B A C H E L O R A R B E I T

im Studiengang
Medieninformatik (MI7)

Semantische Beziehungen in Texten mit Word2Vec

und der Vergleich zwischen allgemeinen und domänenspezifischen Korpora als Trainingsdaten

vorgelegt von Ruben Müller
an der Hochschule der Medien Stuttgart,
am 31.07.2015
zur Erlangung des akademischen Grades eines Bachelor of Science

Erstprüfer: Prof. Dr-Ing. Johannes Maucher,
Hochschule der Medien, Stuttgart
Zweitprüfer: M.Sc. Andreas Stiegler,
Hochschule der Medien, Stuttgart

Erklärung

Hiermit versichere ich, Ruben Müller, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit (bzw. Masterarbeit) mit dem Titel: „Semantische Beziehungen in Texten mit Word2Vec und der Vergleich zwischen allgemeinen und domänenspezifischen Korpora als Trainingsdaten“selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.
Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§26 Abs. 2 Bachelor-SPO (6 Semester), § 24 Abs. 2 Bachelor-SPO (7 Semester), § 23 Abs. 2 Master-SPO (3 Semester) bzw. § 19 Abs. 2 Master-SPO (4 Semester und berufsbegleitend) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Filderstadt, den 31. Juli 2015

Ruben Müller

Kurzfassung

Diese Bachelorarbeit beschäftigt sich mit der Analyse von semantischen Beziehungen innerhalb mit Word2Vec gelernten Modellen.
Dazu sollen zum einen der allgemeine komplette Wikipedia-Korpus gelernt und analysiert werden, was als semantisch ähnlich erkannt wird. Zum anderen soll ein Korpus über eine spezielle Domäne erstellt und gelernt werden. Welche spezielle Domäne analysiert und verglichen werden soll, wird im Laufe der Bearbeitung festgelegt.
Diese beiden Korpora sollen sich dann gegenüber gestellt und analysiert werden, was jeweils als semantisch ähnlich erkannt wird.
Ziel dieser Arbeit soll es sein, festzustellen ob ein allgemeiner Korpus oder ein spezieller Domänenkorpus genauere Resultate im Hinblick auf semantische Ähnlichkeiten erzielt. Anstatt eines allgemeinen Korpus zu verwenden, könnte es sich dann anbieten zwischen mehreren speziellen Korpora auszuwählen, je nachdem welche Domäne aktuell bearbeitet werden soll.

Abstract

This bachelor thesis deals with the analysis of semantic relations in models which are learnd by Word2Vec.
For this purpose on the one hand the whole and general wikipedia body shall be learned and analyzed what is detected as semantic similar. On the other hand a body covering a specific domain shall be created and learned. Which specific domain is choosen will be defined due to the working process.
These both bodys shall then be compared against each other to determine what is recognized as semanic similar.
Goal of this thesis will be to determine whether a general body or a domain specific body gives more precise results with regard to semantic similarity. Instead of using a general body it coud be better to choose between multiple domain specific bodys with regard to the actual domain.

Inhaltsverzeichnis

Begriffsverzeichnis

Begriff	Erklärung
Ähnliche Worte	Im Word2Vec-Modell, mit der Methode *most_similar()*, erhaltene Worte.
SVM	Support Vector Machine
NBC	Naive Bayes Classifier
CBOW	Continuous bag-of-words

Kapitel 1

Einleitung

1.1 Motivation und Problemstellung

Um semantische Beziehungen zwischen Worten abzubilden, gibt es unter anderem die Möglichkeit, Modelle mittels Word2Vec darzustellen. Wie dieses Modell die Beziehungen aufbaut, hängt nicht nur von den Trainingsparametern des Modells ab, sondern auch von den verwendeten Trainingsdaten. So können beispielsweise allgemeine Daten, die keine spezielle Fachdomäne abbilden, als Trainingsdatenkorpus benutzt werden oder aber ausschließlich Daten über eine spezielle Domäne.
In dieser Arbeit wird untersucht, wie sich unterschiedliche Textgrundlagen (Korpora) als Trainingsdaten auf die semantischen Beziehungen zwischen Worten im Word2Vec Modell auswirken.
Die Ergebnisse dieser Arbeit sollen es ermöglichen, dass für eine spezielle Fragestellung oder Anwendung des Word2Vec Modells, das richtige Modell, bzw die richtigen Trainingsdaten ausgewählt werden können.

1.2 Verwandte Arbeiten

1.3 Aufbau der Arbeit

Im folgenden Abschnitt wird kurz der Aufbau dieser Arbeit beschrieben.

Kapitel 2: Daten und Vorverarbeitung
In diesem Kapitel wird erläutert, welche Daten in dieser Arbeit benutzt wurden. Des Weiteren wird kurz erklärt, in welchen Vorverarbeitungsschritten diese Daten bearbeitet wurden um in den nächsten Kapiteln verwendet werden zu können.

Kapitel 3: Word2Vec
Hier werden kurz die unterschiedlichen Möglichkeiten im Word2Vec Modell erklärt und die wichtigsten Parameter vorgestellt.

Kapitel 4: Wikipedia-Korpus
In Kapitel 4 werden der komplette Wikipedia-Korpus sowie ein Teilkorpus daraus vorgestellt und die zum Training des Word2Vec Modells ausgewählten Parameter erläutert. Außerdem werden die Testdaten vorgestellt.

Kapitel 5: Experimente
Kapitel 5 beschäftigt sich mit den Ergebnissen der durch die Testdaten erhaltenen ähnlichen Worte der unterschiedlichen Word2Vec Modelle.

Kapitel 6: Fazit und Ausblick
In Kapitel 6 soll sollen die Ergebnisse der Problemstellung gegenüber gestellt und ein Ausblick auf zukünftige Arbeiten gegeben werden.

Kapitel 7: Anhang
Hier sind die Testdaten und weitere Auflistungen ausführlich dargestellt.

Kapitel 2

Daten und Vorverarbeitung

2.1 Datenbasis

Zum erfolgreichen Training des Word2Vec Modells wird eine sehr große Menge an Daten gebraucht[1].
In dieser Arbeit werden unterschiedliche Datenkorpora als Trainingsdaten für unterschiedliche Word2Vec Modelle benutzt.
Für den ersten Korpus wurde der komplette englischsprachige Wikipedia Korpus verwendet[2].
Der zweite in der Arbeit verwendete Korpus besteht auch aus Wikipedia Artikeln, allerdings wurden hier nur technologiespezifische Artikel verwendet. Der komplette englische Wikipedia Korpus wurde zuerst in die einzelnen Artikel aufgeteilt und diese wurden dann mit einem NBC in die Klassen *tech, entertainment, sport, science, politic* eingeteilt.

