, 2015. [Online; zugegriffen am 18.02.2015].

[25] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101 Suppl:5228–5235, 2004.

[26] Ralph Grishman. Information extraction: Capabilities and challenges. `http://www.cs.nyu.edu/grishman/tarragona.pdf`, 2012. [Online; zugegriffen am 30.11.2015].

[27] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

[28] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[29] Benjamin Helbig. Eine Umgebung zur Informationsextraktion aus Geschäftsbriefen, 2005.

[30] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Distributed Representations, pages 77–109. MIT Press, Cambridge, MA, USA, 1986.

[31] Sorami Hisamoto, Kevin Duh, and Yuji Matsumoto. An Empirical Investigation of Word Representations. In *Proceedings of ANLP*, number C, 2013.

[32] Thomas Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM Press, 1999.

[33] Christian Hüppe. Benutzergeführtes Lernen von Dokument-Strukturauszeichnungen aus Formatierungsmerkmalen, 2003.

[34] Piotr Indyk and Alexandr Andoni. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *47th IEEE Symposium on Foundations of Computer Science*, 51(1):117–122, 2006.

[35] Thorsten Joachims. *Learning to Classify Text using Support Vector Machines*, volume 668 of *Kluwer International Series in Engineering and Computer Science*. Kluwer, 2002.

[36] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *Knowledge and Data Engineering, IEEE Transactions on*, 18(11):1457–1466, 2006.

[37] Wolfgang Klein. Von Reichtum und Armut des deutschen Wortschatzes. In *Reichtum und Armut der deutschen Sprache*, pages 15–55. de Gruyter, 2013.

[38] Roman Klinger and Katrin Tomanek. Classical probabilistic models and conditional random fields. Technical report, 2007.

[39] Geert-Jan M. Kruijff, Oliver Plaehn, Holger Stenzhorn, and Thorsten Brants. *NEGRA Korpus*, 2006. [Online; zugegriffen am 21.02.2015].

[40] Sandra Kübler, Erhard W. Hinrichs, and Wolfgang Maier. Is it really that difficult to parse German? In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 111–119. Association for Computational Linguistics, 2006.

[41] Sandra Kübler and Jelena Prokic. Why is German dependency parsing more reliable than constituent parsing? In *Proceedings of the Fifth International Workshop on Treebanks and Linguistic Theories - 2006*, pages 7–18, Prague, Czech Republic, December 2006.

[42] Sangkyun Lee, Benjamin Schowe, Viswanath Sivakumar, and Katharina Morik. Feature selection for high-dimensional data with rapidminer. Technical Report 1, TU Dortmund University, 2011.

[43] Percy Liang. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

[44] Brian MacWhinney, Elizabeth Bates, and Reinhold Kliegl. Cue validity and sentence interpretation in English, German, and Italian. *Journal of verbal learning and verbal behavior*, 23(2):127–150, 1984.

[45] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: the Penn treebank. *Comput. Linguist.*, 19:313–330, June 1993.

[46] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[47] Ryan McDonald, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 216–220. Association for Computational Linguistics, 2006.

[48] Wolfgang Menzel. *Sprachverarbeitung*, chapter 13, pages 473–526. Oldenbourg, 2014.

[49] Tomas Mikolov. word2vec – Tool for computing continuous distributed representations of words. `http://code.google.com/p/word2vec/`, 2015. [Online; zugegriffen am 16.02.2015].

[50] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[51] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[52] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751, 2013.

[53] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[54] Manuela Moraes. Glosbe API. `https://glosbe.com/a-api`, 2015. [Online; zugegriffen am 18.02.2015].

[55] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252. Citeseer, 2005.

[56] Alessandro Moschitti, Daniele Pighin, and Roberto Basili. Tree kernels for semantic role labeling. *Computational Linguistics Journal*, Special Issue on Semantic Role Labeling, 2008.

[57] Tony Mullen and Nigel Collier. Sentiment analysis using support vector machines with diverse information sources. In *EMNLP*, volume 4, pages 412–418, 2004.

