Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Imon Bashir

Entwicklung eines Ähnlichkeitsmaßes für
natürlichsprachige Texte

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Imon Bashir

Entwicklung eines Ähnlichkeitsmaßes für natürlichsprachige Texte

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Wirtschaftsinformatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Clemen
Zweitgutachter: Prof. Dr. Tropmann-Frick

Abgegeben am 28.01.2019

Imon Bashir

Thema der Arbeit

Entwicklung eines Ähnlichkeitsmaßes für natürlichsprachige Texte

Stichworte

Natürliche Sprachverarbeitung, Semantik, Ähnlichkeit von Texten

Kurzzusammenfassung

Diese Bachelorarbeit befasst sich damit, beurteilen zu können, inwieweit zwei Texte als ähnlich betrachtet werden können. Dabei werden zunächst die theoretischen Grundlagen sowie Methoden präsentiert und anschließend ein System vorgestellt, welches die Fähigkeit von verschiedenen Modellen, Textähnlichkeit zu bestimmen, bewerten soll. Abschließend werden die einzelnen Modelle miteinander verglichen.

Imon Bashir

Title of the paper

Development of a similarity measure for natural language texts

Keywords

Natural Language Processing, NLP, semantics, text similarity

Abstract

This bachelor thesis deals with the question of how two texts can be regarded as similar. First, theoretical concepts and methods are presented and then a system is introduced to evaluate the capability of different models to determine text similarity. Finally, the respective models are compared with each other.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Einige soziale Netzwerke bieten neben ihren ursprünglichen Funktionen mittlerweile weitere Features an. Zu diesen Features gehört die News-Sektion, in der ein User nicht nur klassische Nachrichten aus dem Alltag, sondern auch Branchennachrichten empfohlen bekommt, die auf seine Interessen zugeschnitten sind. So auch bei *XING*, dem zweitgrößten (nach LinkedIn) Karrierenetzwerk der Welt mit Sitz in Hamburg, in dessen Kooperation diese Bachelorarbeit ursprünglich entstand.

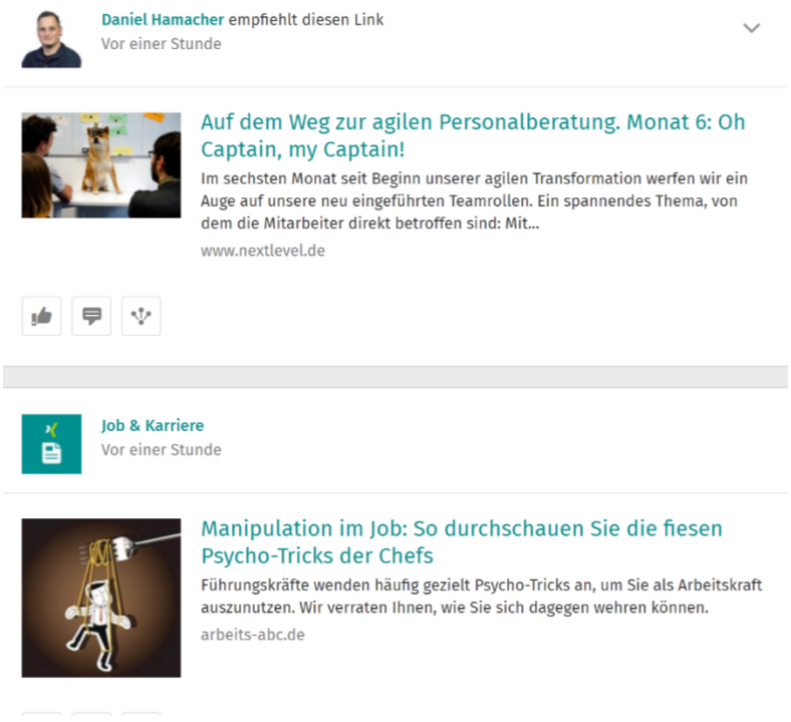


Abbildung 1: Ausschnitt aus einer beispielhaften *XING*-Startseite

Auf der Startseite eines *XING*-Users erscheinen Nachrichtenartikel aus diversen Quellen: Links von Webseiten, geteilte Artikel von Freunden, abonnierte Artikel.

Empfiehlt ein Kontakt einen Artikel, erscheint dieser hier. Ebenso teilt *XING* automatisch Artikel, sofern man das entsprechende Themengebiet – in Abbildung 1 ist es „Job & Karriere“ – abonniert.
Diese Themengebiete gehören zu den kuratierten *XING-News*: Hier werden Artikel in Kooperation mit deutschen Verlagen und Zeitschriften auf *XING* geteilt.
Jedoch hat jeder Kontakt zusätzlich die Möglichkeit, eigene Links zu teilen (Abbildung 1). Hier kommt es nun zu einem Konflikt: Wie ist zu verfahren, wenn ein News-Artikel auf der Startseite eines Users platziert werden soll, aber ein Freund ebendieses Users zufälligerweise diesen Artikel geteilt hat?
Natürlich könnte *XING* dies detektieren. Aus dem Link können diese Information leicht gewonnen werden. Selbst wenn die Links, was oft aus Werbe- oder Abkürzungsgrunden der Fall ist, unterschiedlich sein sollten, wäre ein Abgleich des Textes ein Leichtes. Ein solcher Mechanismus ist bereits im Einsatz.
Spannender ist nun ein Szenario, in dem ebenjener Kontakt eines Nutzers einen Artikel zum selben Thema teilt, zu dem diesem User bereits andere Artikel zur Verfügung stehen. Grundsätzlich klingt das nicht problematisch, weil es sich hier auf den ersten Blick um zwei verschiedene Texte von verschiedenen Verlagen handelt, die wegen verschiedener

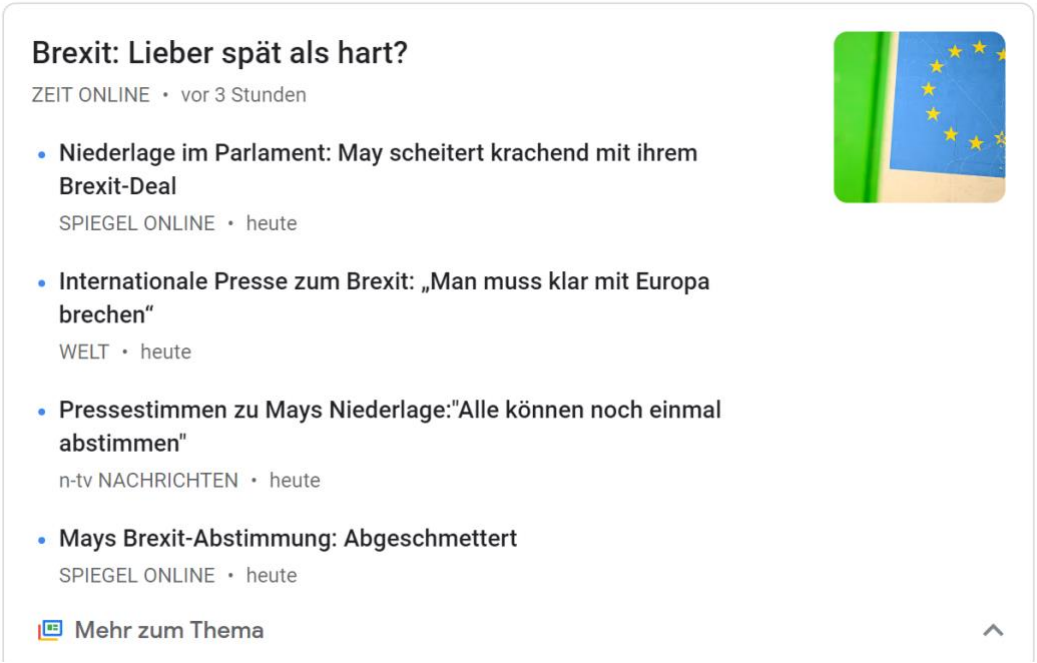


Abbildung 2: Auf *Google-News* werden zusammengehörige Artikel zusammengefasst

Ereignisse Teil der Startseite eines Users sind. Der eine Artikel könnte von einem Freund geteilt, der andere automatisch von *XING* auf der Startseite platziert worden sein. Anderer Artikel, anderer Inhalt. Die Betrachtung eines Beispiels verdeutlich jedoch die dahinterliegende Schwierigkeit, denn die in Abbildung 2 zu sehenden fünf Artikel zum Thema „Großbritanniens Ausstieg aus der Europäischen Union“, kurz *Brexit*, allesamt am selben Tag publiziert, bieten einen Blick auf die Terminologie der *unterschiedlichen* Artikel, die sich offensichtlich nicht aus der Überschrift alleine ableiten lässt.

Google kennt dieses Problem aus seiner News-Sektion, in dem zu jedem Thema mehrere konkurrierende Artikel existieren und tut hier etwas, das dem User zunächst intuitiv erscheint: Es fasst Artikel in klaren Gruppen zusammen, in diesem Fall zum Themenfeld *Brexit*. *Google* gewährt jedem Themengebiet je ein solches, graues Rechteck wie in Abbildung 2, unabhängig davon, wie viele Veröffentlichungen sich in diesem Kasten befinden. Während die Überschriften, die Uhrzeiten und die Art der Berichterstattung je nach Autor und Magazin variieren, sind die Kerninformationen in allen Artikeln dieselben: Ein neuer Versuch einer Vorlage zur Durchsetzung der Abspaltung Großbritanniens aus der Europäischen Union wurde durch das britische Parlament abgelehnt.
Durch die Nutzung der Kästen pro Thema spart *Google* effektiv Platz auf der Seite, was ermöglicht, dass der User mehr Inhalte auf selber Fläche sieht.

So etwas ist in der Nachrichtenwelt gewöhnlich. Jedes Blatt schreibt seinen eigenen Artikel zum selben Geschehen, meist gespeist durch Informationen der deutschen Presseagentur (dpa). Und während *Google* bereits entsprechende Verfahren implementiert hat, fehlt es *XING* an solch einem Mechanismus.
Das Problem war für *XING* nie akut, weil die News kuratiert sind. Doch ein Wandel des Userverhaltens sorgt dafür, dass die eigens geteilten und aus Fremdquellen importieren Nachrichten zusehends gewichtiger und damit zu einem Problem werden.

Werden nun solche thematisch verwandten Artikel auf die Newsseite desselben Users platziert, so führt dies zu mehreren Konflikten:

• Aus Sicht des Seitenbetreibers: „Wir verschwenden Platz, der für einen anderen Artikel besser genutzt werden könnte. Das für uns Sinnvollste wäre, diese Artikel innerhalb des Webseiten-Interfaces gruppieren. So sparen wir Platz und der User sieht mehr Inhalte pro Pixel.“

• Aus User-Sicht: „Ich erhalte denselben Artikel, den ich weiter oben in meiner App bereits gesehen habe, noch einmal. Warum können sie nicht übereinanderstehen."

Dies könnte mittelfristig dazu führen, dass der User *XING* oder zumindest den Newsbereich meidet. Dies würde auch dazu führen, dass andere Meldungen – denn die *XING-News* sind nicht der einzige Inhalt der Startseite – vom User von hieran ignoriert werden. Da der Kern eines solchen sozialen Netzwerkes in der Gänze der Interaktionen von Usern miteinander liegt, in welche gezielt Werbung oder andere Inhalte eingefügt werden, steht ein solches Szenario diametral zu den Unternehmenszielen *XING*s. Außerdem verbraucht der Betreiber unnötig Ressourcen für redundante Informationen.

Lösungen für die Feststellung der Ähnlichkeit solcher Texte sollen im Rahmen dieser Arbeit untersucht werden.

1.2 Zielsetzung

Es wird ein Werkzeug benötigt, dass Texten die nötigen Informationen entnimmt, aus denen ihre Ähnlichkeit mit einem Wert bestimmt werden soll. Dieser Wert soll von einer Natur sein, aus der beispielweise eine Gruppierung wie bei den *Google-News* im Anschluss möglich wäre. Darüber hinaus müsste für diese bestimmte Ähnlichkeit ein Schwellwert existieren, ab dem von grundsätzlich nicht nur themenverwandten, sondern wirklich demselben Ereignis zugehörigen Artikeln gesprochen werden kann.

Zunächst wird in dieser Arbeit zusammengefasst, auf welcher Grundlage überhaupt natürlichsprachige Texte durch Modelle als ähnlich bewertet werden können. Hierbei geht es nicht nur um die Ebene des Wortes; dass *Großbritannien* in dem einen Text vorzugsweise *Vereinigtes Königreich*, in einem anderen *England* et cetera genannt wird, wäre für solche Wortgleichheiten ein einfaches Beispiel.[1]

Vielmehr muss in dieser Untersuchung betrachtet werden, wie ganze Sätze, ganze Texte, miteinander verglichen werden können. Ein deutsches Sprichwort besagt, dass „das Ganze nicht die Summe seiner Teile" sei. Und das ist im Kontext der natürlichen Sprache eine fundamentale Erkenntnis: die semantische Analyse eines Satzes ist – technisch formuliert - keine *for-each-Loop* durch jedes Wort, dass dann mit dem Wort eines anderen Satzes verglichen wird; vielmehr wird hierbei der Satz als Ganzes betrachtet, analysiert und dieses

Analyseergebnis wird alsdann dem [1] Analyseergebnis des zu vergleichenden Satzes gegenübergestellt.

In dieser Arbeit soll zunächst zusammengefasst werden, auf welcher Grundlage natürlichsprachige Texte (von Computern) bezüglich ihrer Ähnlichkeit bewertet werden können, ein System, dass die state-of-art Modelle mit einer dazugehörende Fragestellung entwickelt und abschließend diese Modelle mit den gängigen Verfahren in Bezug auf die Fragestellung verglichen werden.

Das über allem stehende Ziel ist, die verschiedenen state-of-the-art Verfahren, die im Rahmen dieser Thesis betrachtet und umgesetzt werden, in einem gemeinsamen Setting zu bewerten und miteinander zu vergleichen.

Genauer gesagt soll einmal herausgefunden werden, welches Verfahren die Ähnlichkeit von Texten am genauesten klassifizieren kann. Dabei sollen alle Modelle mithilfe eines umfangreichen Datensatzes trainiert werden, um auf dieser Basis zu prüfen, welcher Algorithmus den Diskurs natürlicher Sprache am besten repräsentieren kann.
Auch soll in die Diskussion miteinfließen, welche der Algorithmen hinsichtlich Performance am schnellsten arbeiten. Ist ein Verfahren nur marginal schneller, dafür unvertretbar viel langsamer, muss auch dies in der Beurteilung berücksichtigt und ein entsprechender Malus in die Bewertung einbezogen werden.

[1] Die Begriffe Vereinigtes Königreich und Großbritannien sind nicht per se synonym, werden aber häufig zwecks Vermeidung von Wortdopplungen so verwendet

1.3 Aufbau der Arbeit

1.3.1 Teil 1: Grundlagen der semantischen textuellen Verarbeitung

Kapitel 2, Grundlagen

Nach der Einleitung wird in Kapitel 2 in die Grundlagen der Thematik eingeführt. Hierbei geht es um die Problembewältigung: Wo liegen die Hindernisse beim Vergleichen von Texten? Des Weiteren wird eruiert, wie natürliche Sprache mit Softwaresystemen verarbeitet wird. Schließlich werden die Rolle der semantischen Ähnlichkeit untersucht und die verschiedenen Terminologien und Paradigmen bezüglich inhaltlicher Ähnlichkeit erläutert. Aus dem Verständnis für die Begriffe wissensbasierter, korpus-basierter und string-basierter Ähnlichkeit erfolgt das nötige Wissen, um semantische Modelle verstehen zu können.

Kapitel 3, Semantische Modelle

In diesem Kapitel wird aufbauend auf dem theoretischen Wissen über die Verarbeitung natürlicher Sprache in populäre semantische Modelle eingeführt, die die Konzepte aus Kapitel 2 praktisch umsetzen. Dieses Wissen bildet die Grundlage, um das System, seine Komponenten und die mit ihm durchgeführten Experimente nachvollziehen zu können.

1.3.2 Teil 2: Konzeption, Umsetzung und Evaluation

Kapitel 4, Anforderungen an das System

Um eine Anwendung zu bauen, die die bis hierhin vorgestellten Modelle geeignet nutzt, werden in diesem Kapitel die fachlichen und nicht-fachlichen Anforderungen an das System erläutert und auf Grundlage dieser Anforderungen die zu nutzenden Softwarewerkzeuge

ausgewählt. Dies bildet den ersten Teil der Entwicklung des für die Experimente notwendigen Systems.

Kapitel 5, Entwurf

In diesem Teil werden zunächst die fachlichen Komponenten und Beziehungen des Systems beschrieben, worauf die Konzeption der beiden Experimente der Arbeit genauer erläutert werden. Im letzten Teil des Entwurfs werden die technischen Aspekte der Konzeption genannt und erklärt.

Kapitel 6, Umsetzung

In Kapitel 6 wird beschrieben, welche Verarbeitungsschritte durchlaufen und welche Datensätze verwendet werden. Die bei der Umsetzung aufgetretenen Fehler und Probleme werden ebenso beschrieben, wie diejenigen Dinge, die reibungslos abliefen.

Kapitel 7, Evaluation des Systems

Schließlich werden die Resultate der Ähnlichkeitsmaße ausgewertet, visualisiert und miteinander verglichen. Hierbei wird zunächst die angewendete Testmethodik erläutert, woraufhin die Experimente ausgewertet und diese Auswertungen erklärt werden. Am Ende wird ein gemeinsames Fazit gezogen.

Kapitel 8, Abschluss & Ausblick

Abschließend wird ein Gesamtfazit gezogen, die Arbeit rückblickend betrachtet und es werden mögliche Erweiterungen zur Arbeit präsentiert. Ziel ist es, die Fragestellung zu beantworten und geeignete Lösungswege für eine solche Problematik sowie für eine mögliche Fortsetzung dieser Arbeit aufzuzeigen.

2 Grundlagen

Um die in dieser Arbeit verwendeten Modelle und Methoden verstehen zu können, wird zunächst die der semantischen Analyse inhärenten Probleme beispielhaft erklärt, woraufhin in die Terminologien und Paradigmen der Semantik eingeführt wird.

2.1 Was ist das Problem?

Bevor existierende Lösungen zu den bekannten Problemen der Verarbeitung natürlicher Sprache erklärt werden, ergibt eine Benennung ebenjener Hindernisse Sinn. Ein klares Verständnis der Hürden ist essenziell, um die Lösungen in ihrer Gänze verstehen zu können.
Als Beispiel diene Abbildung 2: Auf *Google-News* werden zusammengehörige Artikel zusammengefasst, nämlich die verschiedenen Artikel zum *Brexit*.
Der Mensch ist hier in der Lage, die Artikel einander zuzuordnen. Sie erscheinen thematisch verwandt, wobei schon hier implizit der Mensch die Information nutzt, dass die Artikel zur selben Zeit publiziert wurden.
Werden zwei der Überschriften beispielhaft genauer betrachtet, ergeben sich die der Aufgabe inhärenten Probleme:

1. (Brexit: Lieber spät als hart?, 2019)

2. (May scheitert krachend mit ihrem Brexit-Deal, 2019)

Werden diese Zeichenketten miteinander verglichen, fällt auf den ersten Blick nicht viel Gemeinsames auf. Beispiel 1 und 2 teilen lediglich die Gemeinsamkeit eines gemeinsamen Wortes, nämlich den Begriff des sogenannten *Brexits*, dem britischen Austritt aus der Europäischen Union.