2.2 Externe Programme und Hilfsmittel

Dieser Abschnitt enthält eine Auflistung mit kurzen Beschreibungen, der in dieser Arbeit verwendeten Hilfsmittel und externen Programme.

gensim[ŘS10][3] ist eine Bibliothek für Python. Sie enthält unter anderem eine performanzoptimierte Implementierung von Word2Vec.

[1] In [MSC+13] werden Trainingsdaten mit bis zu 30 Milliarden Wörtern benutzt.
[2] Dump von März 2015, http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, abgerufen am 09.04.2015
[3] https://radimrehurek.com/gensim/, abgerufen am 24.06.2015

Der **NBC**, der zum Klassifizieren der Artikel verwendet wurde, ist der im DataMining Praktikum[4] selbst implementierte.

PyCharm[5] wurde als Editor und zum Ausführen der Python Skripte verwendet.

2.3 Vorverarbeitung

Der komplette Wikipediadump kann als komprimierte XML-Datei heruntergeladen werden[6]. Die entpackte XML-Datei hat eine Größe von 48,8 GB. Die Daten müssen zunächst bereinigt werden um als Trainingsdaten für Word2Vec Modelle zu dienen. Um das Wiki-Markup, wie unter anderem Links, Referenzen oder Zitate, und die XML-Tags zu entfernen, kann ein schon vorgefertigtes Perl Skript von Matt Mahoney benutzt werden[7], welches von Mikolov et al. auf der Google Code Seite von Word2Vec vorgeschlagen wird[8]. Dieses Skript wurde speziell zum Bereinigen von Wikipediadaten erstellt und wurde leicht verändert, sodass Zahlen, Satzzeichen und die Groß- und Kleinschreibung erhalten bleiben. Außerdem werden die Umlaute in *ae*, *oe* und *ue* umgewandelt[9]. Da die bis hierher bereinigten Daten eine Gesamtgröße von 18 GB haben, ist es nötig die Daten weiter aufzubereiten, sodass sie als Input für die Klasse *gensim.models.word2vec.LineSentence*[10], aus der Gensim Bibliothek dienen. Dazu müssen die einzelnen Sätze in je einer Zeile stehen, alle Worte klein geschrieben, frei von Satzzeichen und mit Leerzeichen getrennt sein.
Um den kompletten Wikipediadump in die einzelnen Wikipedia Artikel aufzuteilen muss die XML-Datei mittels eines SAX-Parsers[11] geparst und dann in einzelne Dateien geschrieben werden. Diese einzelnen Dateien können dann mit dem oben genannten Perl Skript gereinigt und dann mittels NBC klassifiziert werden. Nach der Klassifizierung können die Dateien wieder in eine große Datei zusammengefasst und weiter verarbeitet werden. Auch diese Daten müssen, wie der Gesamtkorpus, mit der Klasse *gensim.models.word2vec.LineSentence* verarbeitet werden.

[4] https://www.hdm-stuttgart.de/~maucher/Data_Mining_SS15.html und https://www.mi.hdm-stuttgart.de/mib/studium/intern/skripte/Data_Mining/WS1213/V4SpamFilter.pdf, abgerufen am 17.07.2015

[5] https://www.jetbrains.com/pycharm/, abgerufen am 02.07.2015

[6] http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, abgerufen am 09.04.2015

[7] http://mattmahoney.net/dc/textdata.html unter Appendix A, abgerufen am 25.06.2015

[8] https://code.google.com/p/word2vec/, abgerufen am 25.06.2015

[9] Für das komplette geänderte Skript siehe Anhang 7.2

[10] Diese Klasse kann auch als Input zum Trainieren des Word2Vec Modells genutzt werden.

[11] Es wurde die Klasse *xml.sax.handler.ContentHandler* verwendet.

Kapitel 3

Word2Vec

Im folgenden Kapitel wird kurz erklärt was Word2Vec[1] ist und welche wichtigen Parameter und Algorithmen bei der Berechnung und dem Erstellen von Word2Vec Modellen benutzt werden können.

Allgemeines zu Word2Vec,
Vergleich zu anderen
syntaktische und semantische Ähnlichkeiten werden mit word2vec abgebildet
Vektoraddition, -subtraktion (http://www.aclweb.org/anthology/N13-1090)

In Word2Vec Modellen werden Worte als Vektoren dargestellt[2]. Hier wird mittels *distributed representation*[MCCD13] ein n-dimensionaler Vektorraum erzeugt, in dem jedes Wort aus den Trainingsdaten durch einen Vektor dargestellt wird.
Als nächster Schritt werden die Vektoren in ein neuronales Netz gegeben und dort mittels eines Lernalgorithmus so verändert, dass Worte mit ähnlicher Bedeutung ähnliche Vektoren haben. Die Ähnlichkeit zwischen zwei Worten kann so berechnet werden, indem die Kosinusähnlichkeit zwischen den Vektordarstellungen der beiden Worte gebildet wird.
Die Berechnung der Wortvektoren kann mit neuronalen Netzen unterschiedlicher Architektur erreicht werden, *CBOW* oder *Skip-gram*. Desweiteren stehen unterschiedliche Lernalgorithmen für die neuronalen Netze zur Verfügung, *hierarchical softmax* und *negative sampling*.
Beim Training können unterschiedliche Parameter eingestellt werden.

[1] https://code.google.com/p/word2vec/
[2] Word2Vec heißt wörtlich Wort zu Vektor.

