Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Dina Tarabih

Textähnlichkeit unter Verwendung von Ansätzen des maschinellen Lernens

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Dina Tarabih

Textähnlichkeit unter Verwendung von Ansätzen des maschinellen Lernens

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

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

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Eingereicht am: 20. Mai 2021

Dina Tarabih

Thema der Arbeit

Textähnlichkeit unter Verwendung von Ansätzen des maschinellen Lernens

Stichworte

Text-Mining, Maschinelles Lernen, Computerlinguistik, Big-Data, Textähnlichkeit, Word2Vec, Doc2Vec, BERT, Kosinusähnlichkeit, Word Mover's Distance, New York Times

Kurzzusammenfassung

Das Internet hat zu einer Zunahme elektronischer Online-Dokumente geführt. Dies hat Internetnutzern aufgrund der Informationsexplosion erschwert, relevante und genaue Informationen zu finden. Heutzutage wurden Techniken und Methoden des Text-Minings, maschinellen Lernens und der Computerlinguistik eingesetzt, um Big-Data zu verarbeiten. Diese Arbeit beschäftigt sich insbesondere mit dem Problem der Textähnlichkeit, eine wachsende Anwendung der Verarbeitung natürlicher Sprache, zur Optimierung der Suche von Dokumenten. Dabei werden vier Kombinationen verschiedener Feature-Extraktionstechniken TF-IDF, Word2Vec, Doc2Vec und BERT untersucht. Zusätzlich werden zwei Ähnlichkeitsalgorithmen Kosinusähnlichkeit und Word Mover's Distance zur Berechnung der Ähnlichkeitswerte eingesetzt. Diese Methoden werden anhand eines Datensatzes getestet, der 16.876 Artikel aus der New York Times-Tageszeitung enthält. Infolgedessen werden Hypothesen auf Basis von initialen Ergebnissen erstellt. Diese werden durch weitere Experimente zur Bewertung der Plausibilität und Zuverlässigkeit der erzielten Ergebnisse getestet.

Dina Tarabih

Title of the paper

Text Similarity Using Machine Learning Approaches

Keywords

Text Mining, Machine Learning, Natural Language Processing, Big Data, Text Similarity, Word2Vec, Doc2Vec, BERT, Cosine Similarity, Word Mover's Distance, New York Times

Abstract

The Internet has led to an increase in electronic documents online. This has made it difficult for internet users to find relevant and accurate information due to the information explosion. Today, text mining, machine learning, and natural language processing techniques and methods have been used to process big data. This thesis deals in particular with the problem of text similarity, a growing application of natural language processing, for the optimisation of document search. Four combinations of different feature extraction techniques, TF-IDF, Word2Vec, Doc2Vec, and BERT, are examined. In addition, two similarity algorithms, Cosine Similarity and Word Mover's Distance, are used to calculate the similarity values. These methods are tested on a data set containing 16,876 articles from the New York Times daily newspaper. As a result, hypotheses are created based on initial results. These are tested by further experiments to assess the plausibility and reliability of the results obtained.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In den letzten Jahren haben wir eine große dynamische Veränderung der Online-Informationen sowie ein schnelles Wachstum des Volumens der im World-Wide-Web verfügbaren Online-Textdokumente erlebt. Daher wurde in [40] vorhergesagt, dass diese Dokumente mit anderen unstrukturierten Daten der vorherrschende Datentyp sein werden, der online zu verschiedenen Verwendungszwecken gespeichert wird. Solche Webdokumente enthalten umfangreiche Textinformationen. Allerdings hat das schnelle Wachstum des Internets zu einer Datenüberlastung geführt und es den Benutzern zunehmend erschwert, die für sie relevanten Informationen zu finden. Dies bot und bietet immer noch eine große Chance, Dokumentengalerien effektiver zu nutzen. Daher besteht ein wachsender Forschungsbedarf, um effiziente Techniken zu entwickeln, mit denen Benutzer weltweit nützliche Informationen finden können.

Textähnlichkeit, die als die Gemeinsamkeit zwischen zwei Textausschnitten definiert ist, wird heutzutage zu einem Schlüsselinstrument für viele NLP-basierte Aufgaben (Natural Language Processing Tasks) [14], wie das Abrufen von Informationen (Information Retrieval, IR) [25], die automatische Beantwortung von Fragen und die maschinelle Übersetzung [77]. Infolgedessen konnte dieser Wissenschaftsbereich dazu beitragen, Informationen in der riesigen Galerie von Webdokumenten zu organisieren und daher auch effektiver zu suchen.

Zumal dieses Thema in den letzten Jahren eine breite Aufmerksamkeit erhalten hat, beteiligen sich Datenwissenschaftler heutzutage aktiv an Recherchen in diesem Bereich. Ihre Hauptaufgabe umfasst die Entwicklung von verschiedenen Metriken, die Ähnlichkeitsberechnungen zwischen Webdokumenten basierend auf einer großen Reihe von Ansätzen anstellen. Dabei argumentieren sie weiterhin ständig für das Werkzeug, das die absolut besten Werte liefern kann. Solche Forschungsprozesse beinhalten jedoch zahlreiche Schritte, die zuvor geprüft bzw. durchgeführt werden sollen, um geeignete Ergebnisse zu erzielen. Beginnend mit einem ausreichend großen Datensatz soll auch eine angemessene Datenqualität gesichert werden, um sinnvolle und zweckmäßige Messergebnisse zu erfassen. Zudem müssen diese Daten hinsichtlich ihrer Eignung, auf das untersuchte Problem spezifisch zugeschnitten zu sein, analysiert und bewertet werden.

Somit lässt sich Textähnlichkeit mit hohem (Zeit-) Aufwand durch Algorithmen berechnen, vor allem, weil die natürliche Sprache durch große Vielfalt gekennzeichnet ist. Dies wird zusätzlich dadurch erschwert, dass es mehrere Dimensionen gibt, nach denen die Ähnlichkeit von Texten beurteilt werden kann, etwa Inhalt, Struktur und Stil [8].

1.1 Ziel der Arbeit

Diese Arbeit konzentriert sich auf die Erforschung verschiedener Techniken des maschinellen Lernens, um ähnliche Dokumente zu identifizieren. Dies umfasst das Kombinieren vorgewählter Feature-Extraktionsmethoden mit Ähnlichkeitsmetriken und deren Anwendung auf variante Korpusse von Daten. Das Hauptziel besteht darin, die Kombinationen solcher Techniken zu vergleichen und ihre Leistung bezüglich des vorgeschlagenen Datensatzes zu untersuchen. Dabei soll folgende Frage insbesondere beantwortet werden können: Welche Methodenkombinationen sollen angewendet werden, um die genauesten Ähnlichkeitswerte zu erzielen.

1.2 Struktur der Arbeit

Im Folgenden wird die Struktur der Arbeit kurz skizziert. In Kapitel 2 wird auf die relevanten Grundlagen eigegangen, die für ein gutes Verständnis der Oberbegriffe vom maschinellen Lernen und der Computerlinguistik sorgen. Zudem wird in diesem Kapitel ein Überblick über die verschiedenen Textähnlichkeitsansätze auf String-, Korpus- und wissensbasierter Ebene gegeben, gefolgt von einer Vorstellung der vorangegangenen, relevanten Arbeiten und Untersuchungen zur Textähnlichkeit. In Kapitel 3 wird das Konzept dieser Arbeit erläutert, wo die zu bewältigende Problemstellung sowie die geplanten funktionalen und nicht-funktionalen Anforderungen vorgestellt werden. Das vierte Kapitel bietet einen tieferen explorativen Einblick in den Datensatz, auf dem diese Arbeit basiert, und geht auf die Hintergründe der Datenstruktur ein. Kapitel 5 enthält eine Einführung und Hintergrundinformationen zu den verschiedenen Feature-Extraktionsmethoden und Ähnlichkeitsmetriken, die im Rahmen dieser Arbeit verwendet werden. Der Versuchsaufbau und die Implementierungsdetails - inklusive aller verwendeten Techniken - folgen in Kapitel 6. In Kapitel 7 wird eine Bewertung der untersuchten Methodenleistungen anhand mehrerer ausgewählter Kriterien vorgenommen. Dabei werden die verschiedenen Ergebnisse der eingesetzten Methoden präsentiert und weiter diskutiert. Eine Zusammenfassung mit einem Ausblick zu diesem Thema schließen die Arbeit in Kapitel 8 ab.

2 Grundlagen

Im Laufe der Arbeit werden einige grundlegende Begriffe verwendet, die zunächst in diesem Kapitel zur weiteren Verdeutlichung vorgestellt werden. Zu Beginn werden die Konzepte des maschinellen Lernens und der Verarbeitung natürlicher Sprache erläutert und aufgezeigt, gefolgt von einer Beschreibung der verschiedenen Arten der Textähnlichkeitsansätze. Zuletzt wird ein Überblick über den derzeitigen Forschungsstand in diesem Zusammenhang diskutiert.

2.1 Maschinelle Lernkonzepte

Seit jeher haben Menschen Werkzeuge und Maschinen gebaut, um ihre Arbeit zu vereinfachen und den Gesamtaufwand für viele verschiedene Aufgaben zu reduzieren. Auch ohne Kenntnis physikalischer Gesetze erfanden sie solche Instrumente, um längere und komplexere Verfahren durchzuführen [6]. Dadurch wurden viele Aufgaben in der Entscheidungsfindung, Mustererkennung und anderen Bereichen, die früher menschliche Intelligenz erforderten, durch maschinelles Lernen automatisiert.

Maschinelles Lernen (ML) fungiert als eine Art Kategoriebezeichnung für alle Typen von Wissenserzeugung aus Daten, ohne dafür explizit programmiert worden zu sein. Vereinfacht dargestellt, lernt ein System aus vorhandenen Daten Zusammenhänge und Gesetzmäßigkeiten und kann diese dann auf neue, vorher nicht gesehene Daten anwenden [78]. Es ist ein Teilaspekt der künstlichen Intelligenz (KI) und beinhaltet technische (künstliche) Konzepte, aus bereits vorhandenem Wissen (Erfahrung) neues Wissen zu generieren. Ausgehend von Daten bzw. Beispielen wird dazu ein künstliches System (ML-Modell) trainiert, um Muster oder Gesetzmäßigkeiten zu erkennen, diese dann für eine Analyse neuer, bisher unbekannter Daten zu verwenden und daraus die richtigen Schlussfolgerungen zu ziehen [23].

In den nächsten Abschnitten werden drei Kategorien kurz beschrieben, in die ML unterteilt werden kann.

2.1.1 Überwachtes Lernen – Supervised Learning

In [42] ist ein überwachtes Szenario durch das Konzept eines Lehrers oder Aufsichtigers gekennzeichnet, dessen Hauptaufgabe darin besteht, dem Agenten ein genaues Maß für seinen Fehler zu liefern (direkt vergleichbar mit den Ausgabewerten). Bei tatsächlichen Algorithmen wird diese Funktion durch einen Trainingssatz bereitgestellt, der aus Paaren besteht (Eingabe und erwartete Ausgabe). Ausgehend davon kann der Agent seine Parameter so korrigieren, um den Wert einer globalen Verlustfunktion zu verringern. Wenn der Algorithmus nach jeder Iteration flexibel genug ist und die Datenelemente kohärent sind, erhöht sich die Gesamtgenauigkeit und die Differenz zwischen dem vorhergesagten und dem erwarteten Wert wird nahe Null.

In der Praxis ist eine der größten Herausforderungen im Bereich des überwachten Lernens, ausreichend und qualitativ hochwertig annotierte Datensätze zur Verfügung zu haben. Außerdem stellen wenig Daten eine besondere Schwierigkeit für den Einsatz komplexer Funktionen dar. Zu den verbreitetsten Algorithmen im Supervised Learning gehören Decision Trees, Support Vector Machines und K-Nearest Neighbour [78].

2.1.2 Unüberwachtes Lernen – Unsupervised Learning

Hier liegt der entscheidende Unterschied zum überwachten Lernen darin, dass unüberwachtes Lernen nicht auf Labels in den Datensätzen angewiesen ist und keinen Input von außen benötigt, um Daten zu klassifizieren. Mittels Hauptkomponenten (Principal Component Analysis, PCA) bzw. Cluster-Analysen ermöglicht unüberwachtes Lernen das Segmentieren, Einteilen und Bündeln von Daten anhand von algorithmisch detektierten Gemeinsamkeiten innerhalb der Daten [78].