Arbeitete man nun intuitiv mit Prozentzahlen, könnte man zu der Aussage verleitet werden: „Die Menge der Wörter aus Beispiel 1 kommt zu 20% in Beispiel 2 vor". Des Weiteren würde eine Betrachtung von Synonymen ähnlich wenig Ertrag bringen; die Formulierung „Lieber spät als hart?" ist gänzlich kontextbefreit und auf den ersten Blick in keiner Weise spezifisch zu einem Thema zugehörig.

Hinzu kommt, dass der Spiegel eine deutlich kreativere Sprache verwendet: das Wort „krachend" fügt dem beschriebenen Ereignis eine bildhafte Komponente hinzu, die faktisch kein Teil der Grundinformation ist.

Was wichtig und am Relevantesten dafür ist, dass der Mensch hier dennoch eine klare thematische Übereinstimmung erkennt, ist nicht die Tatsache, dass es ein gemeinsames Wort gibt, sondern dass das gemeinsame Wort „*Brexit*" extrem spezifisch ist und die Existenz dieses Wortes in einem Satz ein klares Indiz auf dieses Thema ist. So selten dieses Wort in Texten vorkommt – wenn es vorkommt, ist es zumeist Leitmotiv und Kerninhalt des Textes. Je seltener ein Wort in Texten auffindbar ist, desto aufschlussreicher ist es, wenn es vorkommt.

Die Anzahl solcher Probleme und die Art, wie mit diesen umzugehen ist, ist facettenreich und unterscheidet sich von Beispiel zu Beispiel.

Mehr als die Überschrift sagen jedoch die Inhalte der Texte. Im Folgenden werden je eine Passage aus beiden Texten, die bewusst so gewählt wurden, verglichen:

Rechtlich tritt Großbritannien am 29. März, also in knapp zehn Wochen, aus der EU aus. Gibt es bis dahin kein Abkommen zwischen EU und Großbritannien, das eine Mehrheit im britischen Parlament findet, hätte der Exit unkalkulierbare Folgen. (Brexit: Lieber spät als hart?, 2019)

Am 29. März 2019 tritt Großbritannien aus der EU aus. Wenn bis dahin kein Abkommen zustande kommt, erfolgt ein sogenannter harter Brexit - die wirtschaftlichen und finanziellen Folgen wären allen Prognosen zufolge verheerend. (May scheitert krachend mit ihrem *Brexit*-Deal, 2019)

In diesem Fall teilen die beiden Textpassagen deutlich mehr Merkmale:

1. Der erste Teil beider Ausschnitte enthält die Information darüber, wann Großbritannien die Europäische Union verlässt.
2. Ein einfacher Vergleich der verwendeten Wörter reicht, um durch Datum und Subjekte der Sätze eine große inhaltliche Übereinstimmung zu erkennen.
3. Auch im zweiten Satz ist nicht nur die Botschaft dieselbe, sondern auch einige entscheidende Begriff dieselben: *„Folgen“, „bis dahin“, „Abkommen“*

Entsprechend muss ein Automatismus bzw. eine Anwendung für solche Problemstellungen die Wichtigkeit und Relevanz von Wörtern bezüglich ihrer Häufigkeit und Spezifizität in Texten bewerten können und aus diesen Schlüsse ziehen. Selbstverständlich muss es auch in der Lage sein, Synonyme sowie miteinander verwandte Wörter einander zuordnen zu können, weil eine identische Wortwahl eher Ausnahme als Regel in Texten darstellt.
Im Speziellen werden bei Artikeln über bloße Ereignisse oft Inhalte der Deutschen Presseagentur gekauft und verwendet, dann werden in den neu formulierten Texten aktiv Passagen aktiv umformuliert, damit die Texte keine offensichtliche Ähnlichkeit aufweisen. Paraphrasierung, also die Umschreibung von erhaltenen Informationen in den eigenen Worten, ist hier ein entscheidendes Stichwort, um die Ursache dieser Problematik nachvollziehen zu können.

2.2 Verarbeitung natürlicher Sprache

Natural Language Processing (kurz: NLP) ist ein Teilgebiet der Informatik mit Querbezügen zur Linguistik, welches sich damit beschäftigt, wie natürliche Sprache maschinell verarbeitet werden kann (Mitkov, 2005). Zu den natürlichen Sprachen zählen alle Sprachen, die von Menschen geschrieben und gesprochen werden (Carstensen, et al., 2004). Das Spektrum der Anwendungsgebiete von NLP beinhaltet Suchmaschinen, maschinelle Übersetzung, Spracherkennung, Texterzeugung, usw. (Jurafsky, et al., 2008). So wird milliardenfach beim Tippen auf dem Smartphone in Form der „Autokorrekturfunktion“ eine Anwendung aus dem Bereich des Natural Language Processings tagtäglich genutzt.

Bei der maschinellen Verarbeitung natürlicher Sprache kann man sechs Analysebereiche unterscheiden, die wesentlich für das Verständnis von Sprache sind (siehe Abbildung 3: Die sechs Analysebereiche der natürlichen Sprachverarbeitung. Fett hervorgehoben ist der Analysebereich, mit dem sich diese Arbeit am ausführlichsten auseinandersetzt.). Diese

Bereiche stützen sich auf die Erkenntnisse sowie Terminologie der Linguistik (Jurafsky, et al., 2008) (Carstensen, et al., 2004).

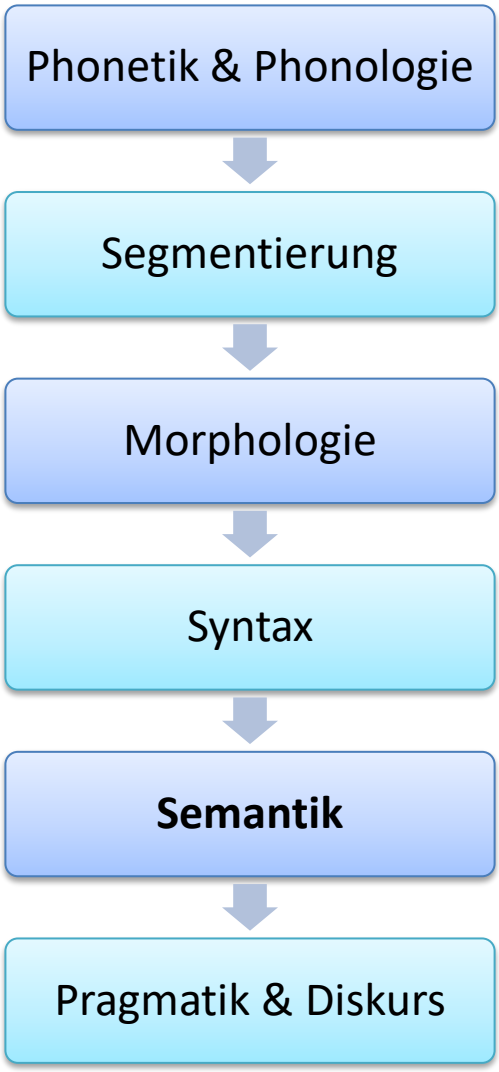


Abbildung 3: Die sechs Analysebereiche der natürlichen Sprachverarbeitung. Fett hervorgehoben ist der Analysebereich, mit dem sich diese Arbeit am ausführlichsten auseinandersetzt.

Der erste Analysebereich umfasst die physikalischen Eigenschaften von Sprachlauten, z.B. deren physiologische Produktion und akustische Wahrnehmung, (Phonetik) sowie die abstrakte, grammatikalische Charakterisierung von Systemen von Klängen oder Zeichen (Phonologie). Bei der natürlichen Sprachverarbeitung betrachtet man hier die Artikulation von Sprachlauten und ihre lautgetreue Transkription mittels der Lautschrift. Dies

entschlüsselt in einer Rede schnell und eindeutig, ob von einem Mitglied einer jüdischen Sekte, einem [ɛˈseːnɐ], oder doch von einer aus Essen stammenden Person, einem [ˈɛsənɐ] die Rede ist, wenn von einem „Essener“ gesprochen wird. Bereich ist heutzutage sehr relevant, wenn es um Sprachassistenzsysteme geht. Wörter, die die gleiche Aussprache, aber eine unterschiedliche Bedeutung haben (sog. „Homophone“), sind hier ein großes Problem. Zum Beispiel kann man die englischen Nomen „knight“ und „night“ nicht unterscheiden, wenn ihr Kontext nicht verfügbar ist, mit der diese Mehrdeutigkeit aufgelöst werden kann. In der deutschen Sprache sind die Wörter Mahl, Mal (mathematisches Symbol), Mal (Muttermal) und mal (Adverb) ein klassisches Beispiel für Homophone.

Die Segmentierung von natürlicher Sprache ist die Zerlegung von komplexen Sprachelementen in ihre nächstkleineren Bestandteile. Beispielsweise kann ein Text als eine Sammlung von Sätzen und jeder Satz als eine Reihe von Wörtern und Satzzeichen betrachtet werden. Das bekannteste NLP-Verfahren hierfür ist die sog. „Tokenisierung“, d.h. die Zerlegung von Umwandeln einer Zeichenfolge in eine Folge von „Token“ (z.B. Wörter, Eigennamen und Satzzeichen). Hierbei stellen das Erkennen und korrekte Zerlegen von Klitika (z.B. „haste“ statt „hast du“) und zusammengehörenden Ausdrücken wie z.B. „New York“ eine Herausforderung dar.

Der dritte Analysebereich ist die Morphologie von Sprache. Diese bezeichnet den Aufbau und die Struktur von Wörtern. Man betrachtet daher Wörtern und Wortteile hinsichtlich Wortstämme, Präfixen, Suffixen, usw. In diesem Bereich analysiert man, wie die zahlreichen morphologische Ausprägungen eines Wortes auf eine einheitliche Stammform abgebildet werden. Dies kann auf zwei Wegen geschehen: durch Stammformreduktion (engl. *stemming*) oder durch Lemmatisierung. Stemming ist die schnellere Variante, erzeugt aber auch Stammformen, die keine richtigen Wörter sind. So wird schnell aus den Wörtern

- operate
- operating
- operates
- operation
- operative
- operatives
- operational

die Stammform “oper”, welche hier mehrere Bedeutungen verschiedener Wörter uneindeutig abbildet (Cambridge University Press, 2008) (Manning, et al., 2008). Dies erhöht die Richtig-positiv Rate bzw. Trefferquote (Recall) und mindert den positiven Vorhersagewert bzw. die Relevanz (Precision).

Obwohl auch das Deutsche und Französische reflektiv sind, also Sprachen, bei denen grammatikalische Fälle und Informationen wie Tempus, Kasus und Numerus durch die Veränderung des Wortstammes (bergen, barg, hat geborgen) ausgedrückt werden, unterscheiden sie sich in diesem Punkt von der englischen Sprache. Bei ihnen funktioniert die Anwendung des Stemmings gut gemäß den Ergebnissen des European CLEF.
Speziell für die deutsche Sprache eigneten sich sogenannte „Compound Splitter“, die deutsche Kofferworte (Apfelbaum -> Apfel, Baum) sehr gut zerlegen.

Dahingegen ist die Lemmatisierung ein Prozess, der zunächst die Wortart (engl. part of speech, POS) eines Wortes bestimmt und dann verschiedene Normierungsregeln für jede Wortart anwendet. Ein bekanntes Problem in diesem Bereich ist die Bildung des korrekten Wortstamms in Sätzen, welche Homographen (Wörter mit gleicher Schreibweise aber mit unterschiedlicher Bedeutung) beinhalten. Beispielsweise ist dies bei „I saw the saw“ der Fall. Im Deutschen gibt es dasselbe Problem mit den Wörtern Zahlen (Plural der „Zahl“) und Zahlen (Nominalisierung des Verbes „zahlen). Dieses Problem betrifft Logographien wie die chinesische Schrift nicht: Die japanischen Wörter 端 (Kante)、橋 (Brücke) und 箸 (Essstäbchen) werden allesamt gleich (Latein: hashi, japanische Silbenschrift: はし) ausgesprochen (jisho, 2019).

Die natürliche Sprachverarbeitung befasst sich außerdem mit der Syntax von Texten. Diese umfasst die geregelte strukturelle Anordnung von Wörtern in Phrasen und Sätze gemäß einer bestimmten Grammatik. Eine Grammatik ist eine Beschreibung der gültigen Strukturen in einer Sprache. Zu den hier verwendeten Methoden gehören das sog. „POS-Tagging“ (die Zuordnung von Tokens zu Wortarten basierend auf ihrer Definition und seinem Kontext) und das „Parsing“ (die Bestimmung der grammatikalischen Struktur in Bezug auf eine bestimmte Grammatik). Problematisch sind hier Holzwegsätze wie z.B. „The government plans to raise taxes were defeated“, welche die Zuordnung von Wortarten erschweren.

Des Weiteren spielt die Semantik von Texten eine wichtige Rolle. Hierbei geht es um die Bedeutung von lexikalischen Einheiten (z.B. Wörtern) und komplexeren Sprachstrukturen (beispielsweise Zeitungsartikel). Diese Arbeit fokussiert sich hauptsächlich auf diesen Analysebereich mitsamt seinen Verfahren. Eine nennenswerte Herausforderung ist die Disambiguierung bei Wörtern, die verschiedene Bedeutungen haben (z.B „Ball“, das Spielgerät beziehungsweise die Tanzveranstaltung). Diese werden in der Linguistik als „Homonyme“ bezeichnet. Außerdem treten bei sog. „Polysemen“ (Spracheinheiten, welches verschiedene Begriffe repräsentiert) ebenfalls sprachliche Mehrdeutigkeiten auf, die ebenfalls aufgelöst werden müssen. Ein Beispiel hierfür ist der Begriff „Bank“, welcher das

Kreditinstitut beziehungsweise die Sitzgelegenheit bezeichnen kann. Homonyme haben eine verschiedene Bedeutung und Herkunft, wohingegen Polyseme einen gemeinsamen Stamm haben und oft voneinander abgeleitet sind.
Der letzte Analysebereich ist die Pragmatik und der Diskurs. Es wird sich damit befasst, wie der Kontext zur Bedeutung von Äußerungen beiträgt. Ziel ist es u.a. herauszufinden, welche Absichten der Sprecher hat, wenn dieser sich in verbaler Form äußert. Eine exemplarische Äußerung ist „Da ist die Tür!“, welche inhaltlich die Absicht verfolgt, den Empfänger darauf hinzuweisen, den Raum zu verlassen. Ein Diskurs ist eine zusammenhängende Gruppierung von Sätzen. Hier geht es darum, die Beziehungen zwischen den Sätzen zu identifizieren. Zum Beispiel: „Der Junge spielt mit seinem Hund. Er hat viel Spaß dabei.“ Referenziert der zweite Satz eine genannte Entität des vorherigen Satzes. An diesen beiden Beispielen erkennt man aber auch, dass je nach Lesart die Sätze anders verstanden werden können: „Da ist die Tür!“ als Hinweis für den Empfänger und „Der Junge spielt mit seinem Hund. Er hat viel Spaß dabei.“ Hieran zeigt sich deutlich die aufzulösende Ambiguität, die sich insbesondere bei sarkastischen Äußerungen als herausfordernd darstellt.

Die vorangegangenen Ausführungen zeigen, dass die Analyse natürlicher Sprache einen vielschichtigen Aufbau hat und besonders durch die Mehrdeutigkeiten, welche in natürlichen Sprachen auftreten, erschwert wird.

2.3 Ähnlichkeit natürlicher Sprache

Für die Entwicklung eines Ähnlichkeitsmaßes muss man sich zunächst damit auseinandersetzen, wann natürlichsprachige Texte als „ähnlich“ betrachtet werden können. Generell werden zwei Objekte auf der Grundlage des Vergleichs ihrer gedanklichen Repräsentationen nach ihrer Ähnlichkeit beurteilt. Diese Repräsentationen sind lediglich Modelle und umfassen daher nur eine begrenzte Anzahl der Objekteigenschaften. Dies kann man auch auf NLP übertragen (Harispe, et al., 2017): Um die Ähnlichkeit zweier Texte zu bestimmen, müssen ihre Repräsentationen miteinander verglichen werden.
Außerdem ist der Begriff „Ähnlichkeit“ in diesem Kontext überladen und kann u.a. als Bedeutungsgleichheit oder als Verwandtschaft von einzelnen Wörtern, Sätzen, Texten, usw. verstanden werden. Die Art und Weise der Repräsentationen hat deswegen auch Einfluss, mit welchen (mathematischen) Verfahren diese Abbilder verglichen werden (Harispe, et al., 2017).

Bei der Ähnlichkeitsuntersuchung von Dokumenten kann man drei Kategorien unterscheiden (Kohila, et al., 2016): String-basierte, Korpus-basierte und wissensbasierte Ähnlichkeit.

2.3.1 String-basierte Ähnlichkeit

Ein String-basiertes Ähnlichkeitsmaß bestimmt die Zeichenähnlichkeit zwischen zwei Zeichenketten. Bei der sog. „Fuzzy-String-Suche“ (eine Technik, um Zeichenketten zu finden, die ein bestimmtes Muster ungefähr erfüllen) können „Sam“ und „Samuel“ als ähnlich betrachtet werden (Lu, et al., 2013).
Bekannte Verfahren sind hier die Levenshtein-Distanz (Levenshtein, 1965) und der Dice Koeffizient (Dice, 1945). Bei String-basierten Methoden wird überwiegend syntaktische Ähnlichkeit (z.B. die Anzahl gleicher Wörter) betrachtet, aber bei der Ähnlichkeit von Texten spielen vor allem Synonyme eine wichtige Rolle. Um diese jedoch erkennen zu dürfen, bedarf es Verfahren die die Bedeutung von Wörtern berücksichtigen.

Es sind daher Maße für die Bestimmung von semantischer Ähnlichkeit notwendig. Nach (Harispe, et al., 2017) umfasst im Allgemeinen solch ein semantisches Maß mathematische Werkzeuge zur Bestimmung des Ausmaßes des semantischen Zusammenhangs zwischen Spracheinheiten, Konzepten, usw. mithilfe einer numerischen Beschreibung, die aus dem Vergleich der bedeutungsspezifischen Informationen hervorgeht.
In diesem Bereich ordnet man Korpus-basierte und wissensbasierte Ähnlichkeit ein.