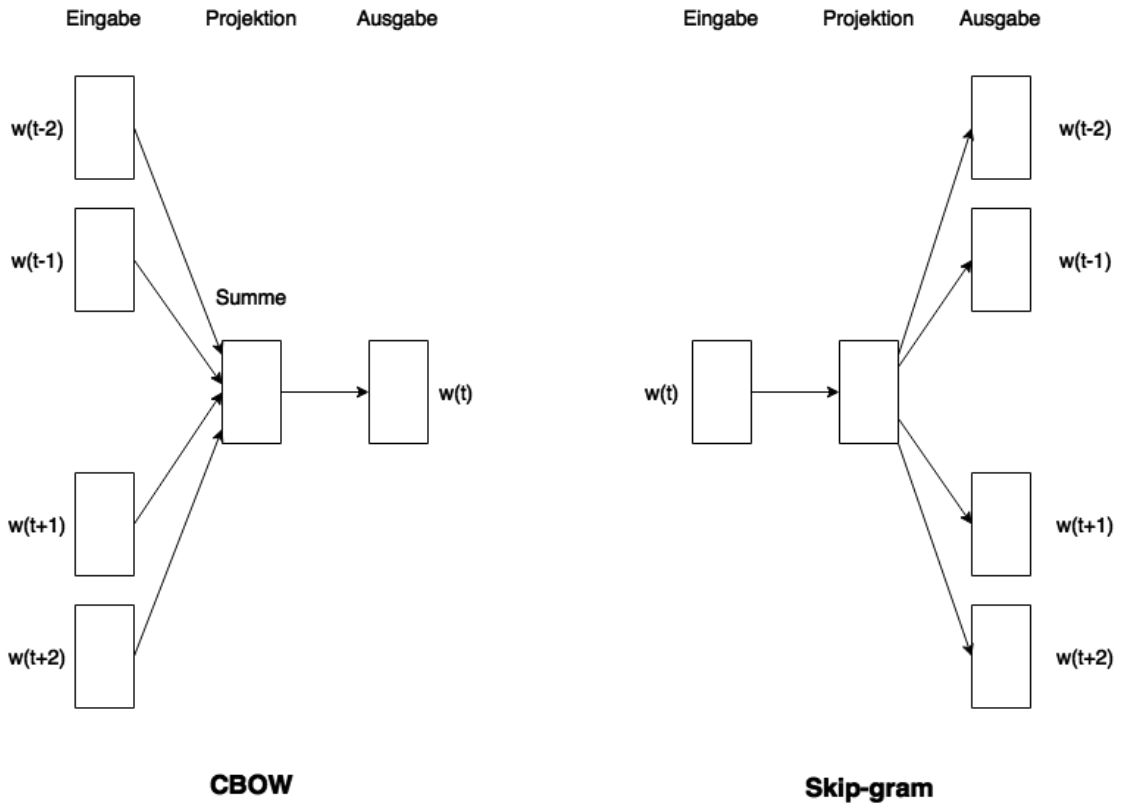

Abbildung 3.1: CBOW und Skip-gram im Vergleich, übersetzt nach [MCCD13]

3.1 CBOW

CBOW ist die Abkürzung für Continuous bag-of-words (dt. stetige Menge an Worten). Beim CBOW wird ein neuronales Netz ohne verdeckte Schichten (hidden layer) verwendet[MCCD13].
In der CBOW Architektur wird aus dem Kontext ein Wort vorhergesagt (siehe Abbildung 3.1). Die Anzahl, der aus dem Kontext zu verwendende Worte, wird mit dem Parameter *window* angegeben.

3.2 Skip-gram

Bei der Skip-gram Architektur wird auch, wie beim CBOW Model, ein neuronales Netz ohne verdeckte Schichten (hidden layer) verwendet [MCCD13].
Allerdings wird hier nicht ein Wort aus dem Kontext vorhergesagt, sondern aus einem Wort wird der Kontext vorhergesagt (siehe Abbildung 3.1).

Mehrere verdeckte Schichten in neuronalen Netzen machen die Modelle genauer, allerdings kommt auch die meiste Komplexität des ganzen Models von solchen verdeckten Schichten [MCCD13]. Mikolov et al. haben deshalb neuronale Netze ohne verdeckte Schichten bevorzugt, da damit sehr große Datenmengen viel effizienter gelernt werden können.

3.3 Hierarchical softmax

Eine Annäherung an das allgemeine softmax ist das hierarchical softmax [MSC+13]. Der Hauptvorteil ist, dass anstatt W Ausgabeknoten im neuronalen Netz nur ungefähr $log_2(W)$ Knoten ausgewertet werden müssen um die Wahrscheinlichkeiten zu errechnen.
Bei diesem Lernalgorithmus wird das Wörterbuch als Huffman Binärbaum dargestellt. Dies hat den weiteren Vorteil, dass häufig genutzte Worte kurze Kodierungen haben, was sich auf das Trainingstempo positiv auswirkt.

3.4 Negative sampling

Alternativ zum hierarchical softmax kann das negative sampling als Lernalgorithmus für das neuronale Netz verwendet werden [MSC+13].
Das negative sampling unterscheidet sich zum hierarchical softmax insofern, dass nicht die Ähnlichkeiten zu anderen Worten berechnet werden, sondern es wird davon ausgegangen, dass zufällig ausgewählte Worte mit einer hohen Wahrscheinlichkeit unähnlich, dem zu testenden Wort, sind. Wie viele solcher zufällig ausgewählter Worte benutzt werden sollen kann mit dem Parameter *negative* angegeben werden.

3.5 Parameter

size

Mit dem Parameter *size* wird die Anzahl der Dimensionen der Wortvektoren eingestellt. In einem n-dimensionalen Vektorraum nimmt n den Wert von *size* an.

window

window ist der maximale Abstand zwischen benachbarten Worten, innerhalb eines Satzes, die zur Berechnung der Wordvektoren betrachtet werden.

min_count

Der Parameter *min_count* gibt an, wie oft ein Wort in den Testdaten mindestens vorkommen muss, um in das Wörterbuch aufgenommen zu werden.

negative

Dieser Parameter wird nur benötigt, wenn als Lernalgorithmus negative sampling verwendet wird[3]. Er gibt an, wie viele zufällig ausgewählten Worte verwendet werden sollen.

[3]siehe 3.4 Negative sampling

Kapitel 4

Wikipedia-Korpus

Im Folgenden Kapitel werden die in der Arbeit verwendeten Korpora erläutert und die im Word2Vec Modell benutzten Parameter begründet. Des Weiteren werden die benutzten Testdaten vorgestellt.

4.1 Gesamtkorpus

Da sich die Qualität der Wortvektoren im Word2Vec Modell wesentlich mit der Menge an Trainingsdaten erhöht[MCCD13], werden möglichst große Textkorpora bevorzugt. Auf der Google Code Seite von Word2Vec[1], werden für Forschungszwecke einige Beispiele für online verfügbare große Korpora genannt. Unter anderem auch der neueste Wikipedia Auszug[2].
Da in dieser Arbeit ein allgemeiner und ein domänenspezifischer Korpus als Trainingsdaten für das Word2Vec Modell verglichen werden sollen, eignet sich der Wikipedia Korpus gut.
Wie in 2.3 beschrieben, müssen die Daten erst einer Säuberung unterzogen werden, um dann anschließend für das Training im Word2Vec Modell benutzt zu werden.
Der bereinigte, komplette englischsprachige Wikipedia Korpus[3] enthält 8.392.453 Artikel, 242.144.317 Sätze und 2.919.802.692 Worte.

Die Skip-gram Architektur verhält sich im Bezug auf syntaktische Ähnlichkeit etwas schlechter als die CBOW Architektur, allerdings ist die Skip-gram Architektur im Bezug auf semantische Ähnlichkeit der CBOW weit überlegen[MCCD13].