2.1.3 Verstärkendes Lernen – Reinforcement Learning

Verstärkendes Lernen bedeutet, zu lernen, wie Situationen auf Aktionen abgebildet werden, um ein numerisches Belohnungssignal zu maximieren. Dem Lernenden wird nicht gesagt, welche Maßnahmen er ergreifen soll, sondern er muss herausfinden, welche Maßnahmen die größte Belohnung bringen, indem er sie trainiert [2]. Mit anderen Worten wird dem System ein Ziel vorgegeben, dessen Erreichen zu einer Belohnung führt. Der Weg, um das Ziel zu erreichen, wird aber nicht vorgegeben.

In [59, 78] wird diese Art von ML als eine Kombination beschrieben, die es ermöglicht, so viele Probleme, die für die Robotik relevant sind, in dem Begriff des verstärkenden Lernens zu formulieren. Genauer kann ein Roboter durch das verstärkende Lernen autonom ein optimales

Verhalten durch Trial-and-Error-Interaktionen mit seiner Umgebung ermitteln. Aber anstatt die Lösung eines Problems explizit zu beschreiben, gibt der Konstrukteur einer Steuerungsaufgabe beim verstärkenden Lernen Feedback in Bezug auf eine skalare Zielfunktion, die die einstufige Leistung des Roboters misst.

Verstärkendes Lernen ist sequentiell, was den großen Unterschied zum überwachten Lernen stellt, d.h. die aktuelle Belohnung hängt vom aktuellen Status ab und die nächste Belohnung hängt vom vorherigen Status und der folgenden Aktion ab [78]. Zudem finden sich Einsatzzwecke in einem breiten Anwendungsspektrum, insbesondere für die Steuerung und Optimierung komplexer Systeme, z. B. bei komplexen Spielen und der Bewältigung von einfachen Roboteraufgaben.

2.2 Natural-Language-Processing

Natural-Language-Processing (NLP, auf Deutsch auch als Computerlinguistik bezeichnet), ist eine der Schlüsselkomponenten der KI. Sie verwendet Computertechniken zum Lernen, Verstehen und Produzieren von Inhalten menschlicher Sprache [47].

Viele Informationen werden in unstrukturiertem Format generiert, sei es Bewertungen, Kommentare, Beiträge, Artikel usw., wobei eine große Datenmenge in natürlicher Sprache vorliegt. Mit NLP können Maschinen Muster aus solchen Textdaten verstehen und extrahieren, indem verschiedene Techniken wie Textähnlichkeit, Dokumentklassifizierung, Entitätsextraktion und Clustering angewendet werden.

2.2.1 Funktionsweise

Mithilfe der Text-Vektorisierung wandeln NLP-Tools Texte in ein Format um, das eine Maschine verstehen kann. Anschließend werden Algorithmen für maschinelles Lernen mit Trainingsdaten und erwarteten Ausgaben (Tags) versorgt, um Maschinen zu trainieren, die dadurch Assoziationen zwischen einer bestimmten Eingabe und ihrer entsprechenden Ausgabe herstellen können. Diese Maschinen verwenden dann statistische Analysemethoden, um ihre eigene Wissensbank (Knowledge Bank) aufzubauen und zu erkennen, welche Funktionen die Texte am besten darstellen, bevor sie Vorhersagen für unsichtbare Daten bzw. neue Texte treffen. Je mehr diese NLP-Algorithmen mit Daten versorgt werden, desto genauer werden natürlich die Textanalysemodelle sein. Dies ist in Abblidung 2.1 dargestellt.

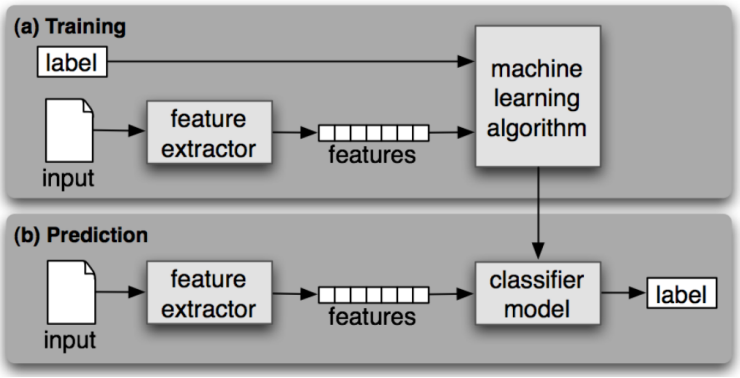


Abbildung 2.1: Textanalyse-Modell [4].

2.2.2 Analyseebenen

Traditionell wird bei der Verarbeitung natürlicher Sprache der Prozess der Sprachanalyse in mehrere Stufen zerlegt, was die theoretischen sprachlichen Unterschiede zwischen Syntax, Semantik und Pragmatik widerspiegelt. Diese dreigliedrige Unterscheidung in Syntax, Semantik und Pragmatik dient jedoch bestenfalls als Ausgangspunkt, wenn wir die Verarbeitung realer Texte in natürlicher Sprache betrachten wollen. Sie wird daher in den nächsten Abschnitten genauer skizziert.

Text in natürlicher Sprache besteht im Allgemeinen nicht aus den kurzen, ordentlichen, wohlgeformten und abgegrenzten Sätzen, die wir in Lehrbüchern finden. Daher wird die Phase der Tokenisierung und Satzsegmenierung normalerweise als einen entscheidenden ersten Schritt identifiziert, gefolgt von der lexikalischen Analyse, wobei sowohl Menschen als auch NLP-Systeme die Bedeutung einzelner Wörter interpretieren. Bei dieser Verarbeitung wird Wörtern, die als mehr als eine Wortart (Part-of-Speech) fungieren können, basierend auf dem Kontext, in dem sie vorkommen, das wahrscheinlichste Part-of-Speech-Tag zugewiesen [42].

Syntaktisches Parsen

Diese Phase erfordert sowohl eine Grammatik als auch einen Parser für die Analyse der Wörter in einem Satz, um die grammatikalische Struktur des Satzes aufzudecken. Die Ausgabe dieser Verarbeitungsebene ist eine möglicherweise abgegrenzte Darstellung des Satzes, die die strukturellen Abhängigkeitsbeziehungen zwischen den Wörtern aufdeckt [42].

Semantische Analyse

Die Identifizierung der syntaktischen Struktur einer Wortfolge ist nur ein Schritt zur Bestimmung der Bedeutung eines Satzes. Dies bietet ein strukturiertes Objekt, das für weitere Manipulationen und anschließende Interpretationen besser geeignet ist [27]. Die möglichen Bedeutungen eines Satzes bestimmt jedoch die semantische Verarbeitung, indem sie sich auf die Wechselwirkungen zwischen Bedeutungen auf Wortebene im Satz konzentriert. Hier kann die semantische Disambiguierung von Wörtern mit mehreren Sinnen umgefasst werden.

Pragmatische Analyse

Diese Ebene befasst sich mit dem gezielten Gebrauch von Sprache in Situationen und nutzt den Kontext über den Inhalt des Textes hinaus, um zu verstehen. Ziel ist es zu erklären, wie zusätzliche Bedeutung in Texte eingelesen wird, ohne tatsächlich in ihnen codiert zu werden. Dies erfordert viel Weltwissen, einschließlich des Verständnisses von Absichten, Plänen und Zielen [42].

2.2.3 Herausforderungen

Es gibt viele Herausforderungen bei der Verarbeitung natürlicher Sprache, aber einer der Hauptgründe, warum NLP schwierig ist, ist die Mehrdeutigkeit der menschlichen Sprache. Sogar Menschen haben Schwierigkeiten, die menschliche Sprache richtig zu analysieren und zu klassifizieren. Nehmen wir zum Beispiel Sarkasmus. Wie bringt man einer Maschine bei, einen Ausdruck zu verstehen, der das Gegenteil von dem sagt, was die richtige bzw. wahre Bedeutung ist? Aus diesem Grund müssen sowohl die Wörter als auch die Zusammenhänge der Konzepte verstanden werden, um die beabsichtigte Botschaft zu übermitteln. Während Menschen eine Sprache leicht beherrschen können, erschweren die Mehrdeutigkeit und die ungenauen Eigenschaften der natürlichen Sprachen die Implementierung von NLP für Maschinen.

2.3 Arten der Textähnlichkeitsansätze

Ähnlichkeit ist ein komplexes Konzept, das in der sprachlichen, philosophischen und informationstheoretischen Gemeinschaft vielfach diskutiert wurde [24]. Aus menschenzentrierter Sicht sagen wir, dass Textähnlichkeit eine Funktion zwischen zwei Texten ist, die informell durch die Sicht der Leser auf die Texteigenschaften charakterisiert werden kann, anhand derer Ähnlichkeit beurteilt werden kann. Andererseits und insbesondere aus einer maschinenzentrierten Perspektive wird diese Art der Textanalyse an einem definierten und strukturierten

Modell des Textes in natürlicher Sprache durchgeführt, das nach Durchführung bestimmter Textverarbeitungsschritte und Transformationen erhalten wird [64]. Mit anderen Worten umfasst diese Aufgabe des Text-Minings [17] mehrere komplexere Vorverarbeitungsschritte. Diese ermöglichen, ein strukturiertes repräsentatives Modell der Dokumente in einem Korpus durch Extraktion deren Merkmale (Features) zu erhalten. Normalerweise werden vektorbasierte Modelle verwendet. Dabei wird ein Textdokument durch Feature-Vektoren dargestellt, die den Inhalt dieses Dokumentes beschreiben. Das Modell einer Sammlung von Dokumenten ist dann der hochdimensionale Raum dieser Dokumentvektoren. Schließlich können verschiedene Ähnlichkeitsalgorithmen verwendet. Einige davon weisen den Dokumenten eine reelle Zahl zwischen 0 und 1 zu. Ein Nullwert bedeutet, dass die Dokumente völlig unterschiedlich sind, während ein Wert von eins angibt, dass die Dokumente praktisch identisch sind. Mehr darauf wird in Kapitel 5 eingegangen.

Die Berechnung der Textähnlichkeit hat angesichts der Informationsproblematik des schnellen Datenwachstums in den letzten Jahren als wesentliche Methode des Text-Minings und als grundlegendes Mittel für viele NLP-Aufgaben zunehmend an Bedeutung gewonnen. In [20, 74] wurden verschiedene Ansätze gefördert, um die lexikalische und semantische Ähnlichkeit zwischen Texten zu messen. Anbei werden die drei Haupttypen dieser Ansätze ausführlich erörtert: String-basiert, Korpus-basiert und Wissensbasiert.

2.3.1 String-basierte Ähnlichkeit – String-based Similarity

String-basierte Algorithmen werden verwendet, um die lexikalische Ähnlichkeit [61] basierend auf Zeichen- und Anweisungsübereinstimmungen zu messen. Solche Algorithmen arbeiten mit String-Sequenzen und Zeichenkompositionen. Sie berücksichtigen den Grad, in dem zwei Zeichenfolgen miteinander übereinstimmen.

String-basierte Ähnlichkeit kann weiter in zeichenbasierte (Character-based) und termbasierte Ähnlichkeit (Term-based) klassifiziert werden:

Zeichenbasierter Ansatz – Character-based Similarity

Hier wird der Bearbeitungsabstand, einschließlich Einfügen, Löschen, Ersetzen eines einzelnen Zeichens oder Transposition zweier benachbarter Zeichen, zwischen zwei Zeichenfolgen definiert. Damit wird die minimale Anzahl von Operationen gezählt, die erforderlich sind, um eine Zeichenfolge in die andere umzuwandeln.

- **N-Gramm**

N-Gramm ist eine Teilsequenz von n Elementen aus einer bestimmten Textsequenz. Dessen Ähnlichkeitsalgorithmen vergleichen die N-Gramme von jedem Zeichen oder Wort in zwei Zeichenfolgen. Die Entfernung wird dann berechnet, indem die Anzahl ähnlicher N-Gramme durch die maximale Anzahl von N-Grammen geteilt wird [32].

Andere Beispiele für diesen Ansatz sind die Levenshtein-Distanz [39] und LCS [28].