2.3.2 Korpus-basierte Ähnlichkeit

Korpus-basierte Verfahren definieren die Ähnlichkeit zwischen sprachlichen Einheiten (z.B. einzelne Wörter) auf der Grundlage von unstrukturiertem beziehungsweise semi-strukturiertem Text (z.B. Wörterbücher oder Textkorpora) (Kohila, et al., 2016) (Harispe, et al., 2017).
Sämtliche Verfahren dieser Art lassen sich in vier Bestandteile zerlegen (siehe Tabelle 1):

Tabelle 1: Bestandteile von Korpus-basierten Methoden nach (Harispe, et al., 2017)

Bestandteile
1. Eine Voraussetzung dafür, was die Bedeutung eines Wortes ist.

2. Annahmen, die textbasierte Ansatzpunkte definieren, aus denen die Bedeutung eines Wortes erfasst wird und ein semantisches Maß abgeleitet werden kann.
3. Eine Repräsentation eines Wortes (in seiner Grundform) und einem Großteil seiner Bedeutung in einem semantischen Model.
4. Ein Algorithmus oder eine mathematische Funktion, die zum Vergleich von zwei Wortdarstellungen herangezogen wird.

Zunächst wird festgelegt, was die Bedeutung eines Wortes umfasst. Dies hat unmittelbaren Einfluss auf die Auswahl der Form der Wortrepräsentation und der Herangehensweise diese Repräsentationen miteinander zu vergleichen. In Korpus-basierten Maßen werden Wörter lediglich bezüglich ihrer Benutzung im Text miteinander verglichen (Harispe, et al., 2017).

Bedeutung eines Wortes & der Kontextbegriff

Gemäß der sog. „distributionellen Hypothese“ von (Harris, 1954) haben Wörter, die in ähnlichen Kontexten vorkommen, eine ähnliche Bedeutung und sind daher semantisch miteinander zusammenhängend. Der „Kontext“ ist ein zentraler Begriff in NLP, um die Bedeutung eines Wortes durch die Untersuchung von syntag-[2] und paradigmatischen[3] Beziehungen zu erfassen. Der syntagmatische Kontext geht auf die Untersuchung von gemeinsamem Auftreten von Wörtern ein. Bei diesem Ansatz wird üblicherweise eine bestimmte Fenstergröße von Wörtern verwendet, um einen Kontext zu definieren (Harispe, et al., 2017). Dahingegen betrachtet der paradigmatische Kontext sog. „indirektes gemeinsames Auftreten“, d.h. Situationen, in denen zwei Wörter mit den gleichen Worten, aber nicht zusammen vorkommen (z.B. Synonyme). Derartige Kontexte werden im Allgemeinen in Bezug auf die grammatikalische Abhängigkeit zwischen Wörtern definiert (Harispe, et al., 2017).

Distributionelle Maße

Im Zusammenhang mit Korpus-basierten Methoden spielen distributionelle Maße, die sich auf die distributionelle Hypothese (Harris, 1954) stützen, eine zentrale Rolle. Derartige Maße bestehen erstens aus einem Modell, welches Wörter in einem multidimensionalen Raum abbildet und zweitens aus einem Verfahren, um die Wortrepräsentationen miteinander zu

[2] bezieht sich auf den Zusammenhang einzelner Wortformen in ihrer Reihenfolge innerhalb eines komplexen Ausdrucks

[3] bezieht sich auf die Menge der Ausdrücke, die je nach Kontext gegeneinander substituierbar sind

vergleichen. Deswegen unterscheiden sich distributionelle Maße dahingehend, welche beiden Herangehensweisen hierfür verwendet werden (Harispe, et al., 2017).

Um ein distributionelles Modell zu konstruieren, kann man folgendermaßen vorgehen (Harispe, et al., 2017):

1. Die Vorverarbeitung des Textkorpus ist optional und umfasst Techniken, um beispielsweise den Text zu segmentieren, Wortarten zuzuordnen, Stoppwörter (wie z.B. Konjunktionen und Artikel) herauszufiltern, das Vokabular zu erstellen, usw.
2. Die Auswahl eines Kontexts, um ein Wort zu charakterisieren (z.B. Sätze, Dokumente, Wortfenster, usw.), indem syntag- und paradigmatische Wortbeziehungen untersucht werden. Wörter können z.B. in einem Vektorraum als Wort-Kontext-Matrix abgebildet werden.
3. Eine optionale Gewichtung der Wörter, um Häufigkeit und Relevanz von Begriffen zu beurteilen. Ein bekannter Ansatz ist das sog. „Term frequency–inverse document frequency-Maß“ (Tf-idf), welches die Suchwortdichte in Abhängigkeit von der Relevanz von Begriffen im gesamten Dokumentenkorpus betrachtet (Luhn, 1957) (Spärck Jones, 1972).[4]
4. Eine optionale Verringerung der Dimensionen der Matrix, um die Dichte zu erhöhen und z.B. indirektes gemeinsames Auftreten von Wörtern hervorzuheben. Ein Verfahren hierfür ist die „Singulärwertzerlegung“ (SVD) (Strang, 2016), welche eine Matrizenzerlegung zur Reduktion einer Matrix auf ihre Bestandteile durchführt.

Nachdem das Modell erstellt worden ist, kann man folgende Arten von Verfahren können zum Wortvergleich heranziehen (Harispe, et al., 2017):

- Geometrisch-räumliche Verfahren vergleichen zwei Wörter bezüglich ihrer Position im multidimensionalen Raum. Da Wörter als Kontextvektoren der Matrix des Modells repräsentiert werden, nutzt man Maße zum Vergleich von Vektoren wie z.B. die „Kosinus-Ähnlichkeit“ (Mikolov, et al., 2013), welche ein Maß dafür ist, wie zwei Vektoren in Bezug auf ihren Vektorwinkel zueinander stehen (unabhängig von ihrer Größe). Zwei Vektoren sind demnach identisch, wenn deren Kosinus-Ähnlichkeit 1 ist und -1, wenn sie entgegengesetzt gerichtet sind. Wenn negative Gewichte ausgeschlossen werden, liegt dieser Wert zwischen 0 und 1.

[4] Demnach ist ein Wort „wichtig“, wenn es in einem bestimmten Dokument häufig, aber in allen anderen Dokumenten selten auftritt.

- Die mengenbasierten Ansätze vergleichen Wörter, indem die Anzahl der Kontexte analysiert werden, in denen die Wörter vorkommen. Für den Vergleich kann der Dice-Koeffizient herangezogen werden (siehe 2.3.1).
- Probabilistische Methoden betrachten die Wahrscheinlichkeit, mit der zwei Wörter im Korpus vorkommen, sowie der Wahrscheinlichkeit, dass die beiden Wörter im gleichen Kontext gemeinsam vorkommen. Ein probabilistischer Ansatz ist „Pointwise mutual information" (PMI), der die Anzahl des gemeinsamen Vorkommens und des individuellen Auftretens von Wörtern untersucht (Fano, 1961).

Neben den distributionellen Maßen gibt es noch andere Arten Korpus-basierter Methoden. (Harispe, et al., 2017) stellt diese kurz und übersichtlich dar.

Beurteilung Korpus-basierter Verfahren

Tabelle 2: Vorteile und Beschränkungen Korpus-basierter Verfahren nach (Harispe, et al., 2017)

Vorteile	Beschränkungen
• können verwendet werden, um den semantischen Zusammenhang von in Korpora enthaltenen Wörtern ohne Vorkenntnisse über ihre Bedeutung oder Verwendung zu bestimmen • ermöglichen eine fein abgestufte Analyse, da die Abstimmung von Maßen unter Berücksichtigung syntagmatischer und paradigmatischer Zusammenhänge erfolgen kann • können verwendet werden, um nicht nur Wörter, sondern auch komplexere Spracheinheiten wie z.B. Texte miteinander zu vergleichen	• die zu vergleichenden Worte müssen mindestens ein paar Male im Korpus auftreten • die Maße sind sehr abhängig vom verwendeten Korpus und die Ergebnisse sind daher nicht unbedingt repräsentativ • es ist hiermit schwierig den semantischen Zusammenhang zwischen Konzepten zu bestimmen • streng genommen ist das Betrachten des gemeinsamen Auftretens zweier Wörter kein Indiz für deren semantische Ähnlichkeit, sondern eher ein Zeichen für deren semantischen Zusammenhang (engl. *semantic relatedness*)[5]

[5] „Eiscreme" und „Löffel" sind semantisch verwandt, wenn die beiden Wörter oft gemeinsam auftreten; dennoch sind beide Begriffe von der Bedeutung her unterschiedlich

Tabelle 2 stellt klar, dass Korpus-basierte Verfahren zwar für den Vergleich aller Bausteine einer Sprache geeignet sind, aber nicht zwingend Aussagen über die semantische Ähnlichkeit von Elementen liefern.

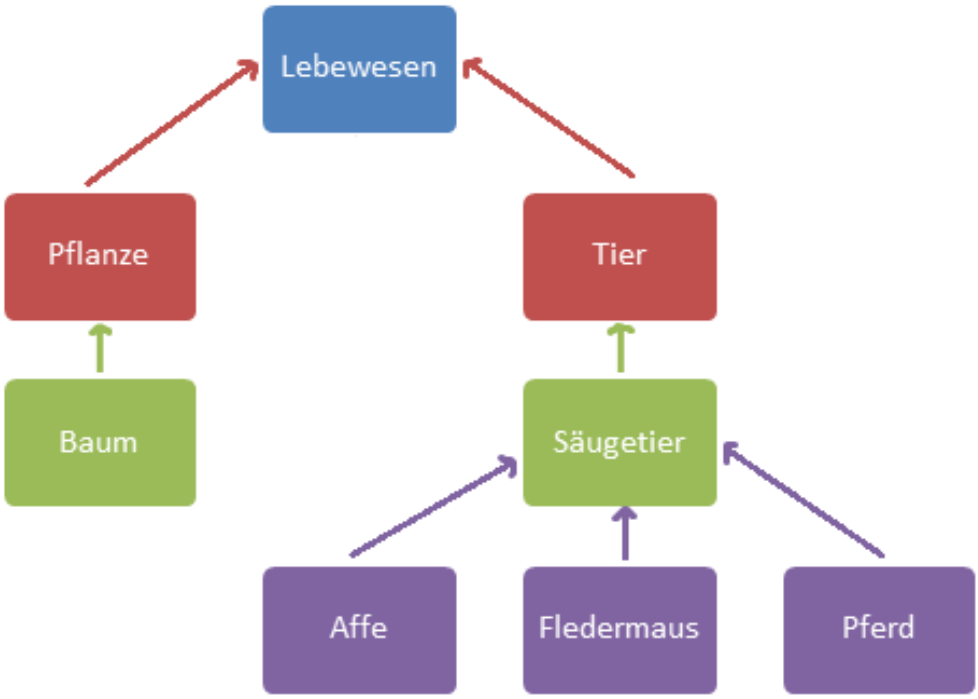
2.3.3 Wissensbasierte Ähnlichkeit

Wissensbasierte Methoden dienen zum Vergleich von Elementen, die strukturiert in Form von in Ontologien vorliegen (Kohila, et al., 2016) (Harispe, et al., 2017). Eine Ontologie ist eine explizite Spezifikation einer abstrakten und vereinfachten Sicht auf Dinge eines gemeinsamen Gegenstandsbereich, die repräsentiert werden sollen, indem domänenspezifische Begrifflichkeiten, Konzepte, Instanzen, Attribute und andere Entitäten sowie ihre Beziehungen zueinander geordnet dargestellt werden. Mithilfe dieser Spezifikation kann Wissen formal als Netzwerk repräsentiert werden (Gruber, 1993).

Graph-basierte Maße

Beispiele für eine einfache Ontologie, dargestellt als Graph, sind Taxonomien. Taxonomien dienen dazu, Elemente mit ähnlichen Merkmalen in geordnete hierarchische Klassen zu untergliedern. Abbildung 4 zeigt eine exemplarische Taxonomie.

Abbildung 4: Eine beispielhafte Taxonomie von Lebewesen dargestellt als gerichteter Graph. Die Pfeile können als „ist ein-Beziehungen“ interpretiert werden.



Auf der Basis des Graph-basierten Ansatzes zur Darstellung von Taxonomien mit partieller Ordnung, kann man drei Ansätze unterscheiden, um die semantische Ähnlichkeit zweier Konzepte[6] miteinander zu vergleichen (Harispe, et al., 2017)[7]:

- Analyse von Graphstrukturen:
  Die Ähnlichkeit zweier Konzepte kann einerseits durch den Grad der Vernetzung abgeleitet, indem der Abstand beider Konzepte im Graph analysiert wird (z.B. die Untersuchung der Länge der Pfade, die beide Konzepte miteinander verbinden). Andererseits können derartige Maße als Funktionen betrachtet werden, die die Ähnlichkeit zweier Konzepte ausgehend vom Aufwand, ein Konzept in ein anderes zu transformieren, abschätzen. Diese strukturellen Maße basieren auf graphentheoretischen Traversierungsalgorithmen (z.B. die Bestimmung des kürzesten Pfades zwischen zwei unterschiedlichen Knoten).

[6] Ein Konzept beziehungsweise Klasse ist die Gesamtheit der Dinge, welche gemeinsame Eigenschaften teilen (z.B. das Konzept „Säugetier“).
[7] Diese Arten von Maßen können miteinander kombiniert werden (Harispe, et al., 2017).

- Analyse von Konzeptmerkmalen:
  Auf der Grundlage extrahierter Merkmale von Konzepten aus dem Graph wird die Ähnlichkeit hinsichtlich der gemeinsamen und unterschiedlichen Merkmale der verglichenen Konzepte abgeschätzt. Konzepte werden hier als Sammlungen von Merkmalen (engl. *Features*) repräsentiert

- Informationstheoretische Maße:
  Derartige Maße bewerten die Ähnlichkeit anhand von Gemeinsamkeiten und Unterschieden des Informationsgehaltes (gemäß (Shannon, 1948) (Resnik, 1995)) zwischen verglichenen Konzepten und betrachten keine Hierarchien von Konzepten. Im Gegensatz zur Analyse von Konzeptmerkmalen wird kein boolescher Abgleich durchgeführt, sondern auf den Grad des Informationsgehalts geachtet.

Eine umfassende Liste jedweder Graph-basierter Maße sowie Methoden, die neben den Graph-basierten existieren (z.B. Maße, die auf logikbasierte Semantik beruhen und semantische Maße für die gleichzeitige Betrachtung mehrerer Ontologien), ist in (Harispe, et al., 2017) dargestellt.

Des Weiteren gibt es Hybridverfahren, die sowohl Korpus- und als auch wissensbasierte Ansätze miteinander vermischen. Bekannte Beispiele sind sog. „Wikipedia[8]-basierte Maße“, die Graphstrukturen der Hyperlink-Beziehungen zwischen Artikeln, den Textinhalt der Artikel und die jedem Artikel zugeordneten zugrunde liegenden strukturierten Kategorien ausnutzen (Taieb, et al., 2013) (Harispe, et al., 2017).

Beurteilung wissensbasierter Verfahren

Tabelle 3: Vorteile und Beschränkungen wissensbasierter Verfahren nach (Harispe, et al., 2017)

Vorteile	Beschränkungen
• können verwendet werden, um alle Arten von Dingen, die in einer Ontologie definiert werden, miteinander zu vergleichen (inklusive Entitäten, die nicht durch eine	• setzen eine Ontologie voraus, die die zu vergleichenden Elemente beschreibt (wenn keine domänenspezifische Ontologie vorliegt, sind die Maße nicht anwendbar)

[8] https://www.wikipedia.org/

Textanalyse miteinander verglichen werden können) • ermöglichen die detaillierte Kontrolle der semantischen Beziehungen, die beim Vergleich der Elemente berücksichtigt werden • geringerer Aufwand zur Berechnung als Korpus-basierte Maße, da keine komplexe und zeitaufwändige Vorverarbeitung des Textkorpus erforderlich ist	• für den Vergleich von Elementen, die in großen Ontologien definiert sind, ist der Berechnungsaufwand bei logikbasierten Ansätzen hoch • Graph-basierte Verfahren erfordern in der Regel, dass das Wissen im Graphen auf bestimmte Weise modelliert wird (abweichende Beziehungsdarstellungen können dann nicht berücksichtigt werden)

Aus Tabelle 3 lässt sich ableiten, dass wissensbasierte Methoden umfassendere Vergleiche ermöglichen und eine geringere Berechnungskomplexität besitzen, jedoch durch die Abhängigkeit einer vorliegenden Ontologie eingeschränkt sind.

3 Semantische Modelle

In diesem Kapitel werden einerseits Modelle betrachtet, die üblicherweise zur semantischen Analyse herangezogen werden. Andererseits werden einige state-of-the-art Modelle präsentiert, die in den vergangenen Jahren eingeführt wurden. Die meisten Verfahren, die im Folgenden vorgestellt werden, sind unüberwachte Algorithmen Korpus-basierter Art. Es ist anzumerken, dass hier der Fokus mehr auf Modelle zur Repräsentation von Sätzen und Dokumenten betrachtet werden, die kein Deep Learning verwenden und weniger auf Word-Embedding-Modelle. Die in den folgenden Ausführungen aufgeführte Beschreibung von Algorithmen soll keineswegs erschöpfend sein, sondern dient als theoretisches Fundament für die spätere Entwicklung des Systems bezüglich der Zielsetzung dieser Arbeit.

3.1 Populäre Modelle

Dieser Unterabschnitt dient dazu, die Konzepte und Eigenschaften der gängigsten Modelle kurz vorzustellen. Einige von ihnen werden in den anschließenden Kapiteln als Benchmarks dienen. Detailliertere Informationen z.B. hinsichtlich der mathematischen Formeln sind den jeweiligen referenzierten Originalquellen zu entnehmen.