[1] https://code.google.com/p/word2vec/, abgerufen am 29.06.2015

[2] http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, abgerufen am 29.06.2015

[3] Dump von März 2015, http://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2, abgerufen am 09.04.2015

Aus diesem Grund wurde die Skip-gram Architektur ausgewählt.
Da der *hierarchical softmax* Lernalgorithmus eher für selten vorkommende Worte und der *negative sampling* Lernalgorithmus eher für häufig vorkommende Worte und niedrigdimensionale Vektoren geeignet ist[4], fiel die Wahl auf den *hierarchical softmax* Lernalgorithmus, da das Modell (wie im Folgenden gezeigt) eine hohe Dimension der Vektoren hat und auf technologiespezifische Daten getestet werden soll (siehe 7.1 Testdaten).

Um die optimalen Parameter für das Training des Modells herauszufinden wurde ein kleinerer Wikipedia Auszug[5] als Trainingsdaten benutzt und mit unterschiedlichen Parametern wurden mehrere Modelle gelernt und evaluiert (siehe Tabelle 4.1). Die Word2Vec-Klasse in der Gensim-Implementierung beinhaltet eine Evaluationsfunktion[6], die die Accuracy des Models berechnet. Die Funktion erwartet einen Dateinamen einer Datei, in der jede Zeile ein 4-Tupel ist und die einzelnen Abschnitte mit „: SECTION NAME“ unterteilt sind. Ein solches 4-Tupel besteht aus zwei Wortpaaren, deren Worte die gleiche Beziehung untereinander haben, zum Beispiel „Athens Greece Berlin Germany“, hier stellen beide Wortpaare die Beziehung zwischen einem Land und dessen Hauptstadt dar. Auf der Google Code Seite ist eine solche Datei als Beispiel vorhanden[7]. In diesem Beispiel sind 14 Kategorien[8] mit insgesamt 19544 4-Tupel aufgelistet.
Das Word2Vec Modell stellt eine Funktion bereit, *most_similar(positive=[], negative=[], topn=N)*, bei der die Vektoren der Wörter im Array *positive* aufaddiert und die Vektoren der Wörter in *negative* davon subtrahiert werden. Die Funktion liefert die N Wörter zurück, deren Vektordarstellungen eine maximale Kosinusähnlichkeit mit dem Ergebnis der Vektoraddition und -subtraktion haben. Diese Funktion wird auch in der Evaluationsfunktion benutzt, indem die ersten drei Wörter des 4-Tupels in *positive* und *negative* eingeteilt werden[9], wird das vierte Wort, hier ‚Germany‘, korrekt vom Model vorhergesagt, wird dieser Testdatensatz als erfolgreich gezählt. Sollten Wörter nicht im Modell dargestellt sein, wirkt sich dies nicht negativ auf die Accuracy aus, denn die Gesamtaccuracy eines Modells wird als Verhältnis, zwischen

[4] https://code.google.com/p/word2vec/#Performance, abgerufen am 01.07.2015

[5] http://mattmahoney.net/dc/enwik9.zip, abgerufen am 30.06.2015, beinhaltet die ersten 1 Milliarde Zeichen des Gesamtkorpus

[6] *gensim.models.word2vec.Word2Vec.accuracy(FILENAME)*

[7] https://code.google.com/p/word2vec/source/browse/trunk/questions-words.txt, abgerufen am 29.06.2015

[8] *capital-common-countries, capital-world, currency, city-in-state, family, gram1-adjective-to-adverb, gram2-opposite, gram3-comparative, gram4-superlative, gram5-present-participle, gram6-nationality-adjective, gram7-past-tense, gram8-plural, gram9-plural-verbs*

[9] *most_similar(positive=[‘Greece‘,‘Berlin‘], negative=[‘Athens ‘], topn=1)*

der Anzahl der richtig vorhergesagten Wörter und der Anzahl der Testdatensätze, bei denen alle Wörter eine Vektordarstellung im Modell haben, dargestellt. In der Tabelle 4.1 werden die unterschiedlichen Parameter[10] und die erreichte Gesamtaccuracy dargestellt. In diesem Modell waren nur bei 13144 der 19544 4-Tupel alle Wörter als Vektordarstellungen im Modell vorhanden.

Tabelle 4.1: Vergleich Parameter

Size	**Window**	**Min_count**	**Gesamtaccuracy**
400	10	5	53,5% (7027/13144)
400	10	10	52,5% (6905/13144)
300	10	5	52,5% (6860/13144)
300	10	10	52,9% (6951/13144)
200	10	5	50,5% (6632/13144)
200	10	10	50,5% (6636/13144)
100	10	5	42,0% (5517/13144)
100	10	10	41,3% (5431/13144)

Auf dem kleineren Wikipedia Korpus erzielte das Modell mit den Parametern *size* = 400, *window* = 10, *min_count* = 5 die beste Gesamtaccuracy mit 53,5%. Diese Parameter wurden dann auch beim Gesamtmodell angewandt. Die Trainingszeit beträgt bei diesen Parametern 10,5h[11] beim Gesamtmodell.
Nachdem ein kleinerer Fehler im Reiningungsskript aufgetretetn war[12], musste das komplette Modell nochmals gelernt werden. Hier fiel dann die Entscheidung, andere Parameter zu benutzen und die *window* Größe auf 300 zu reduzieren, da die Accuracy sich nur sehr gering verändert hat, allerdings verkürzte sich die Trainingszeit auf 7,7h. Dies zahlte sich im Laufe der Arbeit aus, da sich das Perl-Reinigungsskript ein paar Mal geringfügig änderte um so kleinere Fehler aus dem Korpus zu entfernen.

4.2 Teilkorpus

Der zweite Korpus ist ein Teilkorpus des Gesamtkorpus, bestehend aus Technologieartikeln. Um diesen Korpus zu erstellen, muss der Gesamtkorpus, wie in 2.3 beschrieben, zuerst in die einzelnen Wikipediaartikel unterteilt werden. Diese Arti-

[10] Siehe 3.5. Auf der Google Code Seite von Word2Vec wird eine window size bei der Skip-gram Architektur von 10 vorgeschlagen, somit wird sie hier nicht verändert.
[11] Leistungsdaten: PC mit 32 GB RAM, i7-3770 Quadcore bei 3,4 GHz
[12] Unbekannte Zeichen wurden mit einem Leerzeichen ersetzt, anstatt gelöscht zu werden, somit entstanden Wortfragmente als eigene Worte.

kel werden dann mithilfe eines NBC klassifiziert. Ein NBC basiert auf dem Prinzip des überwachten Lernens. Hierzu müssen vor dem Training von Hand ausgewählte Artikel[13] in die fünf zu klassifizierenden Themenbereiche[14] eingeteilt werden. In jeder Kategorie wurden gleich viele Artikel ausgewählt, somit entsteht kein Ungleichgewicht beim Training. Es wurden je Kategorie 100 Artikel verwendet. Mit diesen Daten kann der NBC dann trainiert werden. Im Anschluss daran, kann der NBC dann für die Klassifikation der Wikipediaartikel verwendet werden. Die Artikeln, die in die Kategorie *tech* eingeteilt werden, bilden dann den technologiespezifischen Teilkorpus, der als zweiter Korpus in dieser Arbeit verwendet wird.