[58] Daniel Naber. OpenThesaurus: ein offenes deutsches Wortnetz, March 2005.

[59] Daniel Naber. OpenThesaurus.de. `https://www.openthesaurus.de/about/download`, 2015. [Online; zugegriffen am 18.02.2015].

[60] William E Nagy and Richard C Anderson. How many words are there in printed school english? *Reading Research Quarterly*, pages 304–330, 1984.

[61] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194(0):151–175, 2013. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

[62] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[63] Livia Polanyi and Annie Zaenen. Contextual valence shifters. In JamesG. Shanahan, Yan Qu, and Janyce Wiebe, editors, *Computing Attitude and Affect in Text: Theory and Applications*, volume 20 of *The Information Retrieval Series*, pages 1–10. Springer Netherlands, 2006.

[64] R. Remus, U. Quasthoff, and G. Heyer. Sentiws – a publicly available German-language resource for sentiment analysis. In *Proceedings of the 7th International Language Resources and Evaluation (LREC'10)*, pages 1168–1171, 2010.

[65] Stephen Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60:2004, 2004.

[66] Marc Roessler and Katharina Morik. Using unlabeled texts for named-entity recognition. In Tobias Scheffer and Stefan Rüping, editors, *ICML Workshop on Multiple View Learning*, 2005.

[67] David E. Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5, 1988.

[68] Josef Ruppenhofer, Roman Klinger, Julia Maria Struß, Jonathan Sonntag, and Michael Wiegand. Iggsa shared tasks on German sentiment analysis (GESTALT). In Gertrud Faaßand Josef Ruppenhofer, editors, *Workshop Proceedings of the 12th Edition of the KONVENS Conference*, pages 164–173, Hildesheim, Germany, October 2014. Universität Heidelberg.

[69] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, November 1975.

[70] Wolfgang Seeker and Jonas Kuhn. Making ellipses explicit in dependency conversion for a German treebank. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA).

[71] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[72] Angelika Storrer, Katharina Morik, Alexander Geyken, Erhard W. Hinrichs, Marc Kupietz, and Andreas Witt. *KobRA Projekt*, 2014. [Online; zugegriffen am 22.02.2015].

[73] Charles Sutton and Andrew Mccallum. *Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[74] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June 2011.

[75] Heike Telljohann, Erhard W. Hinrichs, Sandra Kübler, Heike Zinsmeister, and Kathrin Beck. Stylebook for the Tübingen treebank of written German (TüBa-D/Z).

[76] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL

'03, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[77] Antonio Toral and Rafael Muñoz. A proposal to automatically build and maintain gazetteers for named entity recognition by using Wikipedia. Technical report, 2006.

[78] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *In Proceedings of HLT-NAACL 2003*, pages 252–259, 2003.

[79] Peter D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[80] Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *CoRR*, abs/1003.1141, 2010.

[81] Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.

[82] Casey Whitelaw, Alex Kehlenbeck, Nemanja Petrovic, and Lyle Ungar. Web-scale named entity recognition. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 123–132, New York, NY, USA, 2008. ACM.

[83] Michael Wiegand, Christine Bocionek, Andreas Conrad, Julia Dembowski, Jörn Giesen, Gregor Linn, and Lennart Schmeling. Saarland university's participation in the GErman SenTiment AnaLysis shared Task (GESTALT). In Gertrud Faaßand Josef Ruppenhofer, editors, *Workshop Proceedings of the 12th Edition of the KONVENS Conference*, pages 174–184, Hildesheim, Germany, October 2014. Universität Heidelberg.

[84] Wikimedia. Wikipedia Dumps. `http://dumps.wikimedia.org/dewiki/latest/`, 2015. [Online; zugegriffen am 18.02.2015].

[85] Wikimedia. Wikipedia Dumps. `http://dumps.wikimedia.org/enwiki/latest/`, 2015. [Online; zugegriffen am 24.02.2015].

[86] Wikipedia. Probably approximately correct learning — Wikipedia, die freie Enzyklopädie, 2013. [Online; Stand 26. Februar 2015].