3.1.1 Latent Dirichlet Allocation (LDA)

Dieses statistische Modell gehört zu den probabilistischen Methoden (siehe Abschnitt 2.3.2), welches das sog. „Topic Modeling“ umsetzt. Es repräsentiert jedes Dokument als ein Vektor mit einer Wahrscheinlichkeitsverteilung über eine bestimmte Anzahl von vorher festgelegten abstrakten Themen (engl. *topics*). Jedes Thema ist hier eine Sammlung von

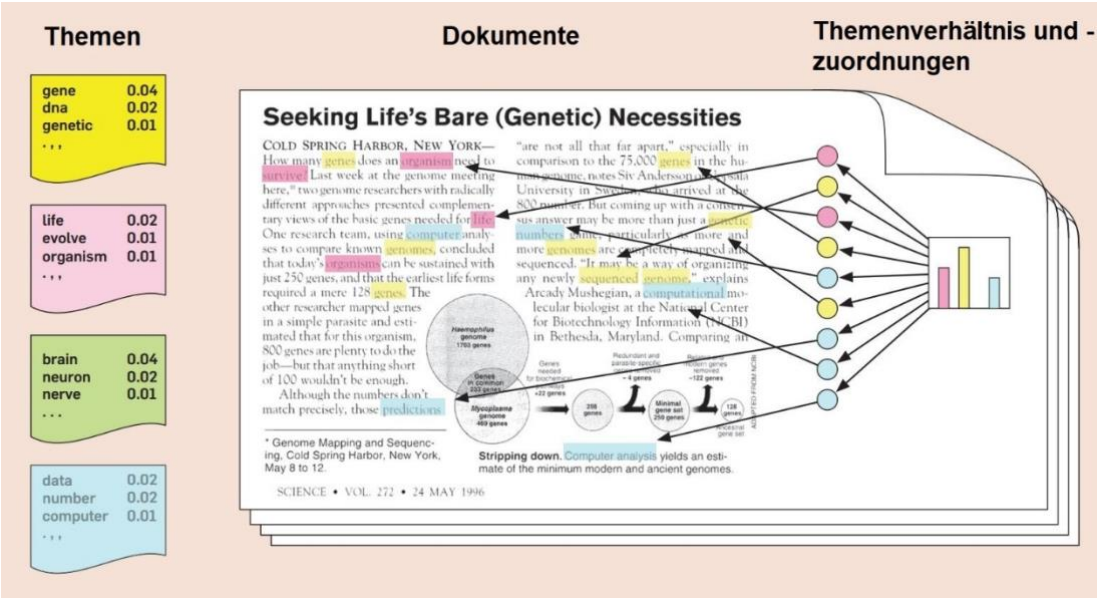
Schlüsselwörtern, welche im Dokumentenkorpus vorkommen und entdeckt wurden (Blei, et al., 2003) (Blei, et al., 2009).

Tabelle 4: Wichtigste Schlüsselwörter zweier Themen in Anlehnung an (Blei, et al., 2009)

Thema „Astronomie“	Thema „Immunologie“
Umlaufbahn	Infektion
Staub	immun
Jupiter	Aids
Strecke	infiziert
System	virusbedingt
solar	Zellen
Gas	Impfstoff
atmosphärisch	Antikörper

Tabelle 4 zeigt anhand eines Beispiels, welche Schlüsselwörter ein bestimmtes Thema repräsentieren. Ein Dokument ist dann ein Wahrscheinlichkeitsverteilung in Form eines Vektors über alle Themen (z.B. 5% Astronomie und 95% Immunologie).

Abbildung 5: Eine einfache Verbildlichung der Funktionsweise von LDA aus (Banerjee, 2016). Ganz links befinden sich die Themen in Form einer Wahrscheinlichkeitsverteilung über alle Wörter des Dokumentenkorpus. Für jedes Dokument wird eine Verteilung über die Themen (siehe Diagramm ganz rechts) bestimmt und jedes Wort des Dokuments wird einem Thema zugeordnet (siehe die farbigen Kreise).



Um herauszufinden, welche Dokumente ähnlich sind, werden die Vektorrepräsentationen miteinander verglichen. Für dieses themenbasierte Ähnlichkeitsmaß kann der Hellingerabstand verwendet werden (Hellinger, 1909) (Blei, et al., 2009).

3.1.2 Word2Vec

Word2Vec ist ein distributionelles Modell (siehe Abschnitt 2.3.2), welches für das Erstellen von sog. „Word Embeddings“ (deutsch *Worteinbettungen*) benutzt wird. Ein Word Embedding ist ein Ansatz, um eine Vektorrepräsentation von Wörtern bereitzustellen, die ihre Bedeutungen erfassen soll.

Word2vec nimmt einen Textkorpus als Eingabe und erzeugt einen multi-dimensionalen Vektorraum, in welchem jedes einzelne Wort als ein entsprechender Vektor repräsentiert und im Vektorraum zugeordnet wird. Vektoren von Wörtern, die in gemeinsamen Kontexten im Korpus auftauchen werden im Vektorraum in unmittelbarer Nähe zueinander positioniert (Mikolov, et al., 2013). Es ist ein zweischichtiges neuronales Netz, welches zwei Techniken verwendet, um die Wortrepräsentationen zu erhalten. Die erste Technik prognostiziert das aktuelle Wort auf der Basis der umgebenden Wörterinnerhalb eines bestimmten Kontextfensters („continuous bag-of-words“ (CBOW)). Das zweite Modell geht genau umgekehrt vor und prognostiziert die umgebenden Wörter eines Kontextfensters auf der Grundlage des aktuellen Wortes („Skip-gram“) (Mikolov, et al., 2013).

Tabelle 5: Parameter, welche die Ergebnisse von Word2Vec stark beeinflussen können nach (Mikolov, et al., 2013) (Elsafty, 2017) (code.google.com, 2013)

Parameter	Kurzbeschreibung
Dimension des Vektorraums	Eine größere Dimensionsanzahl verbessert das Word Embedding, jedoch sind die Verbesserungen ab einem bestimmten Punkt verschwindend gering.
Größe des Kontextfensters	Die Größe k des Kontextfensters betrachtet die k vorausgegangenen und k nachfolgenden Wörter des aktuellen Wortes.
Trainingsalgorithmus	„Hierarchical Softmax“ ist besser für unregelmäßige Wörter geeignet,

	wohingegen „negatives Sampling“ die Trainingszeit verkürzt.
Sub-sampling	Wörter, die sehr oft vorkommen (z.B. Stoppwörter) und einen bestimmten Schwellwert überschreiten, werden entfernt. Dies kann die Genauigkeit verbessern.

Tabelle 6 listet wichtige Parameter auf, die einen großen Einfluss auf das Training von Word2Vec haben können. Erwähnenswert ist zudem, dass Word2Vec im Gegensatz zu LDA die Reihenfolge von Wörtern berücksichtigt.

Auf diesem Modell aufbauend sind einige Varianten und Alternativen wie „Word2VecF“ (Levy, et al., 2014) und „GloVe“ (Pennington, et al., 2014) entstanden.
Word2VecF nimmt im Gegensatz zu Word2Vec sondern Tupel von Wörtern und Kontexten als Eingabe. Während Word2VecF ausgehend von einem bestimmten Wort andere Wörter betrachtet, die im selben Dokument vorkommen, legt Word2Vec den Fokus auf benachbarte Kontextwörter (Elsafty, 2017).
GloVe ist ein Modell, das sich nicht nur lokale Kontextfenster in Betracht zieht, sondern mithilfe einer Matrix erfasst, wie häufig ein Wort in einem Kontext erscheint. Word2Vec ist prädiktiv, während GloVe ein „zählbasiertes Modell“ ist (Alvarez, 2017).

3.1.3 Doc2Vec

Word2Vec ist in der Hinsicht beschränkt, dass lediglich Wörter und nicht ganze Dokumente repräsentiert werden. Eine Möglichkeit, um dies zu erreichen, ist die Darstellung von Dokumentenvektoren durch gemittelte Wortvektoren (Huang, et al., 2012). Als Alternative dazu haben (Le, et al., 2014) ein Modell namens „Doc2Vec“ eingeführt, das zur Einbettung von Textabschnitten (mithilfe von Abschnittvektoren, engl. *paragraph vectors*) dient.

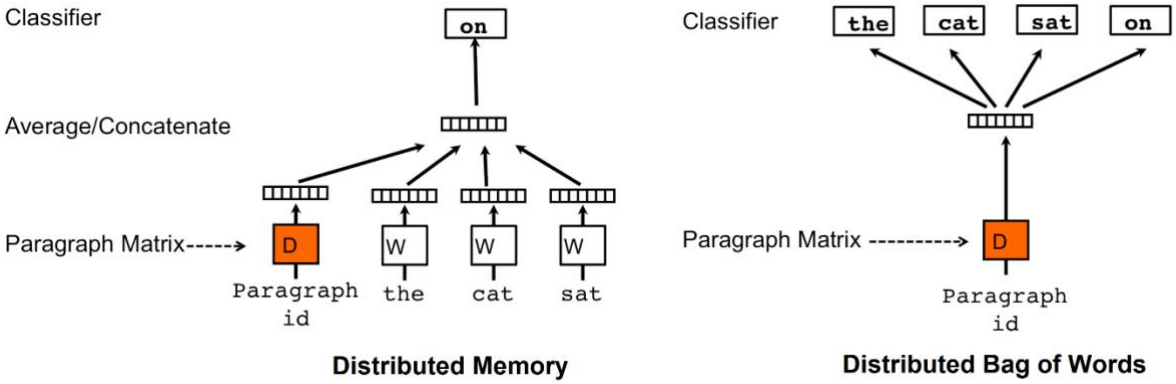
Analog zu Word2Vec werden auch hier zweierlei Architekturen unterschieden (siehe Abbildung 6): Das „distributed Memory“- (PV-DM) und das „distributed Bag of Words“-Modell (PV-DBOW).

- PV-DM funktioniert ähnlich wie Word2Vec-CBOW, indem ein Kontextfenster verwendet wird, um das nachfolgende Wort vorherzusagen. Zusätzlich wird ein entsprechender Abschnittvektor herangezogen, der sich als eine Art „Speicher“ daran erinnert, welches

Thema der Abschnitt hat beziehungsweise was im aktuellen Kontext fehlt (Le, et al., 2014).

- PV-DBOW versucht analog zu Word2Vec-Skip-gram auf der Grundlage eines entsprechenden Abschnittsvektors zufällig ausgewählte Wörter des Dokumentes vorherzusagen. Dieses Modell benötigt für die Klassifikation daher kein Kontextfenster und keine Wortvektoren als Eingabe (Le, et al., 2014).

Abbildung 6: Bei PV-DM wird das aktuelle Wort mithilfe des Abschnittvektors in der Dokumentenmatrix D und der Verkettung beziehungsweise dem Mittel der Vektoren der Kontextwörter in der Wortmatrix W prognostiziert. Dabei wird die Kosinus-Ähnlichkeit zum nächsten Wort optimiert. Bei PV-DBOW werden Wörtermithilfe des Abschnittvektors vorhergesagt, die zufällig aus dem Absatz entnommen wurden. Hierbei kommt es auch zur Optimierung der Kosinus-Ähnlichkeit wischen der Dokumenteneinbettung und den darin enthaltenen Worteinbettungen. Die ursprünglichen Abbildungen sind aus (Le, et al., 2014) entnommen.



Nach (Le, et al., 2014) ist PV-DM ist durchgängig besser als PV-DBOW, jedoch schneidet nach (Lau, et al., 2016) PV-DBOW besser in einem Szenario hinsichtlich semantischer Textähnlichkeit ab. Außerdem ist die Konkatenation der Vektoren bei PV-DM oft besser als der Mittelwert. Der einzige nennenswerte Unterschied von Doc2Vec zu Word2Vec ist das Heranziehen eines Abschnittsvektors. Ansonsten sind beide semantischen Modelle sehr ähnlich hinsichtlich der Worteinbettungen, des Trainings und der Verwendung der Trainingsparameter (Alvarez, 2017). Auf die Nachteile von Doc2Vec wird im Unterabschnitt 3.2.1 eingegangen.

3.2 State-of-the-art Modelle

Im Folgenden wird eine kleine Auswahl von modernen Modellen vorgestellt. Eine ausführliche Übersicht über weitere state-of-the-art Verfahren liefern die beiden vergangenen „SemEval-Workshops“ (Cer, et al., 2017) (Mohammad, et al., 2018). Spezielle Modelle, auf die hier nicht näher eingegangen wird, sind aufwändig zu trainierende Deep Learning-Modelle wie z.B. „Skip-thoughts“ (Kiros, et al., 2015).

3.2.1 Doc2VecC

Der Nachteile von Doc2Vec sind, dass die Absatzmatrix umso größer wird, je größer die Anzahl der Dokumente im Trainingskorpus wird, und dass die Erzeugung neuer Dokumentenrepräsentationen teuer ist, da das Training des Modells dann neugestartet werden muss (Chen, 2017). Zur Abhilfe hat (Chen, 2017) auf der Basis von Doc2Vec eine verbesserte Modellarchitektur namens „Document Vector through Corruption“ (Doc2VecC) vorgeschlagen.

Genauso wie Doc2Vec lernt Doc2VecC Dokumentenrepräsentationen in Form von Vektoren. Aber streng genommen ist es ein Word Embedding-Modell wie Word2Vec, welches Wortrepräsentationen während des Trainings so lernt, dass Dokumente durch Mittelung von zufällig entnommenen Wortrepräsentationen (als *Corruption* bezeichnet) möglichst aussagekräftig und „rauschfrei“ repräsentiert werden können (Alvarez, 2017) (Chen, 2017). Dadurch entsteht eine Regularisierung, die seltene beziehungsweise aussagekräftige Wörter bevorzugt und häufige beziehungsweise nicht-distinktive Wörter vernachlässigt. Dies steht im Gegensatz zu Doc2Vec, welches direkt einen eindeutigen Vektor für jedes Dokument lernt (Chen, 2017).

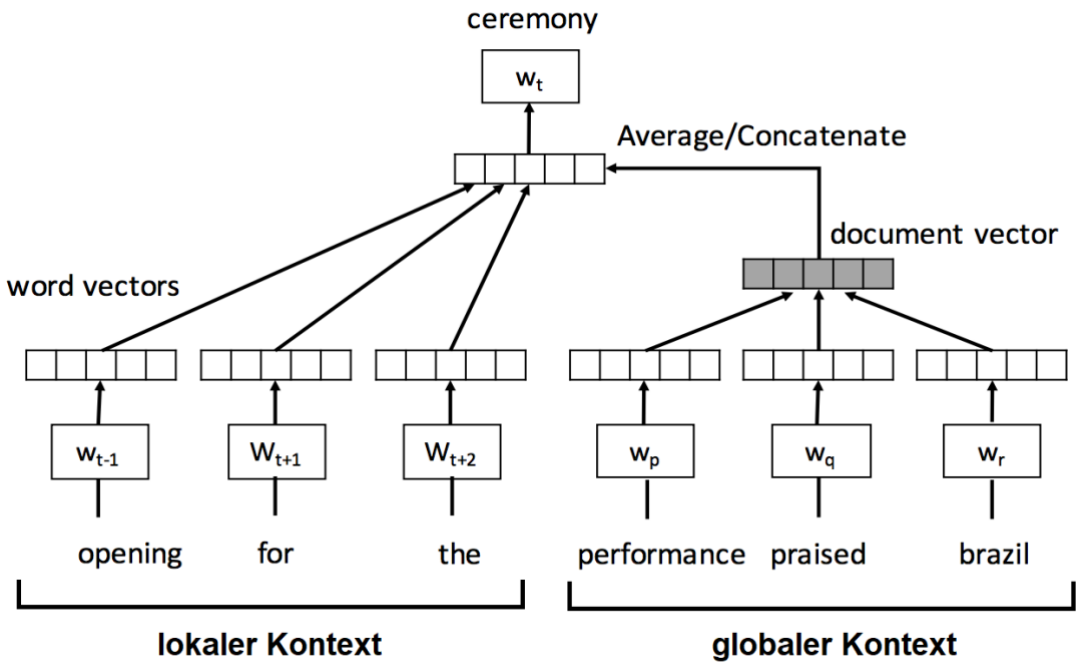


Abbildung 7: Die Architektur von Doc2VecC. Die Repräsentationen benachbarter Wörter liefern einen lokalen Kontext bestehend aus den Wörtern, die das zu prognostizierende Wort (w_t *ceremony*) umgeben, während die Repräsentation des gesamten Dokuments (grau dargestellt) als globaler Kontext bestehend aus zufällig entnommen Wörtern des Dokuments dient. Die Abbildung ist leicht angepasst aus (Chen, 2017) entnommen.

Die Architektur von Doc2VecC ähnelt sehr der von Word2Vec-CBOW, d.h., dass das aktuelle Wort mittels der Kontextwörter innerhalb eines Fensters prognostiziert wird. Zusätzlich entnimmt Doc2VecC, wie bereits beschrieben, zufällig Wörter aus dem Dokument und erzeugt auf dieser Grundlage einen Dokumentenvektor durch die Bildung des Mittelwerts der Wortvektoren (Chen, 2017). Abbildung 7 visualisiert den beschriebenen Sachverhalt.

Doc2VecC hat nicht nur einen geringeren Berechnungsaufwand als Doc2Vec, sondern übertrifft das Modell bei der Generierung von Repräsentationen vorher unbekannter Dokumente während der Testzeit (Chen, 2017).

3.2.2 Sent2Vec

Sent2Vec ist ein von (Pagliardini, et al., 2017) eingeführter Algorithmus zum unüberwachten Lernen von Dokumentenrepräsentationen. Dieser ähnelt sehr der Vorgehensweise von Doc2VecC, da beide Modelle als Erweiterung von Word2Vec-CBOW betrachtet werden, die größere Texteinheiten durch Mittelung der entsprechenden und speziell dafür trainierten Wortrepräsentationen repräsentiert werden.

Die Ausnahme hierbei ist, dass Sent2Vec nicht nur Repräsentationen von Dokumenten lernen kann, sondern auch von Abschnitten und Sätzen, wodurch es im Vergleich zu Doc2VecC für kleinere Texte besser optimiert ist (Pagliardini, et al., 2017) (Alvarez, 2017).

Sent2Vec verwendet als Kontextfenster umfassende semantische Einheiten wie Sätze oder Dokumente [9] und innerhalb des Kontextes n-Gramme, was einerseits aussagekräftige Repräsentationen ermöglicht, aber andererseits das Vokabular aufbläht (Pagliardini, et al., 2017) (Alvarez, 2017).

Eine umfassendere Erläuterung dieses semantischen Modells kann im oben referenzierten Originalpaper nachgelesen werden.

3.2.3 Wpath

Die oben vorgestellten Modelle lassen sich allesamt Korpus-basierten Verfahren zuordnen. Aber auch im Bereich wissensbasierter Methoden hat es in den vergangenen Jahren Neuerungen gegeben. Ein solches state-of-the-art Modell ist „wpath“ von (Zhu, et al., 2017).