Dieser domänenspezifische Teilkorpus enthält 187.144 Artikel (2,2% im Vergleich zum Gesamtkorpus), 9.866.096 Wörter(0,34%) und 3.166.065 Sätze(1,3%). Hier beträgt die Trainingszeit ca. 2 Minuten.
Wird dieser Korpus mit der *.accuracy()*-Methode evaluiert, erreicht er eine Gesamtaccuracy von nur 7,0%. Das hat den Grund, dass die Testfragen sehr allgemeiner Art sind und diese Beziehungen in einem reinen Technologiekorpus sehr selten bis gar nicht vorkommen.

Es wäre auch möglich gewesen, andere Daten zu benutzen. Beispielsweise könnten Korpora aus Foren zu technologischen Themen oder Technews Internetseiten, automatisch mittels Crawling, erstellt werden. Die Wahl fiel aber auf den Teilkorpus von Wikipedia, da hier die gleichen Texte in beiden Korpora vorhanden sind und somit auch die gleichen Beziehungen zwischen den einzelnen Wörtern, um so einen möglichst objektiven Vergleich zwischen den unterschiedlichen Korpora zu ermöglichen.

4.3 Testdaten

Im folgenden Kapitel sollen die beiden Word2Vec Modelle, deren Trainingsdaten aus dem Gesamtkorpus beziehungsweise dem Technologiekorpus bestehen, miteinander verglichen werden. Für diesen Vergleich werden einige Experimente durchgeführt (siehe 5.1 - 5.5). In diesen Experimenten werden die semantischen Beziehungen zwischen Wörtern betrachtet. Um diese Experimente durchführen zu können, werden Testdaten benötigt.

[13] Siehe Anhang CD-Rom!
[14] tech, entertainment, sport, science, politic

Die Testdaten umfassen 236 Begriffe (siehe Anhang 7.1) aus der Domäne Technologie/Internet. Die Testdaten werden dazu verwendet, um die ähnlichen Worte in den Modellen zu untersuchen und zu vergleichen. Die genauen Fragestellungen und Experimente werden im Kapitel 5 Experimente ausführlich beschrieben.

Kapitel 5

Experimente

In diesem Kapitel sollen die unterschiedlichen Korpora (Gesamtkorpus[1] und Techkorpus[2]) untersucht werden. Dies soll durch ausgewählte Fragestellungen realisiert werden.
Die Fragestellungen beziehen sich immer auf die Ergebnisse, die aus den Tastdaten[3] erhaltenen ähnlichen Worten.
Jedes Experiment ist in drei Teile aufgeteilt Beschreibung, Durchführung und Interpretation/Ergebnis.
Zum Vergleichen der ähnlichen Worte der unterschiedlichen Korpora wurde eine Datei erzeugt, in der die Testdaten und ihre fünf ähnlichsten Worte sowie deren fünf ähnlichsten Worte (Rekursion), leserlich formatiert, ausgegeben werden.

5.1 Synonymsuche durch Rekursion

5.1.1 Beschreibung

Es soll untersucht werden, ob es möglich ist, Synonyme eines Begriffs zu finden, indem man die vom Modell erhaltenen ähnlichen Worte dieses Begriffs erneut als Input(Suchbegriffe) der Methode *most_similar()* verwendet. Also durch einmalige Rekursion.
Eine Anwendung hierfür könnte ein Thesaurus-Lexikon sein.

[1] vgl. 4.1
[2] vgl. 4.2
[3] vgl. 7.1

5.1.2 Durchführung

Oft war der Suchbegriff aus den Testdaten mit in den ähnlichen Wörtern der ähnlichen Wörter. Da in diesem Experiment pro Suchbegriff 25 Ergebnisbegriffe untersucht werden mussten, nahm es etwas mehr Zeit in Anspruch als die übrigen Experimente. Zudem waren viele Begriffe sehr themenspezifisch, deren Zusammenhang mit dem Suchbegriff aus den Testdaten erst einmal herausgefunden werden musste.
Im Technologiekorpus waren einige Suchbegriffe nicht enthalten.
Sobald ein Synonym vorhanden war, wurde der Suchbegriff als *Synonym enthalten* markiert.

Tabelle 5.1: Experiment 1: Synonyme durch Rekursion

Korpus	Synonyme nicht enthalten	Synonyme enthalten	Wort nicht im Korpus	Relation
Komplett	187	49	0	20,8%
Technologie	191	20	25	8,5% (9,5%)

Die Prozentangabe in Klammern bezieht sich auf die Relation zu den im Korpus enthaltenen Worte (211 Stück) und nicht auf die Gesamtzahl an Testdaten(236 Stück).

Einige Beispiele:

Tabelle 5.2: Beispiele von Synonyme durch Rekursion

Suchbegriff	Korpus	Synonyme
3d	Komplett	threedimensional, stereoscopic, autostereoscopic
	Technologie	threedimensional, autostereoscopic
bitcoin	Komplett	cryptocurrencies
hd	Komplett	1080p, 1920x1080
	Technologie	1080p, 1920x1080
keyboard	Komplett	synthesizer
wifi	Komplett	wlan

5.1.3 Interpretation/Ergebnis

Wie aus den Daten hervorgeht, werden zwar einige Synonyme der Suchbegriffe gefunden, allerdings nicht in großen Mengen. Dazu kommt auch, dass sich die beiden Korpora nochmal unterscheiden. Im kompletten Korpus wurden in *20,8%* der Testfälle Synonyme gefunden, wobei es im Technologiekorpus sogar nur weniger als halb so viele (*8,5%*) waren.

Somit ist die Vermutung dieses Experiments widerlegt.

5.2 Konkretisierungen

5.2.1 Beschreibung

In diesem Experiment ist zu untersuchen ob die ähnlichen Worte im domänenspezifischen Korpus eine Konkretisierung des Suchbegriffs darstellen.
Dies ist gerade bei mehrdeutigen Worten interessant, da hier dann klar eine Definition dominant sein könnte und somit klar ist, in welchem Kontext das mehrdeutige Wort in dieser Domäne gemeint ist.