Termbasierter Ansatz – Term-based Similarity

Die termbasierte Ähnlichkeit, auch als tokenbasiert bezeichnet, ist etwas komplexer. Sie analysiert Text als eine Reihe von Token (Wörtern). Die Ähnlichkeit zwischen Zeichenfolgen kann dann durch Manipulation von Token-Sätzen wie Wörtern bewertet werden. Die Hauptidee hinter diesem Ansatz besteht darin, zwei Messungen der String-Ähnlichkeit basierend auf allgemeinen Token durchzuführen, die ihren Token-Sätzen entsprechen [81].

- **City-Block-Distanz**

Dieser Algorithmus wird auch als Manhattan-Distanz bezeichnet. Er berechnet die Entfernung, die zurückgelegt werden würde, um von einem bestimmten Datenpunkt zum anderen zu gelangen, wenn ein gitterartiger Pfad verfolgt wird. Der City-Block-Abstand zwischen zwei Elementen ist dann die Summe der Unterschiede deren entsprechenden Komponenten [16]. Das heißt, dass der Abstand d_1 zwischen den beiden Vektoren p, q in einem n-dimensionalen realen Vektorraum mit einem festen kartesischen Koordinatensystem die Summe der Längen der Liniensegmentprojektionen zwischen den Punkten auf die Koordinatenachsen ist [74].

$$d_1(p,q) = \|p-q\|_1 = \sum_{n=1}^{\infty} |p_i + q_i| \tag{2.1}$$

wobei (p,q) Vektoren sind.

Kosinusähnlichkeit [3] und euklidischer Abstand [33] sind weitere Beispiele für diese Art von Ähnlichkeitsalgorithmen, wobei die Kosinusähnlichkeit in Kapitel 5 separat erläutert wird.

2.3.2 Korpus-basierte Ähnlichkeit – Corpus-based Similarity

Dies ist eine semantische Ähnlichkeit, bei der Wörter aufgrund ihrer Bedeutung ähnlich sind. Korpusbasierte Ähnlichkeit [50] bestimmt die Ähnlichkeit zwischen Wörtern basierend auf Informationen, die aus einem großen Korpus extrahiert werden. Dieses große Korpus enthält weiterhin verschiedene Arten von Dokumenten aus verschiedenen Wissensbereichen [60].

- **Hyperspace Analogue to Language (HAL)**

HAL [46, 45] schafft einen semantischen Raum aus der Berechnung des gleichzeitigen Auftretens von Wörtern (Word Co-Occurrences). Eine wortweise Matrix wird gebildet, wobei jedes Matrixelement die Assoziationsstärke zwischen dem durch die Zeile dargestellten Wort und dem durch die Spalte dargestellten Wort abbildet. Der Benutzer des Algorithmus hat dann die Möglichkeit, Spalten mit niedriger Entropie aus der Matrix zu entfernen. Während der Textanalyse wird ein Fokuswort am Anfang eines Zehn-Wort-Fensters platziert, das aufzeichnet, welche benachbarten Wörter als gleichzeitig auftretend gezählt werden. Matrixwerte werden dann akkumuliert, indem das gleichzeitige Auftreten umgekehrt proportional zur Entfernung vom Fokuswort gewichtet wird. Dabei wird angenommen, dass näher benachbarte Wörter mehr von der Semantik des Fokusworts widerspiegeln und daher höher gewichtet werden.

Andere korpusbasierte Ähnlichkeitsmaße sind Latent Semantic Analysis (LSA) [35] und Explicit Semantic Analysis (ESA) [18]. Davon gilt LSA als die beliebteste Technik der korpusbasierten Ähnlichkeit. Sie wird im nächsten Abschnitt 2.4 näher beschrieben.

2.3.3 Wissensbasierte Ähnlichkeit – Knowledge-based Similarity

Wissensbasierte Ähnlichkeit [60] ist ein semantisches Ähnlichkeitsmaß. Es verwendet verschiedene Informationen aus semantischen Netzwerken, um den Ähnlichkeitsgrad zwischen Wörtern zu ermitteln [74].

WordNet [52] ist das beliebteste semantische Netzwerk. Es handelt sich um eine große lexikalische Datenbank mit englischen Wörtern, die als Substantive, Verben, Adjektive und Adverbien gekennzeichnet sind. Diese Wörter sind in Gruppen von Synonymen *Synsets* gruppiert. Sie drücken jeweils ein bestimmtes Konzept aus [20].

Wissensbasierte Ähnlichkeitsansätze, die die WordNet-Ontologie verwenden, können grob in zwei Gruppen unterteilt werden: Maße der semantischen Ähnlichkeit (Semantic Similarity) und Maße der semantischen Verwandtschaft *Semantic Relatedness*. Sie basieren entweder auf Informationsinhalten oder auf Pfadlängen und implementieren Beziehungen zwischen Wörtern und Konzepten, die in semantischen Repositorys verfügbar sind.

Die folgenden Wortähnlichkeitsmetriken Resnik und Lin sind Wort-zu-Wort-Metriken. Sie wählen ein bestimmtes Wortpaar mit der höchsten Konzept-zu-Konzept-Ähnlichkeit aus.

- **Resnik-Ähnlichkeit**

Das von Resnik [62] eingeführte Ähnlichkeitsmaß wird verwendet, um den Informationsgehalt *Information Content*, oder kurz IC, des Least Common Subsumer *LCS* von zwei Konzepten zurückzugeben. LCS ist ein zeichenbasiertes Ähnlichkeitsmaß. Dabei basiert die Ähnlichkeit zwischen den Zeichenfolgen auf der zusammenhängenden Kette von Zeichenlängen, die in beiden Zeichenfolgen vorhanden sind.

$$sim_{resnik}(c_1, c_2) = IC(LCS(c_1, c_2)) = -logP(LCS(c_1, c_2)) \tag{2.2}$$

- **Lin-Ähnlichkeit**

Die Metrik von Lin [43] basiert auf dem Ähnlichkeitsmaß Resniks, das einen Normalisierungsfaktor hinzufügt. Dieser Faktor besteht aus dem Informationsgehalt der beiden Eingabekonzepte.

$$sim_{lin}(c_1, c_2) = \frac{2logP(LCS(c_1, c_2))}{logP(c_1) + logP(c_2)} \tag{2.3}$$

2.4 Stand der Technik

Die Aufgabe der Textähnlichkeit wurde intensiv untersucht, sodass eine Vielzahl von Recherchen zu diesem Thema und somit verschiedene Ähnlichkeitsmaße in der Literatur vorgeschlagen wurden. Frühere Arbeiten auf diesem Gebiet können grob in vier Hauptkategorien eingeteilt werden: Vektor-basierte, korpusbasierte, Hybrid- und Feature-basierte Methoden. Dazu ist eine Übersicht über diese Arbeiten in Tabelle 2.1 zusammengefasst.

Eine der frühesten Anwendungen der Textähnlichkeit ist das Vektormodell, das heutzutage in vielen Information-Retrieval-Systemen verwendet wird [49], wobei das für eine Eingabeabfrage relevanteste Dokument durch Darstellung eines Dokuments als Wortvektor bestimmt wird. Danach werden Abfragen über eine Ähnlichkeitsmetrik mit ähnlichen Dokumenten in der Dokumentendatenbank abgeglichen [66]. Es gibt viele verschiedene Algorithmen, die als Vektor-basierte Modelle betrachtet werden können. Der Hauptanreiz für diese Modelle ist die Fähigkeit, komplexe Informationen in einer relativ vereinfachten Form darzustellen, die es ermöglicht, eine Vektorrechnung auf die Textanalyse anzuwenden. In [30] wurde das aktuelle Problem der semantischen Analyse auf Basis eines Vergleichs von Bildungskursen untersucht. Dies ist generell erforderlich, um Bildungsprogramme zu aktualisieren, wenn die Menge der im Internet verfügbaren Bildungsinhalte kontinuierlich wächst und sich die Anforderungen der Standards immer wieder ändern. Bei dieser Untersuchung wurden verschiedene Algorithmen der semantischen Analyse anhand ihrer Vektordarstellungen überprüft, wobei der

Schwerpunkt auf der experimentellen Qualitätsbewertung von Vektor-basierten Modellen für Textdarstellungen lag.

Später und zur Einschließung lexikalischer und semantischer Eigenschaften eines Textes, wurden Vektorraummodelle der Semantik (Vector Space Models of Semantic, kurz als VSMs bezeichnet) eingeführt, bei denen die lexikalische Bedeutung durch Berechnung der Word-Co-Occurrences und semantische Informationen durch Berechnung der Verteilungsdarstellung des Textes berücksichtigt wurden. Solche Textdarstellungsmethoden haben sich im Laufe der Zeit weiterentwickelt, um die Originalität der Darstellung zu verbessern. Infolgedessen wurde ein Experiment durchgeführt [19], um die Leistung von VSM- und VSMs-basierten Textdarstellungsmodelle in Klassifizierungs- und Abrufaufgaben (Retrieval Tasks) zu bewerten. Dieses Experiment basierte auf einem Datensatz, der von den Organisatoren der *Task Mixed Script Information Retrieval* (*MSIR*) bereitgestellt wurde [1].

Darüber hinaus gibt es korpusbasierte Methoden, die davon ausgehen, dass Wörter mit ähnlicher Bedeutung häufig in ähnlichen Kontexten vorkommen. Eine solche bekannte Methode übernimmt das Latent Semantic Analysis Modell (auch als LSA bekannt) [35]. LSA, ein hochdimensionales lineares Assoziationsmodell, analysiert einen großen Korpus von Texten in natürlicher Sprache und generiert eine Darstellung, die die Ähnlichkeit von Wörtern und Textpassagen erfasst. Eine Matrix, die die Anzahl der Wörter in jedem Dokument enthält (Zeilen, die eindeutige Wörter darstellen, und Spalten, die jedes Dokument darstellen), besteht aus einem großen Textstück. Die Singular-Value-Decomposition (SVD) wird dann verwendet, um die Anzahl der Zeilen zu verringern und gleichzeitig die Ähnlichkeitsstruktur zwischen den Spalten beizubehalten. Zuletzt wird das Dokument verglichen, indem den Kosinus-Wert des Winkels zwischen den beiden Vektoren genommen wird. Dabei stehen Werte nahe 1 für sehr ähnliche Dokumente und Werte nahe 0 für sehr unterschiedliche Dokumente.

Ein weiterer sehr bekannter korpusbasierter Ansatz ist das Bidirectional Encoder Representations from Transformers, oder kurz als BERT-Modell bezeichnet [13]. Im Gegensatz zu anderen Textrepräsentationsmodellen wurde BERT entwickelt, um tiefe bidirektionale Darstellungen aus unbeschriftetem Text durch gemeinsames Konditionieren des linken und rechten Kontexts auf allen Ebenen zu trainieren. Infolgedessen kann das vorab trainierte BERT-Modell durch genau eine zusätzliche Ausgabeebene optimiert werden, um hochmoderne Modelle für eine Vielzahl von Aufgaben zu erstellen, wie Sprachinferenz und Beantwortung von Fragen. Dieses Modell wird in Kapitel 5 ausführlicher erläutert.

Die folgende Arbeit gab einen Leistungsüberblick über verschiedene Arten von korpusbasierten Modellen im Bereich der Paraphrasenerkennung [75]. Dieser Bereich gilt als sehr wichtig für eine Reihe von Anwendungen, einschließlich Plagiatserkennung, Beantwortung von Fra-

gen, Zusammenfassung von Texten und Text-Mining im Allgemeinen. Dabei wurden acht verschiedene Modelle hinsichtlich Genauigkeit, Präzision, Rückruf und F1-Messungen an drei verschiedenen öffentlich verfügbaren Korpora bewertet: *Microsoft Research Paraphrase Korpus* (*MSRP*), *Clough und Stevenson* (*C&S*) und *Webis Crowd Paraphrase Korpus 2011* (*Webis-CPC-11*).