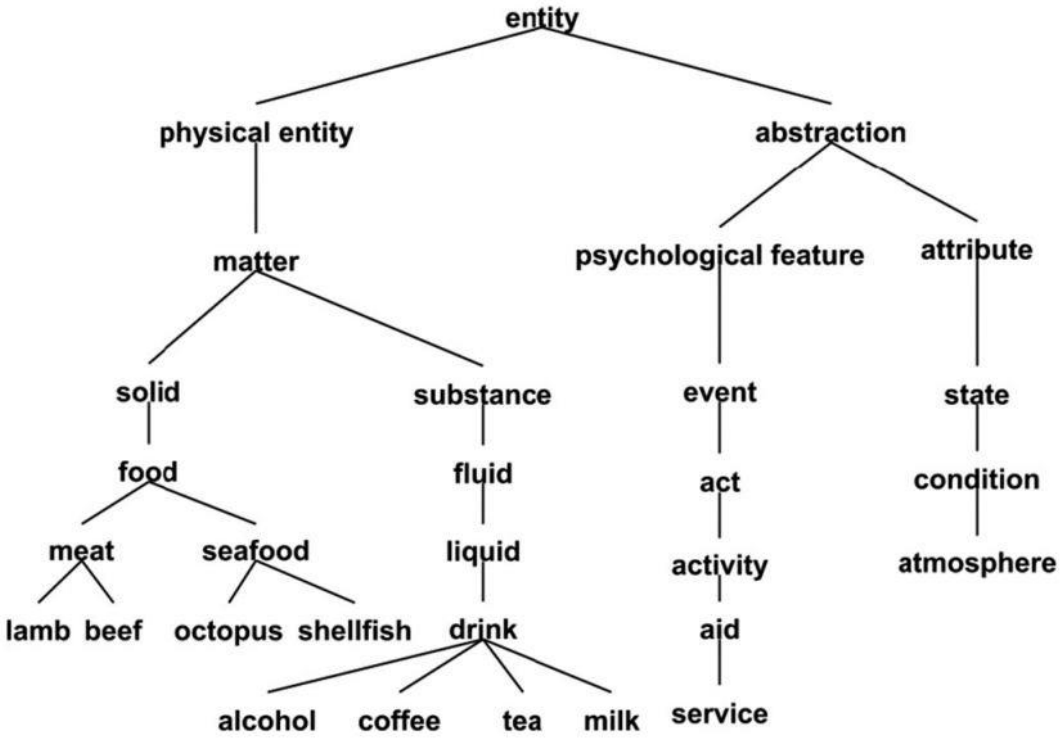
Es ist ein Verfahren zur Messung der semantischen Ähnlichkeit zwischen Konzepten in Wissensgraphen wie „WordNet“ [10] und „DBpedia“ [11], welches sowohl die Analyse von Graphstrukturen (siehe Unterabschnitt 2.3.3) als auch die Untersuchung informationstheoretischer Maße in sich vereinigt (Zhu, et al., 2017).

---

[9] Im Gegensatz dazu betrachtet Word2Vec-CBOW lediglich benachbarte Wörter
[10] https://wordnet.princeton.edu/ (letzter Zugriff: 08.10.2018)
[11] https://wiki.dbpedia.org/ (letzter Zugriff: 08.10.2018)

Analysiert man nur Graphstrukturen als Grundlage für die Ähnlichkeit zweier Konzepte, so kann dies bei der Betrachtung zweier Konzepte mit gleicher Pfadlänge oder Tiefe dazu führen, dass unterschiedliche Konzeptpaare, die gleiche Ähnlichkeit besitzen. Anhand von Abbildung 8 soll dies illustriert werden. Die Konzeptpaare *(meat, seafood)* und *(lamb, beef)* haben die gleiche kürzeste Pfadlänge und deshalb die gleiche Ähnlichkeit. Die oberen Hierarchieebenen enthalten jedoch allgemeinere Konzepte, wohingegen die unteren Ebenen feingranularer sind. Die Tiefe wird in diesem Verfahren nicht berücksichtigt (Zhu, et al., 2017). Die zwei Konzeptpaare *(lamb, beef)* und *(octupus, shellfish)* besitzen die gleiche Tiefe und deshalb die gleiche Ähnlichkeit (Zhu, et al., 2017).

Abbildung 8: Eine exemplarische WordNet-Taxonomie von Konzepten aus (Zhu, et al., 2017).



Um diese beiden Probleme zu lösen und gleichzeitig die Vorteile von graphentheoretischen Metriken ausnutzen ist „weighted path length“ (wpath) eingeführt worden. Dieses Verfahren zieht nicht nur die oben beschriebenen Metriken zur Bestimmung der Ähnlichkeit eines Konzeptpaares heran, sondern betrachtet zusätzlich den Informationsgehalt (siehe Unterabschnitt 2.3.3) der beiden betrachteten Konzepte, welcher zur Gewichtung der kürzesten Pfadlänge verwendet wird. Damit ein Konzeptpaar einen hohen Ähnlichkeitswert

erhält, muss die Distanz zwischen beiden Konzepten minimal sein und beide Konzepte müssen eine Vielzahl an gemeinsamen Informationen aufweisen (Zhu, et al., 2017).

Obwohl wpath zur Bestimmung von Ähnlichkeiten einzelner Wörter beziehungsweise Konzepte entwickelt wurde, können bei der Berechnung der Ähnlichkeit von Dokumenten Entitäten extrahiert und dadurch die Dokumentenähnlichkeit durch die Zusammensetzung von Entitätsähnlichkeitswerten berechnet werden (Sematch, 2017).

4 Anforderungen an das System

Bevor ein System entwickelt werden kann, dass die in Kapitel 3 vorgestellten Modelle evaluiert, muss erst einmal definiert werden, welche konkrete Fragestellungen verfolgt werden und mithilfe des Systems beantwortet werden sollen. Auf dieser Grundlage können anschließend Softwarewerkzeuge ausgewählt sowie funktionale und nicht-funktionale Anforderungen abgeleitet werden.

4.1 Fragestellung

Wie bereits beschrieben, besteht die Zielsetzung darin, die verschiedenen präsentierten state-of-the-art Verfahren in einem gemeinsamen Setting zu bewerten und miteinander zu vergleichen. Dabei soll geklärt werden, ob eines dieser Verfahren sich als nachweisbar besser herausstellt.

Genauer gesagt soll erst einmal herausgefunden werden, welches oben beschriebene distributionelle semantische Modell die Ähnlichkeit von Satzpaaren am genauesten klassifizieren kann. Dabei sollen alle Modelle mithilfe des SICK-Datensatzes trainiert werden. Dieser enthält erzählerische Elemente, wodurch die Daten nicht aus simplen Satzpaaren, sondern aus diversen Daten bestehen. Auf dieser Basis wird geprüft, welcher Algorithmus den Diskurs natürlicher Sprache am besten repräsentieren kann. Insbesondere soll darauf geachtet werden, wie die beiden Nachfolgearchitekturen von Doc2Vec (Doc2VecC und Sent2Vec) im Vergleich zueinander abschneiden und welches der beiden Modelle für diese Datensätze besser geeignet ist.

Zusätzlich soll die Leistungsfähigkeit von wpath mit Korpus-basierten Modellen mithilfe mehrerer verschiedener Datensätze verglichen werden; dies geschieht hinsichtlich Wort- als auch bezüglich Dokumentenähnlichkeit, sodass beides betrachtet werden kann. Ziel ist es, ausgehend auf den Erkenntnissen von (Zhu, et al., 2017) einen Vergleich zwischen den Verfahren der beiden Paradigmen (Korpus- und wissensbasierte semantische Ähnlichkeit) zu ermöglichen.

4.2 Auszuwählende Softwaretools

Für die Umsetzung von Systemen im Bereich NLP (und allgemein im Umfeld der künstlichen Intelligenz) hat sich in den letzten Jahren die multi-paradigmatische Programmiersprache „Python“ durchgesetzt (Zola, 2018) (Mathur, 2018). Diese Sprache überzeugt durch eine einfache Syntax, einen umfangreichen Community-Support sowie zahlreiche, umfassend dokumentierte und schnell zu installierende Softwarebibliotheken, die im Bereich NLP und maschinelles Lernen angeboten werden. Ein genereller Nachteil von Python sind die Unterschiede zwischen den Versionen Python 2 und Python 3. Quellcode, der in Python 2 ausgeführt werden kann, kann in der Version 3 aufgrund der unterschiedlichen Syntax zu Fehlern führen, was umständlich ist. Es wird im Rahmen der Hausarbeit ausschließlich mit Python 3 gearbeitet, da dies für eine perspektivische Weiterarbeit am Projekt der Thesis nur Sinn macht. Außerdem gibt es mittlerweile nahezu sämtliche, relevante Bibliotheken auch für Python 3.
Die Programmiersprache R, die gewissermaßen in Konkurrenz zu Python steht, ist insbesondere im Sektor NLP Python gegenüber stark unterlegen, weswegen diese trotz persönlicher Präferenz nicht gewählt wurde.
Im Gegensatz zu Alternativen wie z.B. der objektorientierten Programmiersprache „Java“ werden, abgesehen von Doc2VecCs Implementation in „C“, alle anderen in dieser Arbeit vorgestellten Modelle durch Python-Bibliotheken unterstützt. Deshalb ist es naheliegend, Python für die Umsetzung der Modelle zu verwenden. Richtigkeitshalber sollte hierbei jedoch ergänzt werden, dass viele Python-Bibliotheken Wrapper um C++ sind.

Um alle Modelle trainieren und evaluieren sowie alle Komponenten in einem gemeinsamen System orchestrieren zu können, soll in „Shell“ programmiert werden, da das System dann über die Kommandozeile gestartet werden kann und mithilfe von anderem Kommando wie z.B. *time* die Ausführungsdauer gemessen werden kann.
Alles in allem werden für die Realisierung des Systems die Programmiersprachen „Python“, „C“ und „Shell“ sowie dazugehörige Softwarebibliotheken beziehungsweise Kommandos

verwendet, wobei nur in Python und Shell eigene Skripte geschrieben werden, wohingegen C-Code nur genutzt wird. Lediglich Variablen eines bereits bestehenden Programms werden hier verändert.

4.3 Funktionale und nicht-funktionale Anforderungen

Im Folgenden sollen sowohl die gewünschten Funktionalitäten als auch die erforderlichen technischen Eigenschaften des Systems beschrieben werden.
Eines der Probleme, die durch das System gelöst werden sollen, ist die Gewährleistung der Vergleichbarkeit der Modelle. Diese sind an und für sich verschieden, müssen aber in eine Gestalt und Form gebracht werden, in der sie auf einer universalen Ebene komparabel werden.
Außerdem sollte es eine Schnittstelle geben, die die Parameter des Versuchs veränderbar macht. Nur so sind unkomplizierte Experimente möglich.
Aus diesen Erschwernissen lassen sich die in Tabelle 6 dargestellten Anforderungen an das System ableiten.
Da mehrere Computer im Rahmen der Thesis genutzt werden, muss der Quellcode auf mehreren Systemen reibungslos funktionieren, weswegen eine Optimierung beispielsweise auf die Windows Shell kontraproduktiv ist.
Aus diesen und ähnlichen Gründen leiten sich die folgenden Anforderungen ab.

Tabelle 6: Funktionale und nicht-funktionale Anforderungen an das zu entwickelnde System

	Erläuterung
Funktionale Anforderung	Die Ausführungsdauer beim Training jedes Modells wird gemessen und auf der Konsole ausgegeben.
	Die Ausführungsdauer bei der Evaluation der Modelle wird gemessen und auf der Konsole ausgegeben.
	Das Training und die Evaluation der Modelle soll über die Kommandozeile gestartet werden können.
	Die Ergebnisse der Metriken werden auf der Konsole angezeigt.
	Es wird auf der Konsole angezeigt, welche Metrik und welcher Datensatz verwendet wurden.
	Die Modellparameter können verändert werden.
	Der Korpus kann verändert (z.B. kann statt ein anderer Datensatz zur Evaluation verwendet werden)

	Die verwendeten Metriken können ausgetauscht werden.
	Alle distributionellen Modelle werden nacheinander mit dem gleichen Datensatz trainiert.
	Alle distributionellen Modelle werden nacheinander mit dem gleichen Datensatz evaluiert.
	Training und Evaluation können unabhängig voneinander ausgeführt werden.
Nicht-funktionale Anforderung	Die trainierten Modelle werden in separaten Dateien persistiert.
	Die Systemkomponenten können über die Kommandozeile installiert und kompiliert werden.
	Während der Programmausführung wird eine Fortschrittsanzeige zur Verfügung gestellt.
	Das System funktioniert auch auf anderen PCs mit einer Unix-Shell.
	Das Training kann durch Verwendung von Multithreading beschleunigt werden.

Die oben genannten Problemstellungen werden durch die Erfüllung der aufgezählten Anforderungen gelöst beziehungsweise gar nicht erst zu einem Hindernis.
Diese Systemanforderungen, die in Tabelle 6 in kompakter Form dargestellt sind, zeigen, dass das System nicht nur den Anwender über Konsolenausgaben Zwischenstände und Resultate informiert, sondern auch so gestaltet sein muss, dass Veränderungen beispielsweise an den Modellparametern vorgenommen werden können und einzelne Systembestandteile (z.B. die Evaluation) unabhängig von anderen Komponenten ausgeführt werden können. Das heißt, dass das System so konzipiert werden musss, damit es vor allem anpassbar und modular ist.

5 Entwurf

5.1 Fachliche Systemkonzeption

Damit die aufgelisteten Anforderungen erfüllt und die definierten Fragestellungen beantwortet werden können, soll in diesem Abschnitt zunächst erläutert werden, aus welchen fachlichen Komponenten das System aufgebaut ist und wie sie in Beziehung zueinanderstehen. Das geplante Gesamtsystem wird in zwei voneinander unabhängige Teilsysteme aufgeteilt, um alle in Abschnitt 3.2 vorgestellten Modelle zu bewerten. Diese werden in den folgenden Ausführungen als Experimente (1 und 2) bezeichnet.

5.1.1 Konzeption des ersten Experiments

Experiment 1 befasst sich mit dem Vergleich von Doc2VecC und Sent2Vec sowohl miteinander, als auch mit zwei klassischen Verfahren zur Bestimmung der Ähnlichkeit von Dokumenten (Doc2Vec und Tf-idf). Ausgehend von der Fragestellung (siehe Abschnitt 4.1) soll in diesem Experiment herausgefunden werden, ob beziehungsweise inwieweit die beiden state-of-the-art Modelle „Doc2VecC“ und „Sent2Vec“ besser beziehungsweise schlechter die Ähnlichkeit zweier Sätze beurteilen können als Tf-idf (siehe Abschnitt 2.2.2) und Doc2Vec, welche als sog. „Baseline“ dienen.
Auf dieser Grundlage werden die Modelle miteinander verglichen und das Ziel ist es, diese Leistungsfähigkeit der Baseline übertreffen. Eine Baseline kann eine Heuristik, eine zufallsbasierte Methode oder ein anderes state-of-the-art Modell sein. In diesem Experiment dient Tf-idf als triviales Verfahren, welches mit Doc2VecC und Sent2Vec verglichen worden soll, um herauszufinden, ob beziehungsweise inwieweit die beiden Modelle hinsichtlich der Dokumentenähnlichkeit besser abschneiden als simple Herangehensweisen. Doc2Vec ist die

zweite Baseline. Sie wird verwendet, um zu überprüfen, in welchen Kategorien (z.B. Trainingsdauer) das Ausgangsmodell (Doc2Vec) im Vergleich zu seinen Erweiterungen (Doc2VecC und Sent2Vec) anders abschneidet.

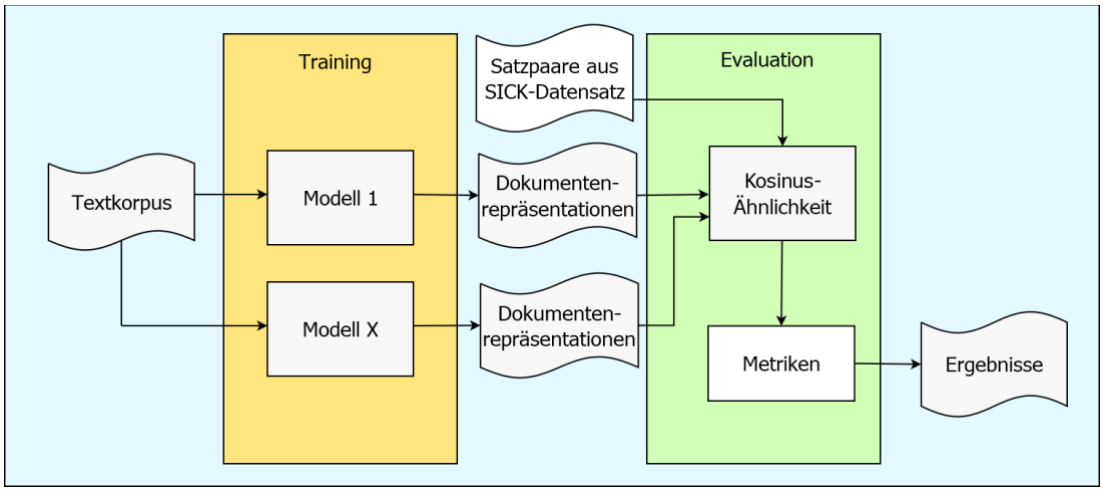


Abbildung 9: Vereinfachte Veranschaulichung des Datenflusses und der zu realisierenden Komponenten des 1. Experimentes

Abbildung 9 dient zur Veranschaulichung des modularen Aufbaus dieses Experiments und des Datenflusses. Wie man sehen kann, besteht das System aus zwei großen Komponenten („Training“ und „Evaluation“), die sich in Subkomponenten aufteilen lassen. Zunächst werden alle vier Modelle mit einem Textkorpus trainiert. Dieser Korpus umfasst u.a. eine Zufallsstichprobe von 1 000 000 Sätzen (entnommen aus Romanen) aus dem Trainingskorpus des LAMBADA-Datensatzes (Paperno, et al., 2016). Dieser ist eine Sammlung von narrativen Textabschnitten, welche die Eigenschaft teilen, dass Menschen in der Lage sind, die letzten Wörter (Zielwörter) zu erraten, wenn sie eine lange Passage lesen, aber nicht, wenn sie nur den letzten Satz vor dem Zielwort sehen. Der Datensatz wird jedoch nicht für den ursprünglich gedachten Zweck (nämlich die Vorhersage des Zielworts) verwendet, sondern lediglich zum Lernen von Dokumentenrepräsentationen. Davon abgesehen wird der Korpus um die knapp 6000 Sätze des sog. „SICK-Datensatzes“ (Baroni, et al., 2014) ergänzt.

Tabelle 7 zeigt einen kleinen Auszug des Trainingskorpus. Da es sich hier um unüberwachte Algorithmen handelt, werden keine Labels o.Ä. zum Lernen benötigt. Nach dem Training werden für jedes Modell jeweils die Vektorrepräsentationen der gelernten Dokumente in separaten Dateien persistiert.

she heard her son-in-law 's name echo through the halls .
she had trusted him , given her permission , but she had n't known the consequences .
i flung it away on autopilot , just as i 'd shake off an earwig that chanced to light on my hand .
he was annoyed that benjamin did n't give him the first option to buy and was even more incensed when his will stated that elle would be the ceo at the completion of her college studies .

Tabelle 7: Ein Auszug aus dem Trainingskorpus von (Paperno, et al., 2016). Dieses besteht hauptsächlich aus erzählerischen Passagen, welche aus dem „BookCorpus“ von (Zhu, et al., 2015) extrahiert wurden.

Anschließend werden diese Dateien mithilfe des „SICK-Datensatzes“ evaluiert[12]. Dieser Datensatz, welcher eine Vergleichsgrundlage für das Testen von distributionellen semantischen Modellen ermöglicht, besteht aus rund 10000 englischen Satzpaaren, die jeweils in Bezug auf ihre Bedeutungszusammenhänge und Folgerungsbeziehungen durch Menschenhand annotiert worden sind.