5.2.2 Durchführung

Die Fragestellung bezieht sich zwar nur auf den domänenspezifischen Korpus, allerdings wurde auch der allgemeine Korpus auf diese Fragestellung hin untersucht.
Auch bei diesem Experiment nahm das Nachschlagen von mehreren ähnlichen Worten einige Zeit in Anspruch. Denn um beurteilen zu können, ob es sich um eine Konkretisierung handelt, müssen die inhaltlichen Beziehungen zwischen Suchbegriff und ähnlichen Worten klar sein.

Tabelle 5.3: Experiment 2: Konkretisierung

Korpus	ähnliche Worte nicht konkretisiert	ähnliche Worte konkretisiert	Wort nicht im Korpus	Relation der konkretisierten Worte
Komplett	199	37	0	15,7%
Technologie	146	65	25	27,5% (30,8%)

Die Prozentangabe in Klammern bezieht sich auf die Relation zu den im Korpus enthaltenen Worte (211 Stück) und nicht auf die Gesamtzahl an Testdaten(236 Stück).

Tabelle 5.4: Beispiele zur Konkretisierung

Suchbegriff	Korpus	Konkretisierung
apple	Technologie	iphone (Kosinusähnlichkeit: 0,587),
architecture	Komplett Technologie	revival (0,774), neoclassical (0,677), gothic (0,667) armv7a (0,514), multicore (0,493), xmos (0,477)
fat	Technologie	exfat (0,751), vfat (0,724), fat32 (0,709)
intel	Technologie	i7 (0,707), t2300 (0,702), i5 (0,666), i52450m (0,663)
photography	Technologie	slrs(0,544), panoramic (0,543), polaroid (0,531)
power	Komplett	electricity (0,679), surgut2 (0,637), egbin (0,627)

5.2.3 Interpretation/Ergebnis

Zu beobachten ist, dass der Technologiekorpus besser konkretisiert, als der komplette allgemeine Korpus. Da dies aber nur in ca. 30% der Testfälle passiert, ist es schwierig, zu verallgemeinern, dass die ähnlichen Worte im Technologiekorpus eine Konkretisierung sind, somit ist auch hier die Annahme des Experiments widerlegt.

5.3 Verallgemeinerungen

5.3.1 Beschreibung

In diesem Experiment soll untersucht werden, ob die ähnlichen Worte im allgemeinen kompletten Korpus eine Verallgemeinerung des Suchbegriffs darstellen.

5.3.2 Durchführung

Die Fragestellung bezieht sich zwar nur auf den allgemeinen Korpus, allerdings wurde auch der Technologiekorpus auf diese Fragestellung hin untersucht.
Wie beim vorherigen Experiment mussten viele Begriffe nachgeschlagen werden, um die inhaltlichen Beziehungen zwischen Suchbegriff und ähnlichen Worten beurteilen zu können.

Tabelle 5.5: Experiment 3: Verallgemeinerung

Korpus	ähnliche Worte nicht verallgemeinert	ähnliche Worte verallgemeinert	Wort nicht im Korpus	Relation der verallgemeinerten Worte
Komplett	102	134	0	56,8%
Technologie	144	67	25	28,4% (31,8%)

Die Prozentangabe in Klammern bezieht sich auf die Relation zu den im Korpus enthaltenen Worte (211 Stück) und nicht auf die Gesamtzahl an Testdaten(236 Stück).

Tabelle 5.6: Beispiele zur Verallgemeinerung

Suchbegriff	Korpus	Verallgemeinerung
apple	Komplett	blackberry (Kosinusähnlichkeit: 0,704), raspberry (0,657), webos (0,643)
asus	Komplett	netbook (0,767), lenovo (0,736), motherboards (0,714)
cookies	Komplett	baked (0,666), cakes (0,663), sandwiches (0,655), muffins (0,654)
dell	Komplett	hewlettpackard (0,583), cisco (0,548), lenovo (0,515), compaq (0,506)
limewire	Komplett	bittorrent (0,703), filesharing (0,702), kazaa (0,689) rapidshare (0,668), gnutella (0,653)
security	Komplett	cybersecurity (0,702), counterterrorism (0,685), iviz (0,668)

5.3.3 Interpretation/Ergebnis

Die Untersuchungen bei diesem Experiment zeigen, dass die ähnlichen Worte eines Suchbegriffs in mehr als der Hälfte der Fälle, eine Verallgemeinerung des Suchbegriffs sind.
Die Annahme dieses Experiments wurde bestätigt.

5.4 Unterschiedliche Beziehungen

5.4.1 Beschreibung

In diesem Experiment soll untersucht werden, ob die Beziehungen zwischen den Suchbegriffen und den gefunden ähnlichen Worten von unterschiedlichen Arten sind. Die Beziehungen sollen in *syntaktisch, synonym, gleiches, gegenteiliges, in Beziehung* und *verschiedenes* eingeteilt werden.

5.4.2 Durchführung

Für dieses Experiment müssen beide Korpora untersucht werden, um eine Aussage machen zu können, ob sich die Beziehungsarten unterscheiden. Die Arten *synonym* und *gegenteilig* traten, bei den in der Arbeit verwendeten Testdaten, nicht auf. Die Art *verschiedenes* wurde gewählt, wenn mehrere andere Beziehungsarten in den ähnlichen Worten vorhanden waren.

Tabelle 5.7: Experiment 4: Arten von Beziehungen

Beziehungsart	Kompletter Korpus	Relation	Technologie-Korpus	Relation
Syntaktisch	1	0,4%	0	0,0%
Gegenteilig	0	0,0%	0	0,0%
Gleiches	81	34,3%	38	16,1% (18,0%)
Synonym	0	0,0%	0	0,0%
In Beziehung	118	50,0%	126	53,4% (59,7%)
Verschiedenes	36	15,3%	47	19,9%(22,3%)

Die Prozentangabe in Klammern bezieht sich auf die Relation zu den im Korpus enthaltenen Worte (211 Stück) und nicht auf die Gesamtzahl an Testdaten(236 Stück).