Weitere semantische Ähnlichkeitsmaße verwenden sowohl wissensbasierte [80, 37] als auch korpusbasierte [73] Ansätze. Solche Maße werden als Hybrid-Methoden bezeichnet. In [50] wurde eine kombinierte Methode zur Messung der semantischen Ähnlichkeit von Kurztexten vorgeschlagen, bei der zwei korpusbasierte und sechs wissensbasierte Ähnlichkeitsmaße verwendet wurden. Dabei wurde durch an einem Paraphrasendatensatz durchgeführte Experimente gezeigt, dass die semantische Ähnlichkeitsmethode Methoden übertrifft, die auf einfachem lexikalischem Matching basieren, was zu einer Reduzierung der Fehlerrate um bis zu 13% in Bezug auf das traditionelle vektorbasierte Ähnlichkeitsmaß geführt hat. Eine andere hybride Methode wurde in [41] vorgestellt. Sie hat sich direkt auf die Berechnung der Ähnlichkeit zwischen sehr kurzen Texten mit Satzlänge konzentriert. Dabei wurde ein Algorithmus vorgeschlagen, der semantische Informationen und in den Sätzen enthaltene Informationen zur Wortreihenfolge berücksichtigt hat. Die semantische Ähnlichkeit zweier Sätze wurde dann anhand von Informationen aus Korpusstatistiken sowie der strukturierten lexikalischen WordNet-Datenbank [52] berechnet.

Feature-basierte Methoden versuchen grundsätzlich, einen Satz mithilfe einer Reihe vordefinierter Features darzustellen. Die Ähnlichkeit zwischen zwei Texten wird durch einen trainierten Klassifikator erreicht. Das Finden effektiver Features und das Erhalten deren Werten aus Sätzen machen diese Kategorie von Methoden jedoch unpraktisch. In [68] wurde ein Feature-basiertes Ähnlichkeitsmaß vorgeschlagen, das versucht hat, Videoserien aus menschlicher Sicht zu interpretieren. Dieses Maß hat Objekte und ihre relativen räumlichen und zeitlichen Positionen berücksichtigt und gleichzeitig die Ähnlichkeit zwischen zwei Videoserien geschätzt. Ein anderes Experiment [7] hat Feature-basierte Methoden zum Abrufen von 3D-Daten untersucht und eine Taxonomie für diese Methoden vorgeschlagen. Dabei wurden auch verschiedene experimentelle Ergebnisse vorgestellt, in denen die Wirksamkeit einiger der untersuchten Methoden verglichen wurde.

Ansatz	Verfahren	Methodik	Methoden	Datensätze	Bewertung
[30]	Vektor-basiert	Verschiedene Ansätze zur semantischen Analyse der Bildungsprogramme werden anhand ihrer Vektordarstellungen bewertet.	TF-IDF, LSA, LDA, averaged Word2Vec und Paragraph2Vec.	Das Korpus besteht aus mehr als 100 verschiedenen Bildungsprogrammen von verschiedenen russischen Universitäten, und 7 gemeinsamen Wissensbereichen: Informationstechnologien, Wirtschaftsstudien, Mathematik & Statistik, Linguistik, Geschichte, Medizin und Recht.	Die Auswertung bestätigte, dass die mit Paragraph2Vec erzeugten Vektoren mit einer F-Messung von **0,90** gut genug für die semantische Analyse des Kursprogramms waren.
[75]	Korpus-basiert	Ein Leistungsüberblick wird über verschiedene Arten von korpusbasierten Modellen gegeben, insbesondere über Deep-Learning-Modelle mit der Aufgabe der Paraphrase-Erkennung.	LSI, TF-IDF, Word2Vec, Doc2Vec, GloVe, FastText, ELMO, und USE	– Microsoft Research Paraphrase-Korpus: besteht aus 5801 Satzpaaren, die über einen Zeitraum von 18 Monaten aus Tausenden von Nachrichtenquellen im Internet gesammelt wurden. – Clough and Stevenson-Korpus: bietet eine Sammlung von fünf Fragen und 95 kurzen Antworten auf diese Fragen. – Webis Crowd Paraphrase-Korpus 2011: enthält 7859 Kandidatenparaphrasen mit 4067 akzeptierten Paraphrasen, 3792 abgelehnten Nicht-Paraphrasen und den Originaltexten.	– (MSRP): F1-Messung von **0,813** mit TF-IDF. – (C&S): F1-Messung von **0.968** mit USE. – (Webnis): F1-Messung von **0.709** unter Verwendung der Modelle USE & ELMO.
[50]	Hybrid (Wissensbasiert & Korpus-basiert)	Eine Methode wird zur Messung der semantischen Ähnlichkeit von Texten unter Verwendung von korpusbasierten und wissensbasierten Ähnlichkeitsmaßen der wortsemantischen Ähnlichkeit vorgestellt. Die Methode soll beweisen, dass dieses Maß den einfacheren vektorbasierten Ähnlichkeitsansatz übertrifft auf einer Paraphrase-Erkennungsaufgabe.	– Korpusbasierte Methoden: (PMI-IR) von Turney (2001) und Latent Semantic Analysis von Landauer (1998). – Wissensbasierte Methoden: Jiang & Conrath (1997), Leacock & Chodorow (1998), Lesk (1986), Lin (1998), Wu & Palmer (1994) und Resnik (1995).	kurze Textausschnitte (z. B. Zusammenfassungen wissenschaftlicher Dokumente, Bildunterschriften und Produktbeschreibungen)	Es wurde eine endgültige Gesamtgenauigkeit (Final Accuracy) von **70,3%** und eine F-Messung von **81,3%** aufgezeichnet, was eine signifikante Verringerung der Fehlerrate um 13,8% in Bezug auf die vektorbasierte Kosinus-Ähnlichkeitsbasislinie darstellt.
[41]	Hybrid (Wissensbasiert & Korpus-basiert)	Diese Forschung konzentriert sich direkt auf die Berechnung der Ähnlichkeit zwischen sehr kurzen Texten mit Satzlänge. Dabei wird ein Algorithmus vorgestellt, der semantische Informationen und Wortordnungsinformationen berücksichtigt, die in den Sätzen enthalten sind.	Die vorgeschlagene Methode leitet Textähnlichkeit aus semantischen und syntaktischen Informationen ab. Im Gegensatz zu bestehenden Methoden, die einen festen Wortschatz verwenden, bildet diese Methode dynamisch einen gemeinsamen Wortsatz, bei dem nur alle unterschiedlichen Wörter im Satzpaar verwendet werden.	Ein vorbereitender Datensatz von Satzpaaren wurde mit menschlichen Ähnlichkeitswerten von 32 Teilnehmern erstellt. Diese Sätze bestehen aus Wörterbuchdefinitionen von Wörtern. Zudem wurde ein weiterer Datensatz nicht definitiver Sätze aus der NLP-Literatur erstellt.	Das Ähnlichkeitsmaß erreichte einen einigermaßen guten Pearson-Korrelationskoeffizienten von **0,816** mit den menschlichen Bewertungen, die bei 0,01 signifikant waren.
[7]	Feature-basiert	Feature-basierte Methoden werden zum inhaltsbasierten Abrufen von 3D-Objekten untersucht. Dabei wird eine Taxonomie für diese Methoden vorgeschlagen. Schließlich werden experimentelle Ergebnisse präsentiert, in denen die Wirksamkeit einiger der untersuchten Methoden verglichen wird.	16 verschiedene Arten von Feature-Vektoren (weiter als FVs bezeichnet) wurden implementiert, darunter: statistische FVs (3D Moments), geometriebasierte FVs (Principal Curvature, Shape Distribution, Ray-based, Ray-based with Spherical Harmonics, Shading, Complex-valued Shading, Cords-based, Segment Volume Occupation, Voxel-based, 3DDFT, Rotation Invariant Spherical Harmonics), bildbasierte FVs (Depth Buffer, Silhouette) und andere Ansätze (Rotation Invariant Point Cloud Descriptor).	Es wurde eine klassifizierte Datenbank von 3D-Objekten verwendet, die aus Modellen aus dem Internet besteht und somit 1.838 3D-Objekte enthält. Aus diesem Datensatz wurden 472 Objekte manuell nach Formähnlichkeit in 55 verschiedene Modellklassen klassifiziert. Der Rest der Objekte wurde als “nicht klassifiziert” belassen, wobei jedes klassifizierte Objekt jeder Modellklasse als Abfrage-Objekt verwendet wurde.	Depth Buffer verzeichnete eine durchschnittliche R-Genauigkeit von **0,3220** mit einer besten Dimensionalität von 366.

Tabelle 2.1: Eine Vergleichstabelle zu den in Abschnitt 2.4 erläuterten Experimenten.

3 Konzept

In diesem Kapitel werden die grundlegenden Problemstellungen dieser Arbeit sowie die geplanten funktionalen und nicht-funktionalen Analyseanforderungen zum Erzielen einer angemessenen Berechnung der Textähnlichkeit eingeführt.

3.1 Problembeschreibung

Wie bereits erwähnt, hat das Volumen der Online-Textdokumente in den letzten Jahren stark zugenommen. Vor allem unter Berücksichtigung der Tatsache, dass das World Wide Web generell zur Durchführung von Forschung und zur Suche nach relevanten Informationen verwendet wird [29], hat diese weit verbreitete Forschung auch zu einer weiteren Explosion von Forschungsarbeiten und wissenschaftlichen Ergebnissen in Form von Zeitschriften und wissenschaftlichen Literaturwerken geführt. Diese Explosion zwingt heutzutage zu einer weiteren Kategorisierung und Klassifizierung von Dokumenten in verschiedenen Online-Repositories, was zu einer ständigen Informationsüberflutung führt [70]. Um diesen Herausforderungen zu begegnen, und daher die Empfehlungssystemansätze und verschiedenen Techniken aus der Informationssuche und -abfrage weiterzuentwickeln, werden Text-Mining-Algorithmen, Dokument-Empfehlungstechniken und Methoden zum Information-Retrieval eingesetzt, um das Big-Data im Internet zu verarbeiten. Diese Techniken werden kombiniert und in Forschungspapierfunktionen verwendet, um das ähnlichste, relevanteste und wichtigste Dokument zu finden, das einem Forscher in seiner Suche weiterhelfen kann.

In den letzten Jahren sind insbesondere Aufgaben des Text-Minings wie die Berechnung der Textähnlichkeit von besonderem Interesse geworden. Viele Unternehmen wie Google würden gerne Millionen von Dollar für Ingenieurleistung und die neueste Rechenleistung ausgeben, um ihre Suchmaschinen um bloß 1% zu verbessern.

Allerdings bleibt die Berechnung der Textähnlichkeit nicht nur für uns Menschen, sondern auch für Maschinen eine sehr schwierige Aufgabe. Dies ist hauptsächlich auf die enorme Variabilität in der natürlichen Sprache zurückzuführen, in der Texte mit unterschiedlichen lexikalischen und syntaktischen Konstruktionen erstellt werden können. Somit, wenn man nach den besten Textähnlichkeitsalgorithmen suchen würde, wenn es darum geht, die genauesten

Ergebnisse zu erzielen, würde man sicherlich verschiedene Suchergebnisse aus wissenschaftlichen Arbeiten, Blogs und Q&As erhalten.

In einem Versuch, diese Suchanfrage zu beantworten, werden verschiedene Möglichkeiten verglichen, Textobjekte zu repräsentieren und diese mithilfe von Distanzmetriken auf Ähnlichkeit zu prüfen. Dabei wird diskutiet, welcher Algorithmus auszuwählen wäre, der in der Praxis für einen gegebenen Datensatz besser geeignet sein könnte.

3.2 Anforderungen

Bei den vorzunehmenden Textähnlichkeitsanalysen sollen ein paar Punkte hinsichtlich der gewünschten Funktionalitäten sowie der Analyseansprüche überprüft und gesichert werden. Die folgenden funktionalen und nicht-funktionalen Anforderungen sollen daher berücksichtigt und schließlich erfüllt werden:

3.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen spezifizieren, welche Funktionalitäten die im Rahmen dieser Arbeit zu implementierenden Ähnlichkeitsmethoden unter festgelegten Bedingungen besitzen bzw. erfüllen sollen.

1. Zuerst sollen solche Datensätze für die Ähnlichkeitsanalysen bzw. Experimente verwendet werden, die Artikel aus populären Medienquellen enthalten, um die Effektivität realer Anwendungen zu entdecken.

2. Der Inhalt der erfassten Artikel soll diverse Themen abdecken, die wiederum in mehrere Kategorien eingeteilt werden können. Dies soll später bei der Ausgaben- bzw. Leistungsevaluation der Ähnlichkeitsmethoden helfen.