Paar-ID	Satz A	Satz B	Score
1	A group of kids is playing in a yard and an old man is standing in the background	A group of boys in a yard is playing and a man is standing in the background	4,5
97	A lone biker is jumping in the air	A man is jumping into an empty pool	1,5
108	A player is throwing the ball	A player is running with the ball	3,5

Tabelle 8: Beispiel für Paarsätze und deren Ähnlichkeitswert aus dem SICK-Datensatz.

Tabelle 8 zeigt einige exemplarische Elemente dieses Datensatzes. Man erkennt, dass das Bewertungsprinzip für ein Satzpaar sich von „1“ für komplett verschiedene Inhalte bis hin zu „5“ für eine sehr starke Ähnlichkeit spannt. Dieser Ähnlichkeitswert (auch bekannt als „Score“) ist daher eine reell wertige Zahl in diesem Intervall. Einige Sätze kommen in mehreren Satzpaaren, sodass der Datensatz aus ca. 6000 verschiedenen Sätzen besteht, welche allesamt für das Training der Modelle verwendet werden. Dadurch, dass das Trainingskorpus so umfangreich ist, machen diese Sätze nur einen geringen Anteil aus und das Risiko von sogenanntem „Overfitting“, wird minimiert.
Overfitting beschreibt einen Vorgang, bei dem beim Trainieren eines Modells der Daten ebendieses Modell sich zu sehr an die gegebenen Daten angepasst hat, dadurch für

[12] die verwendeten Metriken werden im nächsten Kapitel genauer beschriebenen

allgemeinere Fälle immer unbrauchbarer wird. Da das Modell in der Grundidee für jegliche englische Satzpaare funktionieren soll, nicht nur für die gegebenen Datensätze, ist es höchst unerwünscht. Noch stärker, hier nicht gegeben, wäre das Overfitting, wenn alle Beispieldaten derselben Domäne (beispielsweise Pharmazie) angehörten. Hier würde das Modell sich zu sehr an das entsprechende, pharmazeutische Vokabular „gewöhnen“.

5.1.2 Konzeption des zweiten Experiments

Dieses Experiment befasst sich mit der Evaluation von „wpath“ (siehe Abschnitt 3.2.3). und evaluiert dieses Modell mithilfe diverser Datensätze und den bereits vorliegenden Benchmarks für Word2Vec. Mithilfe von drei verschiedenen Datensätzen, sowie einer Taxonomie, werden zunächst Wortähnlichkeiten berechnet und anschließend wird die die im nächsten Kapitel erläuterte Testmethodik verwendet, um die Leistungsfähigkeit von wpath anhand von Metriken mit Word2Vec zu vergleichen. In Abbildung 10 wird der geplante Ablauf sowie die Komponenten mitsamt ihren Beziehungen verbildlicht dargestellt.

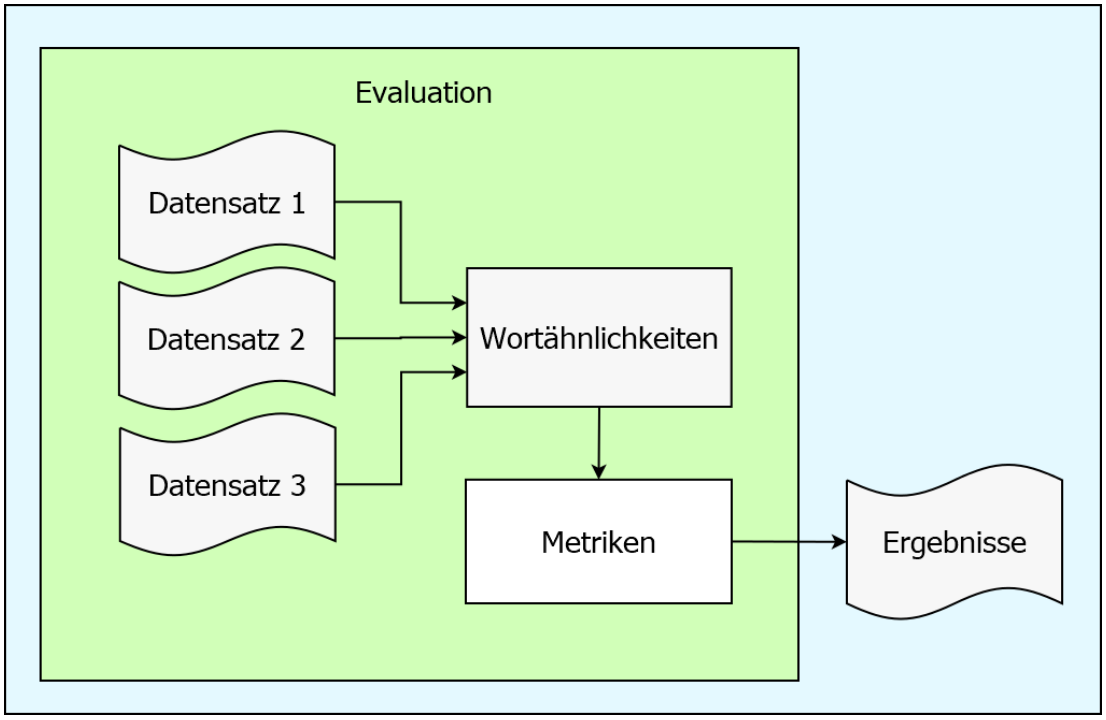


Abbildung 10: Vereinfachte Veranschaulichung des Datenflusses und der Komponenten des 2. Experimentes

Die gesamten Korpora, welche in Tabelle 9 exemplarisch und verkürzt dargestellt werden, bestehen aus den folgenden Datensätzen: SimLex-999 ist ein Datensatz zur Bewertung von Modellen, die die Bedeutung von Wörtern und Konzepten erlernen. Er ermöglicht es zu messen, wie gut Modelle semantische Ähnlichkeit erfassen. Die durchschnittlichen menschlichen Ähnlichkeitswerte (Scores) eines Wortpaares befanden sich ursprünglich im Intervall [0,6] und sind auf [0,10] gemappt wurden, um anderen Datensätzen wie WordSim-353 zu entsprechen (Hill, et al., 2014). Je höher der Score, desto höher ist die Ähnlichkeit zwischen zwei Wörtern.

SimVerb-3500 ist ein Datensatz zur Bestimmung der semantischen Ähnlichkeit von Verben. Er enthält 3500 Verbpaare mit Bewertungen auf einer Skala von 0 (keine Ähnlichkeit) bis 10 (sehr hohe Ähnlichkeit) (Gerz, et al., 2016). Die Definition der Scores ist genauso wie bei SimLex-999.

Wordsim-353 ist eine Textsammlung, die zwei Datensätze englischer Wortpaare sowie vom Menschen zugewiesene Ähnlichkeitswerte (Finkelstein, et al., 2002) enthält. Für dieses Experiment wird eine Kombination beider Datensätze verwendet. Im Gegensatz zu SimLex-999 wird eher nicht auf die semantische Ähnlichkeit, sondern auf die Verwandtschaft beziehungsweise den Grad des Zusammenhangs zwischen zwei Worten geschaut. Dieser Unterschied wird in Tabelle 9 verdeutlicht. Die Scores befinden sich zwischen 0 und 10 (0 = Wörter sind überhaupt nicht verwandt, 10 = Wörter sind sehr eng verwandt).

Datensatz	Wort A	Wort B	Score
SimLex-999	clothes	closet	1,96
Wordsim-353	clothes	closet	8,00
SimVerb-3500	tear	fold	0,66

Tabelle 9: Exemplarische verkürzte Auszüge aus den Datensätzen, die zur Evaluation von wpath verwendet werden. Die restlichen Spalten werden hierfür nicht benötigt.

Der detaillierte Aufbau der jeweiligen Datensätze kann in den referenzierten Papern nachgelesen werden. Da es sich hierbei um ein wissensbasiertes Modell handelt, welches mit Taxonomien arbeitet, ist kein Training des Modells erforderlich und daher befindet sich alles in der Hauptkomponente „Evaluation“.

5.2 Technische Konzeption

Zur technischen Umsetzung der Modelle werden verschiedene Softwarebibliotheken inklusive bereits existierendem Quellcode verwendet:

- Das Tf-idf-Modell wird mithilfe der Python-Bibliothek „scikit learn“ [13] umgesetzt. Hierfür wird der sog. „TfidfVectorizer“ verwendet, mit dem eine Sammlung von Dokumenten in eine Matrix von Tf-idf-Features umgewandelt wird.
- Zur Umsetzung des Doc2Vec-Modells ist die Python-Bibliothek „Gensim“ [14] geeignet. Es wird ein PV-DBOW-Modell mit einer Vektorgröße von 100 instanziiert und 20-mal über das Trainingskorpus iteriert. Die Mindestanzahl der Häufigkeit eines Wortes beträgt 1, um Wörter mit sehr wenigen Vorkommnissen zu verwerfen. Bei der Auswahl der Parameter dient das Paper von (Lau, et al., 2016) als Orientierungshilfe. Die Sätze, die im Korpus vorkommen, sind relativ kurz, so dass in diesem Fall 100 Dimensionen ausreichen. Generell wird für die Vektorgröße in der Regel 100 bis 300 (maximal 500) Dimensionen verwendet (vgl. (Lau, et al., 2016)).
- Für Doc2vecC gibt es aktuell nur eine Implementation in C [15]. Der Beispielcode wird mitsamt aller vorkommenden Default-Parameterwerte übernommen (bis auf die Fenstergröße, welche verkleinert ist) und für diesen Datensatz angepasst.
- Das Sent2Vec-Modell wird mithilfe der gleichnamigen Softwarebibliothek[16] für 20 Epochen trainiert. Die maximale Länge der Wort-N-Gramme ist 2 und die Dimension von Wort- und Satzvektoren beträgt 200.
- Wpath wird auf der Basis der Experimente von (Zhu, et al., 2017), welche als „Python Notebook“ [17] präsentiert werden, und mittels der Python-Bibliothek „Sematch“ [18] umgesetzt

[13] https://scikit-learn.org (letzter Zugriff: 03.01.2019)
[14] https://radimrehurek.com/gensim/ (letzter Zugriff: 03.01.2019)
[15] https://github.com/mchen24/iclr2017 (letzter Zugriff: 03.01.2019)
[16] https://github.com/epfml/sent2vec (letzter Zugriff: 03.01.2019)
[17] https://github.com/gsi-upm/sematch/blob/master/word_sim_evaluation.ipynb (letzter Zugriff: 03.01.2019)
[18] http://gsi-upm.github.io/sematch/ (letzter Zugriff: 03.01.2019)

Eine Liste aller Parameter und deren Bedeutung befindet sich auf der jeweiligen referenzierten Webseite des Modells. Die Modellparameter wie z.B. die Anzahl der Dimensionen der Vektoren (um die Repräsentationsfähigkeit des Modells zu erhöhen), die Art der Architektur (bei Doc2Vec: DBOW oder DM) sowie die Größe der Korpora werden in mehreren Durchläufen experimentell verändert. Anschließend werden die Modelle neu trainiert und evaluiert und die Auswirkungen auf die Metriken dokumentiert.

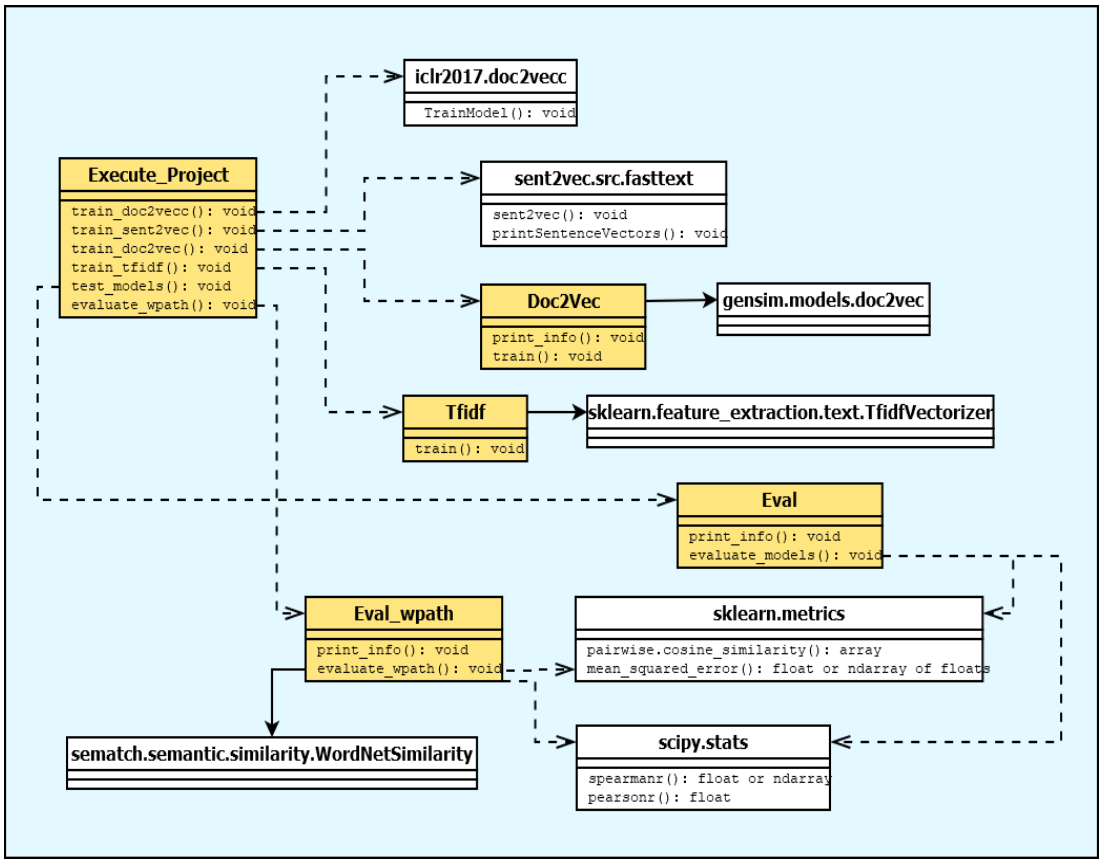


Abbildung 11: Das UML-Klassendiagramm des Systems, welches die Klassen, ihre Beziehungen und wichtigsten Methoden darstellt. Orange gefärbte Klassen sind zu implementieren und die anderen werden bereits als Softwarebibliothek bereitgestellt.

Zur Verdeutlichung der umzusetzenden Softwareelemente und ihren Beziehungen zueinander soll ein UML-Klassendiagramm herangezogen werden (siehe

Abbildung 11). Aus dieser Darstellung geht hervor, dass das System (und damit die beiden Experimente) über eine Klasse zentral gesteuert werden soll. Dabei wird zwischen Methoden unterschieden, die sich entweder auf das Training oder die Evaluation der Modelle beziehen. Des Weiteren ist der Abbildung zu entnehmen, dass Klassen aus Softwarebibliotheken komplett übernommen oder zumindest zur Implementierung einer Klasse verwendet werden. Es gibt keine wechselseitigen Abhängigkeiten zwischen Klassen, was die Komplexität des Systems verringert und seine Anpassbar-, Austauschbar- und Wartbarkeit vereinfacht.

Dies schließt die Konzeption des Systems ab. Auf dieser Grundlage werden die beiden Experimente technisch umgesetzt und abschließend ausgewertet.

6 Umsetzung

Durch das vielfältige Angebot an gut dokumentierten Softwarebibliotheken und bereits vorhandenem Quellcode ist es einfach gewesen, die hier vorgestellten Modelle in Python umzusetzen. Insbesondere wurde dies durch den jeweiligen bereitgestellten Beispielcode ermöglicht. Darüber hinaus ist durch die Verwendung von Shell-Skripten und -Kommandos die Erfassung der Laufzeit, das Bilden von Pipelines und Umsetzung der erforderlichen Konsolenausgaben vereinfacht worden.

Des Weiteren ist anzumerken, dass der Entwurf (vgl. Abbildung 11) von der Grundfunktion übernommen, aber nicht 1:1 umgesetzt wurde, da es sich als simpler herausgestellt hat, anstelle von definierten Klassen lediglich die jeweiligen Methoden mittels Skripte zu realisieren. Zum Beispiel ist die Klasse „Execute_Project“ ein Shell-Skript, welches nur die entsprechenden Methoden beziehungsweise Skripte aufruft und zum Ausführen des ersten Experimentes dient.

Die Quellcodezeilen des Shell-Skripts sind untenstehend aufgeführt. Zunächst wird das Korpus definiert, indem der entsprechende Pfad in einer Variablen „CORPUS_FILE“ angegeben wird. Dann werden die einzelnen Modelle trainiert, indem die jeweiligen Skripte aufgerufen werden. Diese Shell-Skripte führen die jeweiligen Algorithmen mit darin festgelegten Modellparametern aus. Schließlich wird das Evaluationsskript in Python aufgerufen. Dieser Quellcode ist beispielhaft für das gesamte System und soll seinen modularen Aufbau, welcher die Verständlichkeit des Quellcodes erhöht und die Kopplung zwischen den Modulen verringert, verdeutlichen.

```
1.  #! /bin/bash
2.
3.  CORPUS_FILE=corpora/corpus.txt
4.  chmod +x exec_project.sh train_doc2vecc.sh train_doc2vec.sh train_sent2vec.
    sh train_tfidf.sh
5.
6.  echo "Extracting corpora... "
7.  tar xf corpora/corpora.tar.xz --directory=corpora/
8.
9.  echo "Training Doc2VecC with corpus file " $CORPUS_FILE
10. . train_doc2vecc.sh
11. echo "Training Sent2Vec with corpus file " $CORPUS_FILE
12. . train_sent2vec.sh
13. echo "Training Doc2Vec with corpus file " $CORPUS_FILE
14. . train_doc2vec.sh
15. echo "Training Tf-idf with corpus file " $CORPUS_FILE
16. . train_tfidf.sh
17.
18. echo "Training done. Evaluating models..."
19. python main.py
20.
21. echo "Evaluating done."
```

Negativ aufgefallen ist das bereits in Abschnitt 4.2 angesprochene Problem, welches bei den unterschiedlichen Python-Version auftreten kann. Da die Bibliothek „Sematch“ nur für Python 2.7 angeboten wird, gibt es Module, die mit Version 3.7 nicht kompatibel sind und daher nicht funktionieren. Dieses Problem wurde dadurch gelöst, indem der Sematch-Quellcode an entsprechenden Stellen manuell für Python 3 angepasst wurde. Das Problem war daher keine Schwäche des Systems, sondern eher der Tatsache geschuldet, dass die Entwickler von Sematch fehlerhaften Code veröffentlicht haben. Ansonsten sind bei der Realisierung des Systems keine weiteren Probleme aufgetreten, die den Fortschritt dieser Arbeit behindert haben.

```
imon:~/sts-eval$ time . train_doc2vecc.sh
iclr2017/doc2vecc.c: In function ‘ReadVocab’:
iclr2017/doc2vecc.c:322:5: warning: ignoring return value of
    ‘fscanf’, declared with attribute warn_unused_result [
    -Wunused-result]
     fscanf(fin, "%lld%c", &vocab[a].cn, &c);
     ^~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Starting training using file ./corpora/corpus.txt
Vocab size: 26962
Words in train file: 14660059
Alpha: 0.000005  Progress: 100.00%  Words/thread/sec: 229.26k
     finish embedding training
writing sentence vector ...

real    5m52,072s
user    21m39,481s
sys 0m2,936s
```

Abbildung 12: Eine Darstellung der Konsolenausgabe während des Trainings von Doc2VecC. Neben einer Fortschrittsanzeige und der Anzahl Wörter im Korpus, wird auch am Ende die Ausführungsdauer gemessen und angezeigt.