Tabelle 5.8: Beispiele zur den verschiedenen Beziehungsarten

Art	Suchbegriff	Korpus	Ähnliche Worte
syntaktisch	encription	Komplett	encrypted (Kosinusähnlichkeit: 0,833), encrypts (0,798)
gleiches	chrome	Komplett Technologie	firefox (0,784), safari (0,686), ie8 (0,674) firefox (0,733), safari (0,769), flock (0,667), seamonkey (0,650)
in Beziehung	browser	Komplett	browssers (0,882), firefox (0,812), desktop (0,755), javascript (0,751)
verschiedenes	itunes	Komplett	spotify (0,770), beatport (0,756), preorder (0,718), download (0,694)

5.4.3 Interpretation/Ergebnis

Wie man in der Auswertung der Daten erkennt, ist bei beiden Korpora die Beziehungsart *in Beziehung* am stärksten vorhanden, dies kommt daher, dass im Word2Vec Modell Wörter die in ähnlichem Kontext stehen, auch nahe in der Vektorrepräsentation zusammen stehen. Die Art *gleiches* ist im kompletten allgemeinen Korpus mit 34,3% die zweithäufigste Art, allerdings kommt sie im Technologiekorpus nur ca. halb so oft vor (19,9%). Hingegen ist die Anzahl mit *verschiedenen* Arten etwas höher, sowie die *in Beziehung*. Der Unterschied zwischen den Beziehungsarten in den unterschiedlichen Korpora ist also nicht zu groß, allerdings deutlich erkennbar.

5.5 Erkennen von Mehrdeutigkeit

5.5.1 Beschreibung

Das letzte Experiment beschäftigt sich mit der Mehrdeutigkeit von Worten. Es soll untersucht werden, ob im allgemeinen kompletten Korpus die unterschiedlichen Bedeutungen eines mehrdeutigen Wortes in den ähnlichen Worten repräsentiert werden.

5.5.2 Durchführung

Auch hier wurden beide Korpora analysiert um zu vergleichen ob sie sich unterscheiden im Bezug auf mehrdeutige Worte.

Tabelle 5.9: Experiment 5: Erkennen von Mehrdeutigkeiten

	Kompletter Korpus	Relation zur Gesamtzahl	Technologie-Korpus	Relation zur Gesamtzahl
Wort nicht im Korpus	0	0,0%	24	10,2%
nicht mehrdeutig	172	72,9%	148	62,7% (70,1%)
mehrdeutig	64	27,1%	64	27,1% (30,3%)
	Kompletter Korpus	Relation zu mehrdeutigen Worten	Technologie-Korpus	Relation zu mehrdeutigen Worten
Wort nicht im Korpus	0	0,0%	1	1,7%
nicht erkannt	58	90,6%	57	89,1%
erkannt	6	9,4%	6	9,4%

Die Prozentangabe in Klammern bezieht sich auf die Relation zu den im Korpus enthaltenen Worte (211 Stück) und nicht auf die Gesamtzahl an Testdaten(236 Stück).

Tabelle 5.10: Beispiele zur Erkennung von Mehrdeutigkeiten

Suchbegriff	Korpus	Erkannt	Ähnliche Worte
trojan	Komplett	ja	rsplug (Kosinusähnlichkeit: 0,472), athena (0,462)
	Technologie	ja	bundestrojaner (0,496), centaurs (0,475), horse (0,586), malware (0,560), zinaps (0,476)
raspberry	Komplett	nein	apple (0,657), apricot (0,629),
	Technologie	nein	blackcurrant (0,586), pi (0,677), cubieboard (0,524), trimeslice (0,519)

5.5.3 Interpretation/Ergebnis

Erstaunlicherweise unterscheiden sich in diesem Experiment die beiden Korpora nicht in der Anzahl der erkannten Mehrdeutigkeiten. Allerdings hat öfters das eine Modell eine Bedeutung abgedeckt und das andere Modell eine oder die andere.
Es ist deutlich zu erkennen, dass beide Modelle nicht geeignet sind um eine Mehrdeutigkeit in den Testdaten in ihren ähnlichen Worten abzubilden.
Somit ist die Annahme dieses Experiments widerlegt.

Kapitel 6

Fazit und Ausblick

6.1 Fazit

Nach einer ausführlichen Vorverarbeitung können die Artikel aus Wikipedia gut als Trainingsdaten für Word2Vec Modelle dienen. Dank der Gensim-Implementierung ist die Benutzung von Word2Vec sehr vereinfacht und kann komfortabel bedient werden.
Die Experimente liefern interessante Resultate im Bezug auf die Problemstellung zu Anfang. Durch die Experimente können die unterschiedlichen semantischen Beziehungen etwas aufgeschlüsselt werden.

So wird durch die Experimente klar, dass durch einmalige Rekursion der Suchbegriffe, nicht auf Synonyme des ursprünglichen Begriffs geschlossen werden kann.
Des weiteren eignet sich der domänenspezifische Teilkorpus eher für eine Konkretisierung der Suchbegriffe, wobei dies nur in ca. einem Drittel der Testdaten nachgewiesen werden konnte.
Sollen die Testdaten verallgemeinert werden, eignet sich der allgemeine komplette Korpus viel besser als der domänenspezifische Teilkorpus. Soll also ein Überblick über ein gewisses Thema oder Suchbegriff erhalten werden, wird empfohlen das Modell, welches auf dem allgemeinen kompletten Korpus trainiert wurde, zu verwenden.
Die unterschiedlichen Arten von Beziehung zwischen den Testdaten und ihren ähnlichen Wörtern unterscheiden sich nicht grundlegend zwischen den beiden Modellen. Das allgemeine komplette Modell hat einen etwas Größeren Anteil an *gleichen* Worten, hingegen hat das domänenspezifische Teilmodell einen etwas größeren Anteil an *verschiedenen* und *in Beziehung* stehenden Beziehungen. *Synonyme* oder *gegenteilige* Arten an Beziehungen sind in beiden Modellen nicht vorhanden. Und auch *syntaktische* Beziehungen sind nur bei einem Testdatensatz vorhanden.

Soll eine Mehrdeutigkeit gefunden werden, ist die Vorgehensweise nur die ähnlichen Worte zu vergleichen und zu analysieren, nicht zielführend. Die Modelle bilden die Mehrdeutigkeit in den ähnlichen Worten nicht gut ab. Im Ausblick wird ein Vorschlag gemacht, wie die Mehrdeutigkeit eventuell besser erfasst werden könnte.

Zusammenfassend kann gesagt werden, dass sich das allgemeine komplette Modell dann besser eignet, wenn ein genereller Überblick über die Testdaten erhalten werden soll. Denn in diesem Modell werden die Suchbegriffe eher verallgemeinert und die ähnlichen Worte repräsentieren *Gleiches* im Bezug zu den Testdaten.
Soll aber ein Suchbegriff genauer untersucht, bzw. sehr fokussiert betrachtet werden, eignet sich ein domänenspezifisches Modell besser. Es konkretisiert die Testdaten nicht nur besser, sondern liefert auch fachspezifischere ähnliche Worte.