3. Vor der Verarbeitung der Dateneingaben sollen diese gereinigt werden, indem die falschen oder fehlenden Werte beseitigt und eliminiert werden. Dadurch wird die Datenqualität gesichert, da bei inkonsistenten Daten oder Duplikaten die Genauigkeit der Ergebnisse wesentlich beeinträchtigt wird.

4. Drei neue bzw. ungesehene Basis-Suchabfragen sollen aus verschiedenen Kategorien ausgewählt werden. Hiermit soll ein angemessener Ähnlichkeitsvergleich zwischen denen und dem für diese Arbeit ausgewählten Datensatz geschlossen werden.

5. Es sollen vorab trainierte Repräsentationsmodelle eingebaut werden. Das soll später ermöglichen, eigene Textähnlichkeitsanalysen einfach einzurichten und ähnliche Ausgaben zu erwarten.

6. Zudem sollen bis zwei Ähnlichkeitsmetriken auf die Ergebnisse der Repräsentationsmodelle angewendet werden, um den Ähnlichkeitsgrad zwischen den Suchabfragen und den einzelnen Artikeln in dem Datensatz zu berechnen.

7. Nachdem alle Ausgabewerte via den Ähnlichkeitsmethoden gefunden worden sind, sollen diese anhand bestimmter Kriterien, etwa Schlüsselwort- und Kategorieüberschneidungen, bewertet werden.

8. Die Methodenleistungen in Bezug auf die Gesamtausführungszeit und die Gesamtspeicherauslastung soll ebenfalls mit bewertet werden.

9. Es sollen weitere Experimente mit künstlichen bzw. manipulierten Probedaten ausgeführt, um die Gültigkeit der gewonnenen Ausgabewerte stärker zu beweisen.

10. Basierend auf den obigen Evaluationen soll geschlossen werden, ob ein Algorithmus in praktischer Hinsicht genauere Ergebnisse erzielen kann als ein anderer.

3.2.2 Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen beschreiben die Charakteristiken, die ein System aufweisen muss, oder die Rahmenbedingungen, die eingehalten werden müssen. Mit anderen Worten bezeichnen sie, mit welchem Umfang die Ähnlichkeitsmethoden entwickelt werden sollen, und wie gut sie ihre Leistung erbringen sollen.

Funktionalität

Alle festgelegten funktionalen Anforderungen sollen erfüllt und umsetzbar sein. Außerdem sollen die Analysemodelle die geforderte Angemessenheit bzw. Plausibilität der Ergebnisse sicherstellen.

Zuverlässigkeit

Die Ähnlichkeitsmethoden sollen zuverlässig sein, da sonst deren Leistungsniveau sowie zugewiesenen Funktionen in einem bestimmten Zeitintervall nicht erfüllt werden können. Außerdem sollen diese Methoden fehlerfrei durchlaufen, sodass der Nutzer dieselben Ergebnisse bei jedem Durchlauf bekommt.

Effizienz

Die untersuchten Methodenkombinationen werden unterschiedliche Messergebnisse eventuell liefern und daher auch unterschiedliche Leistungsniveaus aufweisen. Aus diesem Grund soll ein Überblick über die effizientesten Methodenkombinationen dadurch gegeben werden, indem die Höhe der erzielten Werte zwischen den Artikeln mit dem Grad an Themen- und Schlüsselwortüberschneidungen dieser Artikel verglichen wird. Mit anderen Worten, ob die beiden Faktoren voneinander abhängig sind.

Ein weiterer Punkt der Effizienz ist der eventuell geringere Ressourcenbedarf durch die Verwendung vorab trainierter Repräsentationsmodelle. Dieser wird dadurch gemessen, indem der durchschnittliche Speicherbedarf jedes Experiments berechnet wird, und danach mit jedem anderen verglichen wird, um zu zeigen, wie ein Verfahren effizienter sein kann als ein anderes, und somit Ressourcen sowie Verarbeitungszeit sparen kann. Außerdem nicht zu vergessen, dass diese Modelle generell an solchen Datensätzen trainiert werden sollen, die branchenüblichen Standards entsprechen, sodass sie bereits auf den Qualitätsaspekt überprüft sein werden.

Änderbarkeit

Die Ähnlichkeitsmethoden sollen anhand der sich immer wechselnden Eingabedaten ohne großen Aufwand erweitert werden können. Da diese Methoden eine Vielzahl von Datenmengen verarbeiten, um Ergebnisse zur weiteren Analyse auszuliefern, soll dem Nutzer möglich sein, seine eigenen ausgewählten Datenressourcen zu verwenden, ohne den Analyseprozess zu beschränken oder beschädigen.

Wartbarkeit

Durch die Verwendung der im Bereich der Datenanalyse verschiedenartigen unterstützten Funktionen bzw. Bibliotheken von Python, soll jegliche Methode eine möglichst einfache Umsetzung von spezifizierten Verbesserungen oder Anpassungen ermöglichen.

4 Datensätze

Der erste Schritt zur Durchführung der bevorstehenden Textähnlichkeitsanalysen ist die Datenerfassung. Im folgenden Kapitel wird der in dieser Arbeit verwendete Datensatz präsentiert und etwas genauer untersucht. Zunächst wird eine allgemeine Beschreibung der Grundmerkmale des Dateninhalts enthalten. Danach wird eine Erläuterung gegeben, warum diese spezifischen Daten ausgewählt bzw. verwendet werden, gefolgt von tieferen Einblicken in die statistischen Merkmale der untersuchten Daten.

4.1 Daten Beschreibung

Der ausgewählte Datensatz wurde von [15] erstellt und infolge von der *Kaggle-Online-Community* extrahiert, die hauptsächlich als eine Wettbewerbsplattform für Datenwissenschaft und maschinelles Lernen bezeichnet wird. Dieser Satz enthält alle 16,786 Artikel aus der einflussreichen *New York Times*-Tageszeitung im Zeitraum vom 1. Januar 2020 bis zum 31. Dezember 2020. Sie wurden mit dem *nytimes-scraper* Package über die New-York-Times-API gesammelt und zusätzlich für weitere flexible Nutzung gepflegt. Dabei besitzen die gebundenen Artikel insgesamt 11 Merkmale (Features), die zwischen nominalen, kontinuierlichen und textbasierten Features variieren.

	newsdesk	section	subsection	material	headline	abstract	keywords	word_count	pub_date	n_comments	uniqueID
0	Editorial	Opinion	NaN	Editorial	Protect Veterans From Fraud	Congress could do much more to protect America...	['Veterans', 'For-Profit Schools', 'Financial ...	680	2020-01-01 00:18:54+00:00	186	nyt://article/69a7090b-9f36-569e-b5ab-b0ba5bb3...
1	Games	Crosswords & Games	NaN	News	'It's Green and Slimy'	Christina Iverson and Jeff Chen ring in the Ne...	['Crossword Puzzles']	931	2020-01-01 03:00:10+00:00	257	nyt://article/9edddb54-0aa3-5835-a833-d311a76f...
2	Science	Science	NaN	News	Meteor Showers in 2020 That Will Light Up Nigh...	All year long, Earth passes through streams of...	['Meteors and Meteorites', 'Space and Astronom...	1057	2020-01-01 05:00:08+00:00	6	nyt://article/04bc90f0-b20b-511c-b5bb-3ce13194...
3	Science	Science	NaN	Interactive Feature	Sync your calendar with the solar system	Never miss an eclipse, a meteor shower, a rock...	['Space and Astronomy', 'Moon', 'Eclipses', 'S...	0	2020-01-01 05:00:12+00:00	2	nyt://interactive/5b58d876-9351-50af-9b41-a312...

Abbildung 4.1: Die ersten vier Zeilen des Datensatzes.

Abbildung 4.1 zeigt die Grundstruktur des Datensatzes. Die Spalten *newsdesk*, *section*, *subsection* und *material* enthalten verschiedene Artikel-Tagging-Merkmale, die im Grunde genommen die Hauptkategorisierung der Artikelthemen darstellen. Diese nominalen Features sollen später als Hilfsmittel für eine genauere Auswertung dieses Analyseprojektes dienen. Die Spalte *headline* vermittelt jedoch das Hauptthema des Artikels und unterstreicht dessen Bedeutung für den Leser. Wie der Name schon sagt, enthält die Spalte *abstract* eine Zusammenfassung des Inhalts für alle Artikel. Sie gilt als zeitsparende Abkürzung für vielbeschäftigte Forscher sowie als Leitfaden für die wichtigsten Teile des schriftlichen Inhalts. Ein wichtiges kontinuierliches Merkmal lässt sich darüber hinaus in der Spalte *keywords* zeigen, die den Inhalt jedes Artikels beschreibt. Da Schlüsselwörter im Allgemeinen als ein Werkzeug bezeichnet werden, mit dem Indexer und Suchmaschinen relevante Artikel finden können, sind sie normalerweise sehr spezifisch für das Feld oder Unterfeld dessen Inhalt. Ein weiteres textbasiertes Merkmal stellt die Spalte *word_count* dar, die die Anzahl der Wörter pro Artikel gibt. Auf der anderen Seite bildet die Spalte *pub_date* die verschiedenen Veröffentlichungsdaten der Artikel ab. Schließlich bleiben die Spalten *n_comments* und *uniqueID* übrig. Diese geben die Anzahl der Leserkommentare sowie die dazugehörige URL pro Artikel.

Die Entscheidung bei der Auswahl dieser spezifischen Daten basiert auf mehreren Gründen. Die New York Times ist eine der beliebtesten Nachrichtenplattformen der Welt und wird täglich von unzähligen Benutzern besucht. Sie zählt zu einer der wenigen Medienpublikationen, die eine breite und öffentlich zugängliche Palette von APIs - derzeit 10 APIs - anbieten. Deshalb neigen viele Wissenschaftler dazu, Daten aus ihren verschiedenen APIs zur weiteren Forschung zu extrahieren. Ein weiterer wichtiger Punkt ist die gesamte Größe des Datensatzes. Mit über 16,000 Einträgen bietet diese Kollektion eine ausreichende Anzahl von Datenelementen, um eine breite Menge unterschiedlicher Experimente durchzuführen und zu angemessenen Ergebnissen zu gelangen als mit einem etwas kleineren Datensatz. Darüber hinaus spielt die weitreichende Klassifizierung zwischen den Datensatzelementen eine wichtige Rolle, um diese Daten nicht nur vielfältig zu gestalten, sondern auch bei der Ausgabenevaluation der verschiedenen Analysen zu helfen.

4.2 Explorative Datenanalyse

Newsdesk/ Section/ Subsection/ Material

Die Features Newsdesk, Section, Subsection und Material der Artikel sind wahrscheinlich die leistungsstärksten Prädiktoren für die zu berechnenden Textähnlichkeiten. Die verwendeten Daten besitzen insgesamt 181 dieser Features mit 63 Newsdesk-, 42 Section-, 66 Subsection- und 10 Material-Typen.

Abbildung 4.3a zeigt, dass die Spalte *newsdesk* viele verschiedene Typen von Artikeln umfasst. Mit 11,8% ist die Mehrheit der Newsdesks von dem Typ OpEd, gefolgt von 6,3% für Foreign und 6,2% für Culture. Opinion Editorials (hier als OpEds bezeichnet) ziehen deswegen viel eher die Aufmerksamkeit der Leser auf sich, da sie wahrscheinlich so geschrieben sind, dass sie Kontroversen hervorrufen. Sie befassen sich mit aktuellen Ereignissen und Problemen und versuchen, Standpunkte auf der Grundlage einer objektiven Analyse und widersprüchlichen bzw. gegensätzlichen Meinungen zu formulieren. NYT Opinion-Artikel sind außerdem immer offen für Kommentare, was natürlich die Wahrscheinlichkeit erhöhen würde, solche Artikel zu lesen und daher zu veröffentlichen.

Angesichts der Tatsache, dass interaktive Artikel und redaktionelle Meinungsbeiträge in der Regel beliebter sind, ist es vorhersehbar, dass Opinion-Artikel in der Spalte *section* wieder einen großen Anteil aller enthaltenen Artikeltypen ausmachen. Abbildung 4.3b zeigt einen Anteil von 14% für US- und 13,5% für Opinion-Artikel. In Abbildung 4.3c wird deutlich, dass Typen wie Middle East, Eat, Politics usw. den Mehrheitsanteil von 23,8% in der Spalte *subsection* bilden. Abbildung 4.3d stellt allerdings dar, dass die meisten Artikel in der Spalte *material* mit insgesamt 77,9% als News gekennzeichnet sind, gefolgt von Op-Ed mit 12,2%.