Alles in allem konnten alle Systemanforderungen zufriedenstellend erfüllt werden. Dies wird besonders in Abbildung 12 erkennbar. Mithilfe der Konsolenausgabe werden dem Anwender Informationen über die Trainingszeit, verwendeten Korpus, usw. über die Konsole mittgeteilt, wodurch bereits einige funktionale Anforderungen aus Abschnitt 0 durch Verwendung von Shell-Kommandos wie *time* und den individuellen Ausgaben der jeweiligen Modelle abgedeckt werden.

7 Evaluation des Systems

7.1 Testmethodik

Für dieses Experiment werden lediglich die Satzpaare und der Ähnlichkeitswert verwendet, indem pro Modell die Kosinus-Ähnlichkeit (siehe Abschnitt 2.3.2) eines jeden Satzpaars berechnet wird. Mit dieser welche ein Maß dafür ist, wie zwei Vektoren in Bezug auf ihren Vektorwinkel zueinanderstehen (unabhängig von ihrer Größe). Zwei Vektoren sind demnach identisch, wenn deren Kosinus-Ähnlichkeit 1 ist und -1, wenn sie entgegengesetzt gerichtet sind. Wenn negative Gewichte ausgeschlossen werden, liegt dieser Wert zwischen 0 und 1.
Gleichzeitig wird die Bewertungsskala so normiert, dass die Ähnlichkeitswerte der Satzpaare zwischen 0 und 1 liegen. Wie beschrieben, Diese Werte beschreiben die Ähnlichkeit der Dokumentenvektoren. Schließlich wird auf dieser Grundlage die Korrelation bestimmt. Hierfür werden folgende drei Metriken betrachtet: Der Pearson-Korrelationskoeffizient (Pearson, 1895), Spearman-Rangkorrelationskoeffizient (Spearman, 1904) und die mittlere quadratische Abweichung 19.
Der Pearson- und Spearman-Korrelationskoeffizient ist eine Zahl zwischen -1,0 und 1,0, die angibt, inwieweit zwei Variablen linear miteinander zusammenhängen. In diesen Experimenten gilt: Je näher dieser Wert an 1,0 (positiver linearer Zusammenhang) ist, desto besser ist das Modell.
Die mittlere quadratische Abweichung misst in diesem Szenario die durchschnittliche quadratische Differenz zwischen Modellwerten und menschlichen Werten. Je kleiner der Wert ist, desto besser ist das Modell.

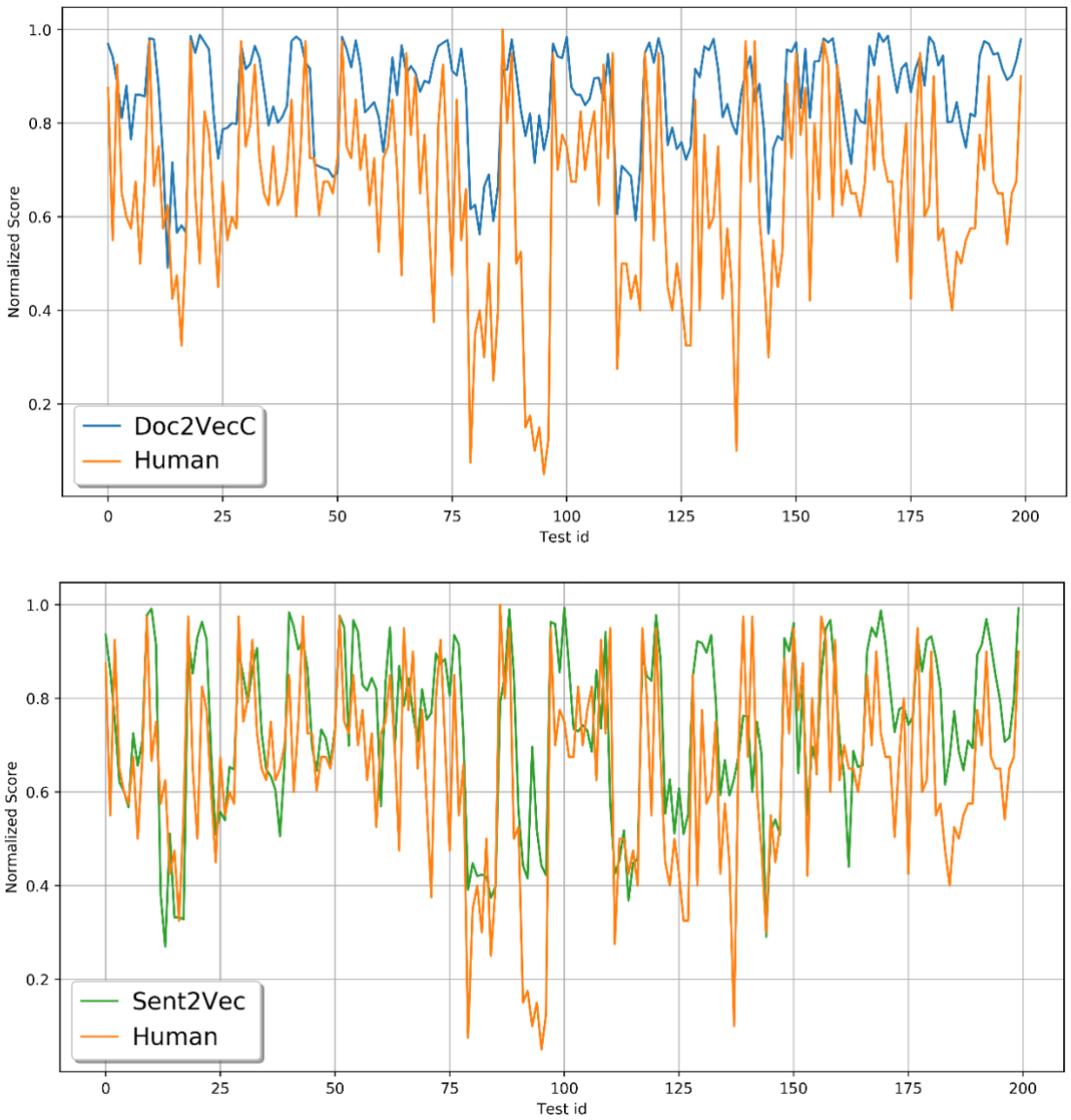
[19] nachzulesen in (Rüschendorf, 2014)

Alle Experimente sind auf einem PC mit einem Intel Core i7 (1,8 bis 4 GHz) mit vier Prozessorkernen, 16GB Arbeitsspeicher und dem Betriebssystem Ubuntu 18.04.1 LTS durchgeführt worden.

7.2 Auswertung der Ergebnisse von Experiment 1: Doc2VecC & Sent2Vec

Trotz mehrerer Durchläufe mit unterschiedlich gewählten Parametern, kam es zu keinen nennenswerten Veränderungen der Modelle zueinander, so dass im Folgenden lediglich die Ergebnisse eines repräsentativen Durchlaufs vorgestellt werden.
Zur Visualisierung der Ergebnisse sind in Abbildung 13 zwei Diagramme der ersten 200 Stichproben des SICK-Datensatzes und deren Ergebnisse (Mensch und Modell) enthalten. Jede Stichprobe (nummeriert von 0-199) ist ein Satzpaar (als Vektoren) und die Zahl von 0 bis 1 (y-Achse) für jede Stichprobe ist der normierte Wert beziehungsweise der Grad der Ähnlichkeit (Score) zwischen diesen Sätzen, der durch einen Menschen und durch das Modell (Doc2Vec/Sent2Vec) angegeben wird.

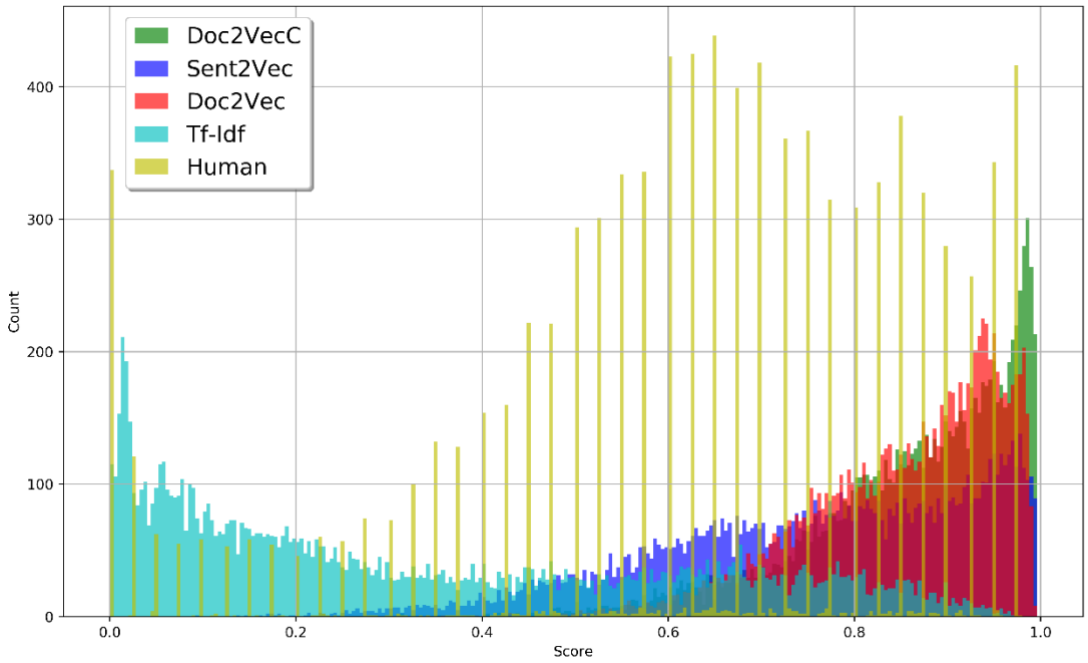
Abbildung 13: Korrelationen der Kosinus-Ähnlichkeiten von Vektoren für eine kleine Stichprobe



Wie in Abbildung 13 zu sehen ist, schätzt Doc2VecC die meisten Satzpaare als ähnlich ein, wohingegen Sent2Vec auch größere Ausschläge „nach unten“ hat und dadurch genauer mit den menschlichen Werten übereinstimmt. Diese Übereinstimmung ergibt sich aus der Ähnlichkeit der Form der Graphen: Je ähnlicher sich diese aussehen, desto ähnlicher sind offensichtlich ihre Werte. Vermutlich könnte das daran liegen, dass Sent2Vec für kurze Sätze besser geeignet ist (vgl. Abschnitt 3.2.2).

Abbildung 14 zeigt die Verteilung der Werte für jedes Modell im Bereich von 0,0 bis 1,0. Der „menschliche Score“ (die Ähnlichkeitswerte des SICK-Datensatzes) wurde von 1,0 bis 5,0 auf 0,0 bis 1,0 normalisiert. Beispielsweise hat tf-idf 100 Werte im Bereich von 0,0 bis 0,004. Außerdem gibt es z.B. 350 menschliche Scores mit dem Wert 0 (keine Ähnlichkeit). „Count“ beschreibt daher die Häufigkeit des Vorkommens. Das Histogramm zeigt, wie die Scores der einzelnen Modelle verteilt sind und ob ein Modell eher dazu neigt, die Satzpaare als eher ähnlich oder eher unterschiedlich (in Relation auf die menschlichen Scores) zu klassifizieren. Außerdem bietet es eine Möglichkeit, die Scores für jedes Modell in einem einzigen Diagramm zu visualisieren.

Abbildung 14: Ein Histogramm der Ergebnisse für jedes Modell. Die normierten Ähnlichkeitswerte befinden sich auf der x-Achse und auf der y-Achse werden die Häufigkeiten angegeben.



Wie bereits erwähnt, wurden ca. 1 Millionen Sätze für das Training verwendet (knapp 66 MB an Trainingsdaten. Die Verwendung einer großen Anzahl von Sätzen des Korpus verbessert zwar die Pearson-Korrelation aller Modelle, erfordert aber mehr Zeit und Ressourcen für die Berechnung. Während des Trainings haben wir die gleichen Argumente bei Doc2VecC/Doc2Vec verwendet. Darüber hinaus dauerte es im Falle von Sent2Vec 3 Minuten und 18 Sekunden, um das Modell zu trainieren und 1 Minute und 14 Sekunden, um die Dokumentenvektoren zu generieren.

Es dauerte insgesamt 2 Minuten und 40 Sekunden bis alle Modelle evaluiert wurden. Tabelle 10 fasst alle Ergebnisse zusammen.

Tabelle 10: Eine Darstellung der Evaluationsergebnisse von semantischen Modellen zum Vektorisieren von 1006077 Sätzen. Die besten Werte einer Spalte sind fett gedruckt.

Modell	Pearson-Korrelations-koeffizient	Spearman-Rangkorrelations-koeffizient	mittlere quadratische Abweichung	Trainingsdauer
Doc2VecC	**0,6510**	0,5727	0,0946	5m 52s
Sent2Vec	0,5682	0,5287	**0,0615**	4m 33s
Doc2Vec (Baseline)	0,5837	0,5583	0,0788	22m 42s
Tf-idf (Baseline)	0,5419	**0,5796**	0,1517	**2m 5s**

Anhand von Tabelle 10 sieht man, dass kein Modell über alle Metriken hinweg deutlich überlegener als die anderen Modelle ist. Genauer gesagt, schneiden alle fast gleich gut ab. Trotzdem fällt insbesondere Doc2Vec mit seiner deutlich längeren Trainingsdauer auf, welche auf die in Abschnitt 3.2.1 und 3.2.2 erläuterte Funktionsweise zurückzuführen ist. Zwar hat Tf-idf in zwei verschiedenen Spalten die besten Werte, jedoch in den beiden anderen Spalten deutlich am schlechtesten abgeschnitten.
Bezüglich der beiden Korrelationskoeffizienten (Pearson und Spearman) konnte sich Doc2VecC am besten absetzen, benötigte jedoch hierbei die längste Trainingszeit. Dennoch muss man beachten, dass dieses Modell im Gegensatz zu Doc2Vec neue Dokumentenrepräsentationen effizienter generiert, was in der Tabelle nicht berücksichtigt wurde. Sent2Vec hat zwar in der Stichprobe besser agiert als Doc2VecC, ist jedoch über den gesamten Datensatz hinweg nur bei der mittleren quadratischen Abweichung hervorgestochen.

Insgesamt überwiegen bei Doc2VecC die Vorteile am stärksten als die Nachteile und deswegen kann man schlussfolgern, dass dieses Modell (insbesondere aufgrund des hohen Pearson-Werts) am besten abschneidet. Abschließend ist festzustellen, dass Doc2VecC am besten abschneidet, gefolgt von Sent2Vec und dahinter Doc2Vec. Wie erwartet landet Tf-idf auf den letzten Platz. Die Durchläufe mit veränderten Modellparametern bestätigen diese Schlussfolgerung. Des Weiteren ging aus dem Experiment hervor: Je größer das Korpus ist, desto mehr neigt es sich zu diesem Ranking.

7.3 Auswertung der Ergebnisse von Experiment 2: wpath

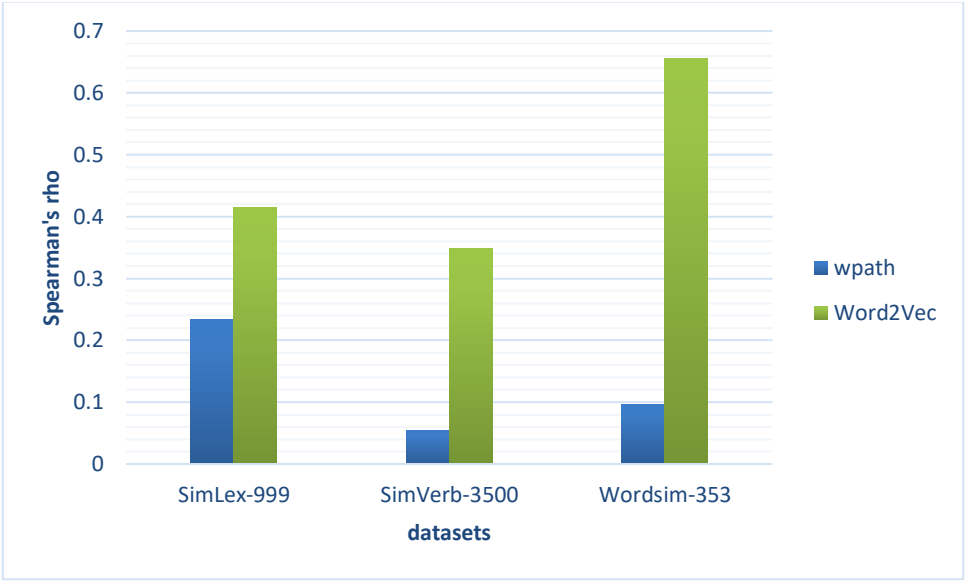
Tabelle 8 zeigt die Ergebnisse der Evaluation. Wie zu sehen ist, sind die Korrelationen bei jedem Datensatz schwach und spaltenweise variieren die Korrelationswerte relativ stark voneinander. Bei der mittleren quadratischen Abweichung gibt es nur leichte Unterschiede.

Tabelle 11: Eine tabellarische Darstellung der Evaluation von wpath mithilfe dreier Datensätze.

Datensatz	Pearson-Korrelations-koeffizient	Spearman-Rangkorrelations-koeffizient	mittlere quadratische Abweichung
SimLex-999	0,2870	0,2331	24,2465
SimVerb-3500	0,1610	0,0548	24,1013
Wordsim-353	0,1907	0,0959	24,8711

Vergleicht man dieses Modell mit den vorliegenden Benchmarks anderer semantischer Modelle, so stellt man fest, dass wpath schlechter abschneidet als z.B. Word2Vec. Trainierte Word2Vec-Modelle erreichen eine Spearman-Korrelation von 0,414 mit SimLex-999, 0,655 mit Wordsim-353 (Hill, et al., 2014) sowie 0,348 mit SimVerb-3500 (Gerz, et al., 2016). Eine Veranschaulichung dieses Vergleichs ist in Abbildung zu sehen.

Abbildung 15: Darstellung des Vergleichs von Spearman-Rangkorrelationskoeffizienten zweier Modelle zur Bestimmung der Wortähnlichkeiten für drei verschiedene Datensätze



Nichtsdestotrotz ist hier positiv anzumerken, dass wpath schon nach wenigen Sekunden die Ergebnisse liefert und dadurch deutlich schneller als Word2Vec ist, da u.a. kein aufwändiges Training durchgeführt wird.