Die Qualität der Beziehungen hängt auch stark von den verwendeten Trainingsdaten ab. Es sollte gewährleistet sein, dass ausreichend viele und auch qualitativ gute Daten zum Training verwendet werden.
Soll ein domänenspezifisches Modell erstellt werden, sollte hier auch darauf geachtet werden, dass nicht nur ein Teil dieser Domäne in den Trainingsdaten abgebildet ist, außer es soll nur dieser spezielle Teil der Domäne im Modell dargestellt werden.

6.2 Ausblick

In der vorangegangenen Arbeit wurden einige Beziehungen zwischen den Testdaten und ihren ähnlichen Worten analysiert. In weiterführenden Arbeiten wäre es möglich, weitere und andersartige Beziehungen zwischen den Wörtern zu untersuchen. Also einfach noch mehrere Experimente durchzuführen.
Zum Beispiel könnte die Mehrdeutigkeit noch anders untersucht werden, indem man mehrere unterschiedliche Modelle hat und dann die ähnlichen Worte eines gleichen Testwortes vergleicht.

Auch könnte analysiert werden wie sich subsampling[1] im Word2Vec Modell, im Blick auf die in dieser Arbeit analysierten Fragestellungen, verhält.
Es wäre auch möglich das Word2Vec Modell mit N-Grammen, auch Phrases genannt, zu lernen, dann könnten auch Mehrwortbegriffe abgebildet und gesucht werden.

[1] Beim subsampling wird das Ungleichgewicht von sehr häufig und sehr selten vorkommenden Worten im Trainingskorpus verringert[MSC+13].

Literaturverzeichnis

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[MSC+13] 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.

[MYZ13] 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.

[ŘS10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 7

Anhang

7.1 Testdaten

Tabelle 7.1: Testdaten Teil 1

3d	3ds	3g	4chan
4g	acer	acta	activision
adobe	amazon	android	anonymous
aol	apple	app	augmented
arcade	architecture	arpanet	asus
auto	automobile	battlefield	bing
biometrics	bitcoin	bittorrent	blackberry
blizzard	blogging	blog	bluray
broadband	browser	casual	chatroulette
chrome	chromebook	cispa	computing
console	cookies	craigslist	crowdfunding
crowdsourcing	cryptocurrency	cybercrime	cyberwar
darknet	data	dell	diablo
doodle	dotcom	drone	dropbox
e3	ebay	email	emoji
encryption	energy	engine	engineering
ereader	events	facebook	fat
filesharing	firefox	flickr	foursquare
gadget	game	gameplay	gamergate

Tabelle 7.2: Testdaten Teil 2

games	gaming	ghz	gmail
google	googlemail	gps	groupon
gta	hacking	halo	handheld
hardware	hashtag	hd	heartbleed
htc	html5	i	ibm
icloud	ie	imac	indie
instagram	intel	internet	ios
ipad	iphone	ipod	isp
itunes	keyboard	kickstarter	kindle
kinect	laptop	lenovo	lg
limewire	link	linkedin	linux
live	machinima	macintosh	macworld
malware	mario	megaupload	microsoft
minecraft	mmorpg	mobile	monitor
motoring	mouse	mozilla	myspace
nes	net	netbook	nfs
nintendo	nokia	oracle	ouya
p2p	paypal	pc	phablet
phishing	photography	photoshop	pi
pinterest	piracy	pirate	platform
playback	playstation	pokemon	power
processor	programming	ps	ps2
ps3	ps4	psp	python
raider	ram	raspberry	rayman
recommendation	reddit	retro	robot
rpg	rts	safari	samsung
search	security	seo	skype
smartphone	smartphones	smartwatch	smartwatches
software	sonic	sony	sopa
spam	spotify	steam	stream
starcraft	stuxnet	sun	surface
symbian	tablet	technology	technophile
ted	telecom	television	tetris
titanfall	tomb	trojan	tumblr
twitch	twitter	viber	vine
virus	warcraft	web	whatsapp
wheel	wifi	wii	wikipedia
windows	windows7	wireless	worms
wow	xbox	xp	y2k
yahoo	youtube	zelda	zynga

7.2 Reinigungsskript

Das Originalskript von Matt Mahoney kann unter http://mattmahoney.net/dc/textdata.html gefunden werden.

```
#!/usr/bin/perl

# Program to filter Wikipedia XML dumps to "clean" text consisting only
# of lowercase letters (a-z, converted from A-Z), and spaces
# (never consecutive).
# All other characters are converted to spaces.  Only text which normally
# appears in the web browser is displayed.  Tables are removed.
# Image captions are
# preserved.  Links are converted to normal text.
# Digits are spelled out.

# Written by Matt Mahoney, June 10, 2006.  This program is
# released to the public domain.

$/=">";                     # input record separator
while (<>) {
  if (/<text /) {$text=1;}  # remove all but between <text> ... </text>
  if (/#redirect/i) {$text=0;}  # remove #REDIRECT
  if ($text) {

    # Remove any text not normally visible
    if (/<\/text>/) {$text=0;}
    s/<.*>//;               # remove xml tags
    s/&amp;/&/g;            # decode URL encoded chars
    s/&lt;/</g;
    s/&gt;/>/g;
    s/<ref[^<]*<\/ref>//g;  # remove references <ref...> ... </ref>
    s/<[^>]*>//g;           # remove xhtml tags
    s/\[http:[^] ]*/[/g;    # remove normal url, preserve visible text
    s/\|thumb//ig;          # remove images links, preserve caption
    s/\|left//ig;
```

```
    s/\|right//ig;
    s/\|\d+px//ig;
    s/\[\[image:[^\[\]]*\|//ig;
    # show categories without markup
    s/\[\[category:([^|\]]*)[^]]*\]\]/[[$1]]/ig;
    s/\[\[[a-z\-]*:[^\]]*\]\]//g;  # remove links to other languages
    s/\[\[[^\|\]]*\|/[[/g;  # remove wiki url, preserve visible text
    s/{{[^}]*}}//g;         # remove {{icons}} and {tables}
    s/{[^}]*}//g;
    s/\[//g;                # remove [ and ]
    s/\]//g;
    s/&[^;]*;/ /g;          # remove URL encoded chars

    $_=" $_ ";
    ###### begin changed lines ######
    s/Ä/Ae/g;
    s/ä/ae/g;
    s/Ö/Oe/g;
    s/ö/oe/g;
    s/Ü/Ue/g;
    s/ü/ue/g;
    s/ß/ss/g;
    s/-//g;
    #removes everything else than this characters
    tr/0-9A-Za-z,.!?;\r\n / /csd;
    ###### end changed lines ######
    chop;
    print $_;
  }
}
```