Headline

Die Daten enthalten 217 doppelte Überschriften. Dies sind wiederkehrende wöchentliche oder monatliche Artikel. Die ersten fünf wiederholten Überschriften werden wie folgt aufgelistet:

- Coronavirus in N.Y.C.: Latest Updates (**32** Mal)
- Variety: Acrostic (**25** Mal)
- What the Heck Is That? (**17** Mal)
- Homes for Sale in New York and New Jersey (**16** Mal)
- Homes for Sale in New York and Connecticut (**14** Mal)

Word Count

Wie in Abbildung 4.2 angezeigt, beträgt die durchschnittliche Länge der enthaltenen Artikel 1.301 Wörter. Es ist auch erkennbar, dass es eine große Anzahl von Artikeln gibt, die eine Wortzahl von 0 besitzen. Dies weist auf interaktive Merkmale hin, die keine echten "Wörter" im herkömmlichen Sinne haben.

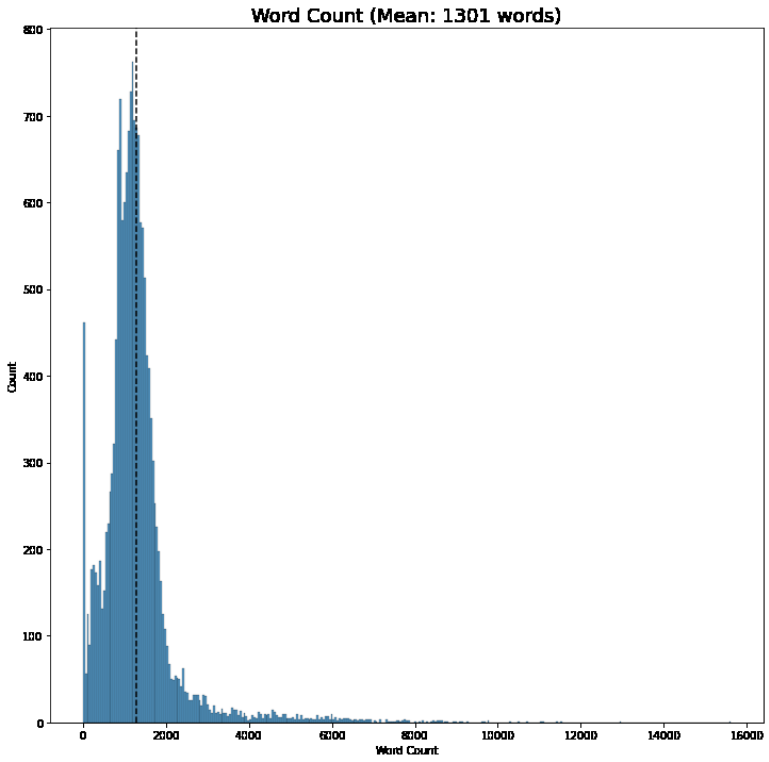
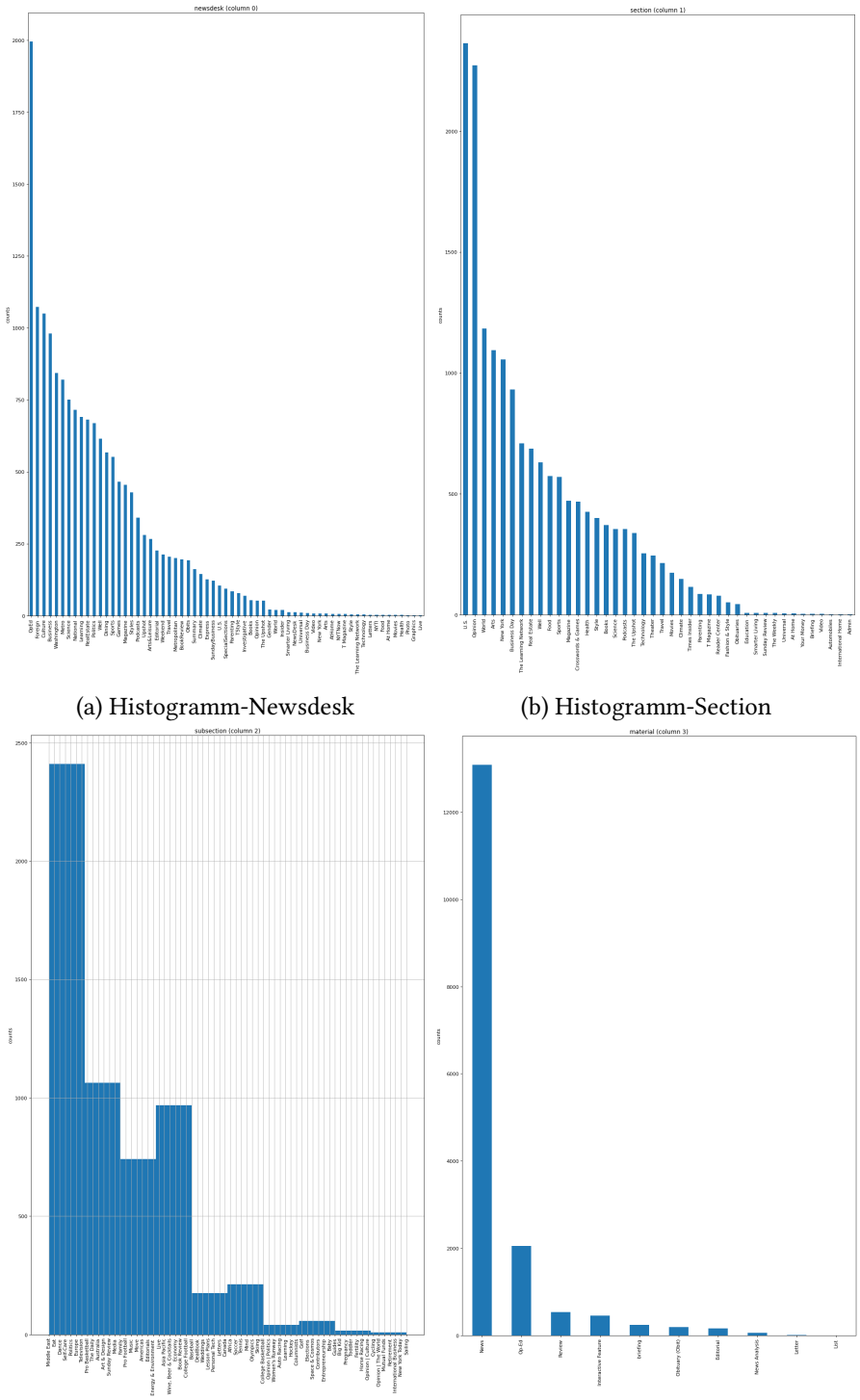


Abbildung 4.2: Wortzahlverteilung.

Insgesamt lassen die untersuchten Eigenschaften darauf schließen, dass der Dateninhalt ziemlich vielfältig ist. Das ermöglicht eine große Anzahl von Experimenten, die später zu genaueren Ergebnissen führen werden.



(a) Histogramm-Newsdesk

(b) Histogramm-Section

(c) Histogramm-Subsection

(d) Histogramm-Material

Abbildung 4.3: Histogramme verschiedener Datensatzspalten.

5 Methoden

Die in dieser Arbeit durchzuführenden Experimente bestehen aus zwei Hauptschritten: Feature-Extraktion und Definition von Ähnlichkeitsmetriken. Zuerst soll ein Weg definiert werden, um die Ähnlichkeit mathematisch zu messen. Dies soll weiterhin vergleichbar sein, damit Maschinen uns sagen können, welche Artikel basierend auf dem Datensatz aus Kapitel 4 am ähnlichsten oder welche am wenigsten ähnlich sind. Daher wird der Text aus diesen Artikeln vorverarbeitet und in einer quantifizierbaren Form bzw. einem mathematischen Objekt dargestellt (im Rahmen dieser Arbeit eine Vektorform ist), damit Ähnlichkeitsmetriken in einem weiteren Schritt ausgeführt werden können. Dieser Vorgang wird während des Feature-Extraktionsschritts realisiert. Danach werden spezifische Ähnlichkeitsalgorithmen ausgewählt bzw. definiert, die auf die nun vektorisierten Daten angewendet werden, und somit Ähnlichkeitsberechnungen durchführen.

Das vorliegende Kapitel stellt die verschiedenen Repräsentationsmethoden und Algorithmen vor, die in dieser Arbeit weiter verwendet werden, indem es einen detaillierten Einblick in ihre Arbeitsweise und den Grund gibt, warum sie für die Experimente ausgewählt wurden.

5.1 Feature-Extraktion

Die meisten Webdokumente enthalten Textdaten, die sowohl hochdimensional als auch unstrukturiert sind. Dies bedeutet, dass jedes einzelne Wort als separate Dimension betrachtet werden kann, und dass diese Dokumente im Allgemeinen sehr unterschiedlich strukturiert sind. Aus diesem Grund kann es äußerst schwierig sein, Ähnlichkeitsmetriken auf einen Rohtext anzuwenden. Daher müssen die charakteristischsten Merkmale eines Textes zunächst extrahiert werden, um so die Dimensionalität zu verringern. Dieser Vorgang wird generell als Feature-Extraktion oder Text-Mining bezeichnet.

Das Extrahieren von Textmerkmalen ist ein wesentlicher Schritt beim Data-Mining und Information-Retrieval. Es quantifiziert die Merkmalswörter, die aus dem Text extrahiert wurden, um die Textinformationen darzustellen, und konvertiert sie von einem unstrukturierten Originaltext in strukturierte Informationen, die der Computer erkennen und verarbeiten kann

[69]. Dies bedeutet also, den Text beschreiben, ihn durch die Dimensionsreduzierung des Textwortraums ersetzen und anschließend sein mathematisches Modell erstellen.

Wie bereits erwähnt, müssen Daten in ein numerisches Format konvertiert werden, damit Computer maschinelle Lernaufgaben darauf ausführen können. Dafür werden verschiedene Methoden zur Feauture-Extraktion verwendet, die diese Daten codieren. In diesem Abschnitt werden einige ausgewählte Techniken zur Abbildung von Texten in numerische Feature-Vektoren vorgestellt, die später zur Berechnung der Textähnlichkeit durch die Experimente verwendet werden.

Die Ansätze, die als primäre Feature-Extraktionsmethoden zum Vergleich ausgewählt wurden, sind *term frequency-inverse document frequency (TF-IDF), Dence Vector Representations*, einschließlich *Word2Vec & Paragraph Vector*, und *BERT*. Sie werden in zwei häufig verwendete Hauptmethoden grob eingeteilt: *Bag-of-Words* und *Word-Embeddings*.
Aus der Vielzahl der verfügbaren Methoden wurde TF-IDF ausgewählt, da es sich um eine sehr beliebte und aktuelle Strategie im Bereich Text-Mining handelt, die den Extraktionsprozess erheblich vereinfacht. Diese Methode extrahiert die aussagekräftigsten Begriffe in einem Dokument, was häufig zu signifikanten Ergebnissen führt. Allerdings wird beim Extrahieren von Wörtern mit TF-IDF jedes Wort im Text als separate Einheit behandelt, sodass die semantischen Merkmale des Textes nicht effektiv erhalten werden können. Zum Beispiel sollten *Kid* und *Child* zu demselben Konzept in unserem Leben gehören. Allerdings behandelt die traditionelle TF-IDF-Methode diese zwei Wörter als unterschiedliche Konzepte, was zum Verlust von Textinformationen führt, die durch die Textmerkmale dargestellt werden. Infolgedessen haben frühere Studien gezeigt, dass semantisch bedeutsame Darstellungen von Wörtern und Text durch neuronale Einbettungsmodelle erfasst werden können [36]. Insbesondere haben die Dense-Vector-Repräsentationsmodelle bei einigen Aufgaben der NLP eine beeindruckende Leistung gezeigt. Das erklärt somit, warum die Wahl weiter auf den beiden Modellen Word2Vec und Paragraph Vector fiel. Die Verwendung dieser Modelle trägt im Rahmen dieser Arbeit dazu bei, die Frage zu beantworten, ob neuronale Netze generell eine bessere Leistung als andere Methoden zur Erzeugung von Merkmalen aufweisen. Zuletzt wurde BERT als Vertreter der neueren unüberwachten Techniken zur Sprachdarstellung ausgesucht. Das neue Modell bietet große Verbesserungen beim Kontextverständnis und schlägt bereits erhebliche Wellen in der Welt der NLP.