7.4 Fazit

Hinsichtlich der Beantwortung der Fragestellungen aus Abschnitt 4.1 lässt sich Folgendes schlussfolgern: Auf der einen Seite ist es gelungen, ein System zu entwickeln, welches Doc2VecC und Sent2Vec und mit ihrem Vorgängermodell Doc2Vec vergleicht, um das leistungsfähigste Modell für dieses Szenario zu bestimmen. Mehrere Durchläufe mit unterschiedlichen Modellparametern haben gezeigt, dass Doc2VecC in dieser Arbeit am besten abschneidet und die bessere Nachfolgearchitektur von Doc2Vec ist. Immerhin konnte Sent2Vec den zweiten Platz im Ranking aller Modelle belegen. Es wurde sogar bestimmt, welches der beiden state-of-art Modelle in welcher Kategorie beziehungsweise unter welchen Bedingungen besser abschneidet als das andere. Beispielsweise hat sich Doc2VecC bezüglich der Pearson-Korrelation am deutlichsten von der Konkurrenz absetzen können und dadurch unter Beweis gestellt, dass es das leistungsstärkste Modell ist, wohingegen Sent2Vec hinsichtlich der menschlichen Scores des SICK-Datensatzes die geringste Gesamtabweichung erzielt hat, was ebenfalls für die Leistungsfähigkeit dieses Modells spricht. Insgesamt haben sich die beiden distributionellen state-of-the-art Verfahren nur knapp durchsetzen können, was hervorhebt, dass die konventionellen Algorithmen zumindest in einigen Kategorien wie z.B. bei der Spearman-Korrelation mithalten können und die die neuartigen Modelle im Bereich der semantischen Textähnlichkeit keine bahnbrechend besseren Ergebnisse erzielen.

Auf der anderen Seite ist es möglich gewesen wpath mit Korpus-basierten Verfahren zu vergleichen, jedoch war dies nur auf der Ebene von Wortähnlichkeiten möglich. Weder war es im Rahmen dieser Arbeit möglich, wpath hinsichtlich seiner Dokumentenähnlichkeit zu bewerten, noch konnten alle vorgestellten state-of-art Modelle in einem gemeinsamen Setting miteinander verglichen werden. Das abschließende Kapitel „Abschluss & Ausblick“ geht auf diese Punkte noch einmal näher ein. In dem Experiment konnte zwar wpath für alle Datensätze schnell angewandt werden, jedoch hat das Modell insbesondere bezüglich der Pearson-Korrelation deutlich schlechter abgeschnitten als Word2Vec. Es sprechen deshalb mehr Argumente für die Bestimmung von Wortähnlichkeiten Korpus- statt die leistungsschwächeren wissensbasierten Methoden zu verwenden und den zeitlichen Mehraufwand und die erhöhte Komplexität dafür in Kauf zu nehmen.
Nichtsdestotrotz konnten die Experimente genug Auskunft geben, um zumindest den Großteil der definierten Fragen aus Abschnitt 4.1 zu beantworten.

8 Abschluss & Ausblick

8.1 Abschließende Bemerkung

Es gibt verschiedene Möglichkeiten natürlichsprachige Texte nach ihrer Ähnlichkeit zueinander zu beurteilen. Die beiden bekanntesten Paradigmen sind Korpus- und wissensbasierte Methoden und Modelle, die sich beide mit der Semantik von Texten befassen. In dieser Arbeit wurden einige state-of-the-art Modelle aus beiden Bereichen in mehreren Experimenten verglichen und diese Ergebnisse im Rahmen der Fragestellungen untersucht und dabei bewertet.

Diese Experimente konnten deutlich aufzeigen, dass die moderneren und neuen distributionellen Modelle *Sent2Vec* und *Doc2VecC* bezüglich der Metriken nur marginal besser agieren als das Baseline-Modell Doc2Vec, auf welchem sie basieren. Hierbei ist jedoch anzumerken, dass Doc2Vecs im Voraus bekannte und beschriebene Schwäche (siehe Abschnitt 3.2.1) in Form einer sehr langen Laufzeit nicht von der Hand zu weisen war. Überraschenderweise konnte ein simples Modell wie Tf-idf in manchen Kategorien (bspw. Trainingsdauer) erstaunlich gute Ergebnisse erzielen. Diese gesamten Erkenntnisse sind dann in einem Ranking festgehalten worden, in dem alle Feststellungen zu sehen sind.

Der Algorithmus *wpath* mag nach (Zhu, et al., 2017) zwar im Vergleich zu anderen wissensbasierten Modellen besser abschneiden, jedoch vermochte er nicht mit den Benchmarks des Korpus-basierten Word2Vec-Modells mitzuhalten. Nichtsdestotrotz bieten wissensbasierte Modelle durch ihre simplere Architektur und Schnelligkeit einige Vorteile, die hier positiv anzumerken sind. Sollte eine Notwendigkeit bestehen, ein schnelles,

performantes System aufsetzen, führen nach den gemachten Erfahrungen mehrere Gründe zu *wpath*.
Die beiden Experimente bestätigen die Erkenntnisse von (Elsafty, 2017) und (Alvarez, 2017), welche ähnliche Versuche durchgeführt haben und ebenso zu dem Schluss gelangten, dass Doc2VecC bessere Ergebnisse liefert. Zusätzlich liegen im Bereich semantischer textueller Ähnlichkeit kaum state-of-the-Art Verfahren vor. Die Methoden, die bereits existieren, sind nicht in der Lage, erheblich besser als Baseline-Modelle abzuschneiden.

Alles in allem sollte man bei der Beurteilung der Modelle nicht nur die Evaluationsergebnisse der gewählten Metriken berücksichtigen, sondern auch die Art der Ähnlichkeit, nach der die Texte verglichen werden sollen sowie die Stärken und Schwächen, die sich bei einer bestimmten Modellarchitektur ergeben. Genau dies ist im Rahmen dieser Arbeit geschehen: Es wurde nicht nur eines, sondern beide bekannten Systeme genutzt und diese wurden miteinander verglichen. Außerdem wurden innerhalb der beiden Strategien diverse Gesichtspunkte betrachtet, um so einen möglichst akkuraten Vergleich abzubilden. Auf diese Weise konnten die state-of-the-art-Verfahren am gerechtesten gegenübergestellt werden.

8.2 Ausblick

Im Anschluss an diese Arbeit kann bei den Experimenten angeknüpft und diese auf verschiedene Art und Weise erweitert werden. Die in den folgenden Kurzabschnitten beschriebenen Vorschläge bieten daher einige Anhaltspunkte, um solche Experimente fortzuführen.

8.2.1 Gemeinsames Experiment

Ursprünglich ist geplant gewesen, ein gemeinsames Experiment durchzuführen und wpath als Vergleichsmodell oder zumindest in Kombination mit einem anderen Modell zu implementieren. Leider ist dies aufgrund der höheren Komplexität nicht möglich gewesen, da Korpus- und wissensbasierte Modelle zu unterschiedlich in ihrem Aufbau und ihrer Vorgehensweise sind, so dass z.B. bereits die Suche nach passenden Datensätzen für das Training und die Evaluation sich als schwierig erwiesen hat. Deswegen wurde wpath nur zur

Bewertung von Wortähnlichkeiten verwendet.[20] Um diese Arbeit fortzuführen, wäre es interessant, hier anzusetzen und die Experimente so zu gestalten, dass genau dieser Vergleich von Modellen, unabhängig davon, ob sie Korpus- oder wissensbasierter Art sind, möglich ist.

8.2.2 Berücksichtigung weiterer Modelle

Eine weitere interessante Erweiterung wäre, genauso wie (Elsafty, 2017) die Modelle miteinander zu kombinieren (z.B. LDA, Tf-idf und Doc2VecC) sowie Vorverarbeitungsschritte wie die Stammformreduktion anzuwenden. Außerdem kann man bereits trainierte Modelle verwenden (z.B. „Toronto Books Bigrams“ für Sent2Vec (Pagliardini, et al., 2017)). Des Weiteren könnte man Deep Learning-Modelle, welche hier nicht berücksichtigt wurden, zur Evaluation von semantischer Ähnlichkeit heranziehen. Rekurrente neuronale Netze eignen sich gut für das Lernen von Dokumenten, da sie Sequenzen von Daten ohne vorher festgelegt Eingabegröße verarbeiten können. Diese im Rahmen der Arbeit nicht genutzte Technologie verspricht nicht nur im Rahmen des Natural Language Processings Größeres, sondern gilt als eines der beliebtesten Mittel des Machine Learnings.

8.2.3 Systematische Veränderung der Modellparameter

Man könnte die Parameter der distributionellen Modelle umfassender und feingranularer verändern und anschließend die trainierten Modelle überprüfen, um festzustellen, mit welcher Parameterkombination (z.B. die Dimension der Vektoren) das beste Metrikergebnis für Doc2VecC und Sent2Vec erzielt werden kann. Inwiefern können kleine Parameter eine große Auswirkung haben? Dies wäre eine der Kernfragen in einem solchen Versuchsaufbau, der sich grundsätzlich sehr stark an den Arbeiten dieser Thesis anlehnen würde.

8.2.4 Verwendung im *XING*-Kontext

Ursprünglich war diese Arbeit darauf ausgelegt, das in Kapitel 1 beschriebene Problem zu lösen, mit dem es beispielsweise das Karrierenetzwerk *XING* zu tun hat: Mehrere, täglich eingehende Artikel thematisch zu gruppieren.

[20] Nach (Sematch, 2017) soll es trotzdem theoretisch möglich, wpath für die Bestimmung von Dokumentenähnlichkeit zu verwenden (siehe Abschnitt 3.2.3).

Um diese Aufgabe zu lösen, wäre es notwendig, den Service in einer automatisierten Umgebung einzusetzen. Daraus würde man Werte generieren, anhand derer die Artikel zusammengefasst werden könnten, sofern die entsprechenden Mechanismen auf Frontend-Seite existierten. Erschwerend wäre hier allerdings, dass beim aktuellen Stand jeder Text mit jedem verglichen werden müsste, was aus Speicher- und Leistungssicht nicht vertretbar wäre.

Daher wäre es vom Standpunkt der Effizienz sinnvoll, vorher die Texte grob einem Thema zuzuordnen, bevor ihre Gleichheit überhaupt evaluiert wird, sodass nur diejenigen Texte verglichen werden, bei denen eine semantische Ähnlichkeit in Frage kommt.

Grundsätzlich sollte die Anwendung jedoch vor einem Deployment weiter verbessert werden, um so die Zuverlässigkeit und Stabilität ihrer Aussagen zu erhöhen.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

Alvarez, Jon Ezeiza. 2017. A review of word embedding and document similarity algorithms applied to academic text. *Universität Freiburg.* [Online] 2017. [Zitat vom: 24. 09 2018.] http://ad-publications.informatik.uni-freiburg.de/theses/Bachelor_Jon_Ezeiza_2017.pdf.

Banerjee, Prithu. 2016. UBC Wiki. *Course:CPSC522/Latent Dirichlet Allocation.* [Online] 16. März 2016. [Zitat vom: 20. November 2018.] https://wiki.ubc.ca/Course:CPSC522/Latent_Dirichlet_Allocation.

Baroni, Marco, et al. 2014. A SICK cure for the evaluation of compositional distributional semantic models. *9th Edition of its Language Resources and Evaluation Conference.* 2014.

Blei, David und Lafferty, John. 2009. Topic Models. [Buchverf.] Ashok Srivastava und Mehran Sahami. *Text mining: classification, clustering, and applications.* s.l. : Crc Pr Inc, 2009.

Blei, David, Ng, Andrew und Jordan, Michael. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research.* 2003, Bd. 3.

Cambridge University Press. 2008. *Stemming and lemmatization.* [Online] 2008. [Zitat vom: 20. 01 2019.] https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html.

Carstensen, Kai-Uwe, et al. 2004. *Computerlinguistik und Sprachtechnologie. Eine Einführung.* Heidelberg : Spektrum-Verlag, 2004. 978-3827414076.

Cer, Daniel, et al. 2017. SemEval-2017 Task 1: Semantic Textual Similarity - Multilingual and Cross-lingual Focused Evaluation. *CoRR.* 2017.

Chen, Minmin. 2017. Efficient Vector Representation for Documents through Corruption. *CoRR.* 2017.

code.google.com. 2013. https://code.google.com/archive/p/word2vec/. [Online] 30. Juli 2013. [Zitat vom: 15. 10 2018.] https://code.google.com/archive/p/word2vec/.

Dice, Lee. 1945. Measures of the Amount of Ecologic Association Between Species. *Ecology.* 1945, Bd. 26, 3.

Elsafty, Ahmed. 2017. Document Similarity using Dense Vector. *Universität Hamburg.* [Online] 2017. [Zitat vom: 26. 09 2018.] https://www.inf.uni-hamburg.de/en/inst/ab/lt/teaching/theses/completed-theses/2017-ma-elsafty.pdf.

Fano, Robert. 1961. *Transmission of Information: A Statistical Theory of Communications.* Cambridge, MA. : MIT Press, 1961. 978-0262561693.

Finkelstein, Lev, et al. 2002. Placing Search in Context: The Concept Revisited. *ACM Transactions on Information Systems.* 2002, Bd. 20.

Gerz, Daniela, et al. 2016. SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. *CoRR.* 2016.

Gruber, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge acquisition.* 1993, Bd. 5.

Harispe, Sébastien, et al. 2017. *Semantic Similarity from Natural Language and Ontology Analysis.* s.l. : Morgan & Claypool publishers, 2017. 10.2200/S00639ED1V01Y201504HLT027.

Harris, Zellig. 1954. Distributional Structure. *Word.* 1954, Bd. 10, 2-3.

Hellinger, Ernst. 1909. Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *Journal für die reine und angewandte Mathematik.* 1909, Bd. 136.

Hill, Felix, Reichart, Roi und Korhonen, Anna. 2014. SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. . *Computational Linguistics.* 2014.

Huang, Eric H., et al. 2012. Improving word representations via global context and multiple word prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers.* 2012, Bd. 1.

jisho. 2019. *jisho search.* [Online] 2019. [Zitat vom: 10. 01 2019.] https://jisho.org/search/hashi.

Jurafsky, Daniel und Martin, James H. 2008. *Speech and Language Processing. An Introduction to Natural LanguageProcessing, Computational Linguistics, and Speech Recognition.* 2. s.l. : Prentice-Hall, 2008. 978-0135041963.

Kiros, Ryan, et al. 2015. Skip-Thought Vectors. *CoRR.* 2015.

Kohila und Arunesh. 2016. TEXT MINING: TEXT SIMILARITY MEASURE FOR NEWS ARTICLES BASED ON STRING BASED APPROACH. *Global Journal of Engineering Science and Research Management.* 2016, 3.

Lau, Jey Han und Baldwin, Timothy. 2016. An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *Proceedings of the 1st Workshop on Representation Learning for NLP.* 2016.

Le, Quoc und Mikolov, Tomas. 2014. Distributed Representations of Sentences and Documents. *CoRR.* 2014.

Levenshtein, Vladimir. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Doklady Akademii Nauk SSSR.* 1965, Bd. 163, 4.

Levy, Omer und Goldberg, Yoav. 2014. Dependency-Based Word Embeddings. *ACL.* 2014.

Lieber spät als hart? **ZEIT. 2019.** 2019.

Lu, Jiaheng, et al. 2013. String Similarity Measures and Joins with Synonyms. *SIGMOD '13 Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data.* 2013.

Luhn, Hans Peter. 1957. A Statistical Approach to Mechanized Encoding and Searching of Literary Information. *IBM Journal of Research and Development.* 1957, Bd. 1.

Manning, Chris, Raghavan, Prabhakar und Schütze, Hinrich. 2008. *Introduction to Information Retrieval.* s.l. : Cambridge University Press, 2008. 978-0521865715.

Mathur, Natasha. 2018. Packtpub. *Top languages for Artificial Intelligence development.* [Online] 5. Juni 2018. [Zitat vom: 1. 12 2018.]

May scheitert krachend mit ihrem Brexit-Deal. **SPIEGEL. 2019.** 2019.

Mikolov, Tomas, et al. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR.* 2013.

Mitkov, Ruslan. 2005. *The Oxford handbook of computational linguistics.* s.l. : OxfordUniversity Press, 2005. 978-0199276349.

Mohammad, Marianna Apidianaki | Saif M., et al. 2018 . *Proceedings of The 12th International Workshop on Semantic Evaluation.* New Orleans, Louisiana : Association for Computational Linguistics , 2018 . 978-1-948087-20-9.

Pagliardini, Matteo, Gupta, Prakhar und Jaggi, Martin. 2017. Unsupervised Learning of Sentence Embeddings using Compositional n-Gram Features. *CoRR.* 2017.

Paperno, D., et al. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. *Proceedings of ACL 2016 (54th Annual Meeting of the Association for Computational Linguistics).* 2016.

Pearson, Karl. 1895. Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London.* 1895, Bd. 58.

Pennington, Jeffrey, Socher, Richard und Manning, Christopher D. 2014. GloVe: Global Vectors for Word Representation. *ACL.* 2014.

Resnik, Philip. 1995. Using information content to evaluate semantic similarity in a taxonomy. *Proceedings.* 1995, Bd. 1.

Rüschendorf, Ludger. 2014. *Mathematische Statistik.* Berlin, Heidelberg : Springer Verlag, 2014. 978-3-642-41996-6.

Sematch. 2017. Sematch Documentation. *An integrated framework for the development, evaluation, and application of semantic similarity for Knowledge Graphs.* [Online] 2017. [Zitat vom: 30. 11 2018.] http://gsi-upm.github.io/sematch/.

Shannon, C. 1948. A mathematical theory of communication. *Bell System Technical Journal.* 1948, Bd. 27.

Spärck Jones, Karen. 1972. A Statistical Interpretation of Term Specificity and Its Application in Retrieval. *Journal of Documentation.* 1972, Bd. 28.

Spearman, Charles. 1904. The proof and measurement of association between two things. *American Journal of Psychology.* 1904, Bd. 15.

Strang, Gilbert. 2016. *Introduction to Linear Algebra.* s.l. : Wellesley-Cambridge Press, 2016. 978-0980232776.

Taieb, Mohamed Ali Hadj, Aouicha, Mohamed Ben und Hamadou, Abdelmajid Ben. 2013. Computing semantic relatedness using Wikipedia features. *Knowledge-Based Systems.* 2013, Bd. 50.

Zhu, Ganggao und Iglesias, Carlos. 2017. Computing Semantic Similarity of Concepts in Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering.* 2017, Bd. 29, 1.

Zhu, Yukun, et al. 2015. Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. *CoRR.* 2015.

Zola, Andrew. 2018. The 5 Best Programming Languages for AI. *Springboard Blog.* [Online] 7. November 2018. [Zitat vom: 01. 12 2019.] https://www.springboard.com/blog/best-programming-language-for-ai/.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____