5.1.1 Bag-of-Words

Eine der typischen Arten von Feature-Eextraktionsmodellen heißt Bag-of-Words [72]. Der Name Bag-of-Words bezieht sich auf die Tatsache, dass solche Modelle die Reihenfolge der Wörter gar nicht berücksichtigen. Stattdessen kann man sich vorstellen, dass jedes Wort in eine Tasche gesteckt wird, in der die Reihenfolge der Wörter verloren geht. Dabei wird jedoch gezählt, wie oft ein Wort in einem Dokument vorkommt. Das Ergebnis wird dann als Vektor der Worthäufigkeiten dargestellt, wobei der Vektor so lang wie das Vokabular ist. Wenn man *["cats", "dogs", "I", "like", "you"]* als Vokabular betrachtet, wird der Satz *I like cats* als *[1, 0, 1, 1,0]* dargestellt. Auf diese Weise bleiben die Häufigkeiten der Begriffe erhalten, obwohl Grammatik und Reihenfolge verloren gehen.

Bag-of-Words-Modelle werden häufig beim Clustering, Klassifizieren und Modellieren von Themen verwendet, indem spezielle Wörter und relevante Terminologien abgewogen werden.

TF-IDF

Term Frequency - Inverse Document Frequency (kurz als TF-IDF bezeichnet) [82] ist eine der beliebtesten Techniken zum Information-Retrieval, mit der die Relevanz von Begriffen in Dokumenten in Bezug auf eine Abfrage bestimmt werden kann [9]. Beispielsweise wird *the* häufig in Dokumenten verwendet, sodass TF-IDF diesen Ausdruck nicht als wichtig für die Charakterisierung von Dokumenten betrachtet. Im Gegensatz dazu wird *Python* in IT-relevanten Themen verwendet, sodass es als wichtiges Merkmalswort zum Erkennen von Themen und Kategorien betrachtet wird. Diese Methode kann auch als eine Form des Bag-of-Words-Modells angesehen werden, da weder Grammatik noch Reihenfolge dabei berücksichtigt werden.

TF-IDF besteht aus zwei Schritten: Zuerst wird die Termfrequenz *TF* und dann die inverse Dokumentfrequenz *IDF* berechnet. Der erste Schritt zur Berechnung von TF besteht darin, zu ermitteln, wie oft ein Term in einem Dokument vorkommt. Der Grund hierfür ist, dass Wörter, die häufig in einem Dokument vorkommen, wahrscheinlich wichtiger sind. Das Ergebnis wird dann normalisiert, indem es durch die Anzahl der Wörter im gesamten Dokument dividiert wird. Diese Normalisierung wird durchgeführt, um eine Tendenz zu längeren Dokumenten zu verhindern, so dass die Häufigkeit des Auftretens des Begriffs und nicht nur die Rohzahl des Begriffs erhalten wird [38].

$$tf_{t,d} = \frac{n_{t,d}}{\sum n_{k,d}} \tag{5.1}$$

wobei $n_{t,d}$ die Häufigkeit ist, mit der der Term t in Dokument d vorkommt, und $n_{k,d}$ die Anzahl der Vorkommen jedes Terms in Dokument d ist.

Um den IDF-Teil der gesamten Formel zu berechnen, wird die Gesamtzahl der Dokumente im Korpus durch die Anzahl der Dokumente dividiert, in denen ein Begriff erscheint. Das Ergebnis wird dann logarithmiert.

$$idf_t = \log \frac{|D|}{|D|_t} \tag{5.2}$$

wo $|D|$ die Gesamtzahl der Dokumente ist, und $|D|_t$ die Anzahl der Dokumente ist, in denen der Begriff t erscheint.

Durch Multiplikation der beiden Teile TF und IDF für einen bestimmten Begriff wird ein Maß dafür erhalten, wie kennzeichnend dieser Begriff ist.

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \tag{5.3}$$

5.1.2 Word-Embeddings

Bei der Verwendung von Worteinbettungsmodellen werden Wörter in einem reellen Vektorraum dargestellt [67]. Solche Modelle erfassen den Kontext und die Semantik von Wörtern im Gegensatz zu Bag-of-Words-Modellen, die nur die Anzahl der Wortvorkommen in Dokumenten darstellen, ohne Beziehungen oder Kontexte zu erkennen. Andererseits bewahren Worteinbettungsmodelle Kontexte und Beziehungen von Wörtern, so dass sie ähnliche Wörter genauer erkennen können. Damit würde ein gutes Worteinbettungsmodell idealerweise Wörter so darstellen, dass zwei verschiedene Wörter mit ähnlichen semantischen Bedeutungen ähnliche Vektorraumdarstellungen haben würden. Dabei können alle anderen sprachlichen Beziehungen zwischen den Wörtern ebenfalls beibehalten werden. Also, wenn diese Vektorraumdarstellungen für Wörter beispielsweise verwendet werden, werden die Operationen *König - Mann + Frau* ein Ergebnis ergeben, das der Vektorraumdarstellung für das Wort *Königin* sehr ähnlich ist [58].

Word2Vec

Ein häufig verwendetes Worteinbettungsmodell ist Word2Vec [51]. Die Hauptannahme, auf die sich dieses Modell stützt, ist die folgende: Wörter mit ähnlichen Kontexten haben eine ähnliche Bedeutung. Dies ist jedoch keine neue Idee, da sie bereits in [22] vorgeschlagen wurde. Allerdings hat sich das Training eines Modells unter dieser Voraussetzung als überraschend effektiv erwiesen. Dabei beginnt Word2Vec mit einer Reihe von Wortvektoren, die zufällig initialisiert werden. Es wird dann trainiert, indem der Korpus nacheinander gescannt wird, wobei immer ein Fenster mit fester Größe beibehalten wird. Dieses Fenster enthält ein zentrales

Zielwort und einige benachbarte Wörter, die als Kontext bezeichnet werden, wobei ihre Größe die im Kontext während des Trainings verwendeten Wörter auf den doppelten Wert begrenzt. Beispielsweise umfasst die Fenstergröße 5 die 5 Wörter links und die 5 Wörter rechts für jedes beobachtete Wort im Satz als Kontext. Das Erhöhen der Fenstergröße kann Kontextwörter enthalten, die für das aktuelle Wort nicht relevant sind. Durch Verringern der Fenstergröße können jedoch Beziehungen zwischen Wörtern und Stoppwörtern erfasst werden, was ebenfalls nicht bevorzugt wird.

In der ursprünglichen Veröffentlichung von Word2Vec [51] wurden zwei alternative Techniken vorgeschlagen: *CBoW* und *Skip-Gram*.

- CBoW steht für Continuous Bag of Words und lernt die Worteinbettungen so, dass angesichts des Kontexts das Zielwort vorhergesagt wird
- Skip-Gram sagt jedes Kontextwort anhand des Zielworts voraus

Abbildung 5.1 zeigt die Architektur der CBOW- und Skip-Gram-Modelle, wobei $w(t)$ das Wort in einem Satz am Index t bezeichnet. In einem Fenster der Größe N reicht der Kontext von $w(t-N)$ bis $w(t+N)$ ohne $w(t)$, daher der Term $w(t \pm 1..N)$.

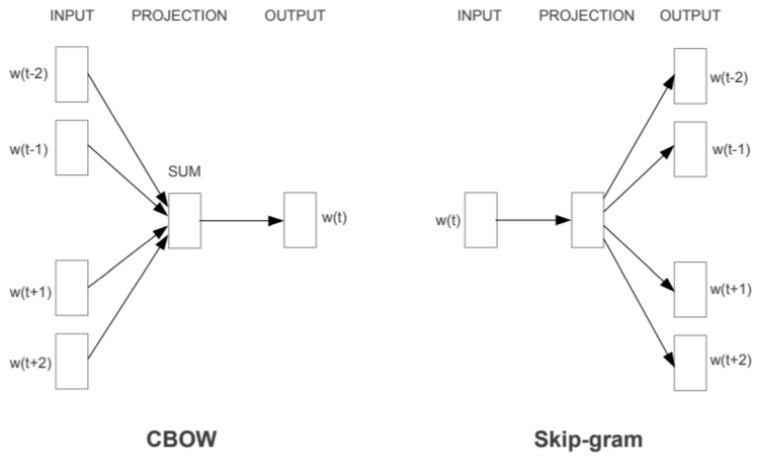


Abbildung 5.1: In einem Fenster der Größe N verwendet das CBOW-Modell den Kontext $w(t \pm 1..N)$, um das aktuelle Wort $w(t)$ vorherzusagen, während das Skip-Gram den Kontext für das aktuelle gegebene Wort vorhersagt [51].

Die Anzahl der Vektordimensionen gibt die Größe der in Abbildung 5.1 gezeigten Projektionsschicht (Projection Layer) an. Die verborgene Schicht (Hidden Layer) wird während

des Trainings des neuronalen Netzwerks wie in herkömmlichen neuronalen Netzwerken unüberwacht gelernt. Während die Projektionsschicht eine Matrix ist, die aus den verborgenen Schichtgewichten in Kombination mit der Eingabeebene (Input Layer) besteht, wobei die Matrix für alle Wörter im Vokabular geteilt wird [51].

Dichte Vektordarstellungen (Dense Vector Representations) extrahieren semantische Beziehungen basierend auf Word-Co-Occurrences im Datensatz, wobei die Genauigkeit der Darstellung von zwei Wörtern davon abhängt, wie oft das Modell diese Wörter im gesamten Korpus im selben Kontext sieht. Mehr Wort- und Kontext-Co-Occurences während des Trainings verändern die verborgene Darstellung, wodurch das Modell zukünftig erfolgreichere Vorhersagen treffen kann. Dies führt zu einer besseren Darstellung von Wörtern und Kontexten im Vektorraum.

Das Training eines neuen Word2Vec-Modells geht über den Rahmen dieser Arbeit hinaus. Daher wird eine vorab trainierte Word2Vec-Einbettung verwendet, die in Kombination mit der Word Mover's Distance-Ähnlichkeitsmetrik in Kapitel 6 eingesetzt wird.

Paragraph-Vektor

Das zuvor beschriebene Verfahren, Word2Vec, erzeugt Wortvektoren, die gemittelt werden, um den Dokumentvektor darzustellen. Damit Dokumente jedoch direkt dargestellt werden können, wurde in [36] den Paragraph-Vektor eingeführt, in der Literatur auch als Doc2Vec bezeichnet. Dabei handelt es sich um ein unüberwachtes Framework, das kontinuierlich verteilte Vektordarstellungen für Textteile lernt. Der Name Paragraph-Vektor soll die Tatsache hervorheben, dass die Methode auf Textstücke variabler Länge angewendet werden kann, von einer Phrase oder einem Satz bis zu einem großen Dokument. Bei diesem Verfahren hat jeder Absatz einen eindeutigen Vektor in der Matrix D und jedes Wort einen eigenen Vektor in der Matrix W (dieselbe lokale Kontextarchitektur wie Word2Vec). Diese Vektoren werden gemittelt und kombiniert, um das nächste Wort im Kontext in einem bestimmten Absatz vorherzusagen.

Es gibt zwei Hauptversionen von Doc2Vec: die *Distributed Memory- (PV-DM)* und die *Distributed Bag-of-Words*-Version *(PV-DBOW)*. In der Version mit Distributed Memory wird der Absatzvektor (Paragraph-Vektor) nur von Wörtern desselben Absatzes gemeinsam genutzt. Er kann als ein anderes Wort in dem Kontext dargestellt werden, der für alle Sätze und Fenster in diesem Absatz festgelegt ist. Somit wird das Thema des Absatzes gespeichert (oder auswendig gelernt). Aus diesem Grund wird diese Architektur als *Distributed Memory* bezeichnet, siehe Abbildung 5.2.

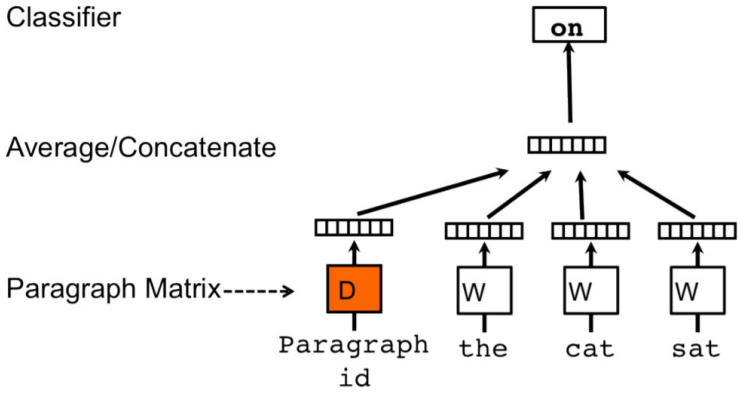


Abbildung 5.2: Das Doc2Vec Distributed Memory-Modell verwendet den Absatzvektor zusammen mit den lokalen Kontextwörtern, um das Wort $w(t)$ vorherzusagen. Es dient auch als einen Speicherplatz für das Thema des Absatzes [36].

Auf der anderen Seite ignoriert die Distributed Bag-of-Words-Variante das Kontextfenster und die Berechnung der Wortvektoren. Daher ist das Modell gezwungen, zufällig abgetastete Wörter im Dokument unter Berücksichtigung des Dokumentvektors vorherzusagen, wie in Abbildung 5.3 dargestellt. PV-DBOW benötigt weniger Speicherplatz, da es den Paragraph-Vektor nur aktualisiert, und die Wortvektoren daher ignoriert.

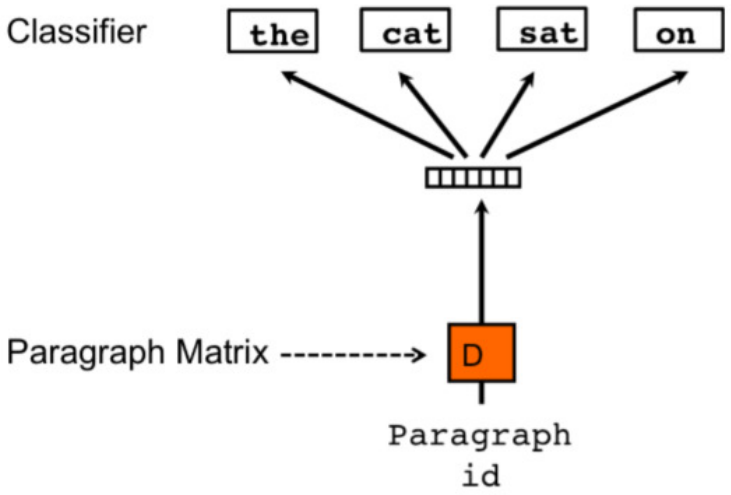


Abbildung 5.3: Das Doc2Vec Distributed Bag-of-Words-Modell versucht, eine zufällig ausgewählte Menge von Wörtern im Absatz unter Berücksichtigung dessen Absatzvektors vorherzusagen [36].

Der Nachteil von Doc2Vec ist seine rechnerische Komplexität, wenn die Absatzmatrix in Bezug auf die Anzahl der Dokumente an Größe zunimmt. Aufgrund des Fehlens eines negativen Stichprobenansatzes bei der Aktualisierung der Dokumentgewichte sind alle Dokumente am Lernprozess beteiligt. Also, um Vektoren für unsichtbare Dokumente zu generieren, müssen diese Dokumente zunächst dem Korpus hinzugefügt und das Modelltraining von Anfang an neu gestartet werden, was in der Industrie sehr schwierig sowie nicht gut skalierbar ist [10].

Der Vergleich von PV-DM und PV-DBOW geht über den Rahmen dieser Arbeit hinaus. Daher wird die Standardeinstellung der Implementierung aus der Python-Bibliothek verwendet.

BERT

BERT steht für *Bidirectional Encoder Representations from Transformers* [13]. Diese hochmoderne Technik wurde von Google entwickelt, um tiefe bidirektionale Darstellungen aus unbeschriftetem Text zu trainieren, indem sowohl der linke als auch der rechte Kontext in allen Ebenen gemeinsam konditioniert werden. Infolgedessen kann das vorab trainierte BERT-Modell mit nur einer zusätzlichen Ausgabeschicht verfeinert werden, um modernste Modelle für eine Vielzahl von Aufgaben wie Beantwortung von Fragen und Sprachinferenz ohne wesentliche Anforderungen zu erstellen.

Dieses Framework besteht aus zwei Hauptschritten: *Pre-training* und *Fine-tuning*. In Abbildung 5.4 wird der Mechanismus zur Beantwortung von Fragen (Question-Answering, oder kurz als QA bezeichnet) veranschaulicht, der in seiner Rolle als laufendes Beispiel für diesen Abschnitt dienen wird.

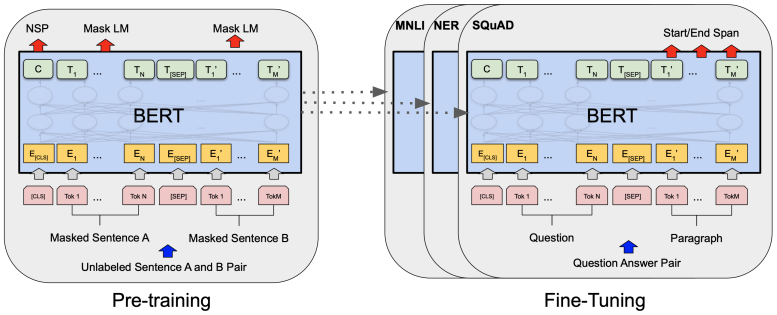


Abbildung 5.4: Allgemeine Vorbereitungs- und Feinabstimmungsverfahren bei BERT [13].

- **Pre-training**

BERT wird nicht mit den traditionellen Sprachmodellen von links nach rechts oder von rechts nach links vorab trainiert. Stattdessen werden zwei unüberwachte Aufgaben, *Masked Language Modelling (MLM)* und *Next Sentence Prediction (NSP)*, verwendet. Dieser Schritt ist im linken Teil der Abbildung 5.4 dargestellt.

Masked Language Modelling (MLM): Dieses Modell ist von der Cloze-Aufgabe inspiriert [71]. Unter Verwendung der WordPiece-Einbettungen [79] mit einem 30.000 Token-Vokabular maskiert das Modell zufällig 15% aller WordPiece-Eingabetoken. Dabei ist das Hauptziel, nur die maskierten Token vorherzusagen, anstatt die gesamte Eingabe zu rekonstruieren.

Next Sentence Prediction (NSP): Um das BERT-Modell zu trainieren, Satzbeziehungen zu verstehen, wird es für eine binärisierte Aufgabe zur Vorhersage des nächsten Satzes vorab trainiert, die trivial aus jedem einsprachigen Korpus generiert werden kann.

Für das Pre-training-Korpus wird das Books-Korpus (800 Millionen Wörter) [83] und die englische Wikipedia (2.500 Millionen Wörter) verwendet, wobei nur die Textpassagen ohne Listen, Tabellen oder Überschriften für die Wikipedia extrahiert werden.

- **Eingabe- /Ausgabedarstellungen**

Damit BERT eine Vielzahl von Downstream-Aufgaben bewältigen kann, ist seine Eingabedarstellung in der Lage, sowohl einen einzelnen Satz als auch ein Satzpaar (z. B. ⟨Frage, Antwort⟩) in einer Token-Sequenz eindeutig darzustellen. Dabei kann ein „Satz“ eine beliebige Spanne zusammenhängenden Textes und kein tatsächlicher sprachlicher Satz sein. Um diese Sätze jedoch zu unterscheiden, werden sie zunächst mit einem speziellen Token ($[SEP]$) getrennt. Zweitens wird jedem Token eine gelernte Einbettung hinzugefügt, die angibt, ob dieses Token zu Satz A oder Satz B gehört.

Für ein gegebenes Token wird die Eingabedarstellung durch Summieren dessen entsprechenden Token-, Segment- und Positionseinbettungen konstruiert. Eine Visualisierung dieser Konstruktion ist in Abbildung 5.5 zu sehen.

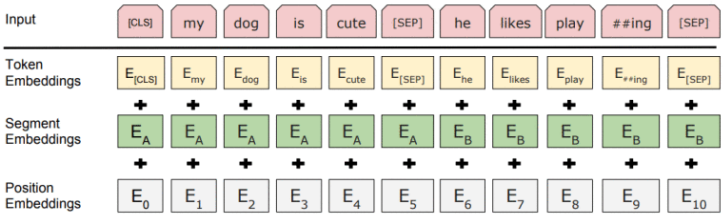


Abbildung 5.5: BERT-Eingabedarstellung [13].

- **Fine-tuning:**

Für jede Aufgabe werden die aufgabenspezifischen Ein- und Ausgaben einfach in BERT eingesteckt und alle Parameter von Ende zu Ende fein abgestimmt. Bei der Eingabe sind Satz A und Satz B aus dem Pre-training analog zu den Frage-Passage-Paaren bei der QA-Aufgabe und bei der Ausgabe werden die Token-Darstellungen in eine Ausgabeschicht für verschiedene Aufgaben auf Token-Ebene eingespeist, in diesem Fall für QA.

5.2 Ähnlichkeitsmetriken

Sobald die Dokumente als Vektoren dargestellt sind, werden statistische Methoden verwendet, um die Ähnlichkeit zwischen den generierten Vektoren und daher zwei gegebenen Dokumenten zu berechnen. Es gibt verschiedene Ähnlichkeitsmetriken, sodass die Auswahl eines geeigneten Algorithmus von zentraler Bedeutung für die Durchführung verschiedener NLP-Aufgaben ist. Diese Aufgaben erfassen z. B. Klassifizierung, Information-Retrieval, Topic-Modeling oder den Entwurf eines effizienten und effektiven Empfehlungssystems.

Mit der Verfügbarkeit verschiedener Algorithmen wurden die folgenden Metriken ausgewählt: *Kosinusähnlichkeit (Cosine Similarity)* und *Word Mover's Distance*. Die erste Methode wurde aufgrund ihrer Beliebtheit unter anderen Ähnlichkeitsmetriken ausgewählt. Sie wird als die am weitesten verbreitete Methode, um zwei Vektoren zu vergleichen. Ein weiterer Vorteil dieser Metrik ist ihre Geschwindigkeit und die Tatsache, dass sie für spärliche Daten sehr nützlich ist. Auf der anderen Seite wurde Word Mover's Distance als eine Repräsentation neuerer, fortgeschrittenerer Generationen von Ähnlichkeitsmetriken ausgesucht. Sie basiert daher auf jüngsten Ergebnissen bei Worteinbettungen, die semantisch bedeutsame Darstellungen für Wörter aus lokalen Vorkommen in Sätzen lernen. Während frühere Ansätze von Feature-Extraktionsmethoden entweder mit syntaktischen oder semantischen Worteinbettungen arbeiten. Damit zielt die Word Mover's Distance sowohl auf den semantischen als auch syntaktischen Ansatz ab, um Ähnlichkeiten zwischen Textdokumenten zu erfassen.

5.2.1 Kosinusähnlichkeit

Eines der Standardmaße für die Ähnlichkeit von Dokumenten und Wörtern ist die Kosinusähnlichkeit [76, 12, 36]. Diese Metrik zeigt, wie zwei Vektoren in Bezug auf ihren Vektorwinkel zusammenhängen. Es ist also ein Urteil über die Orientierung und nicht über die Größenordnung. Wenn Dokumente als Vektoren dargestellt werden, kann der Ähnlichkeitsgrad als Korrelation zweier Dokumente gemessen werden, und dies wird normalerweise durch die

Verwendung des Kosinuswinkels zwischen den beiden vektorisierten Dokumenten quantifiziert [26].

Um die Ähnlichkeit zwischen einem Forschungspapierdokument d und einer Abfrage q zu finden, kann der Winkel θ zwischen ihnen betrachtet werden. Dies wird durch Verwendung der folgenden Gleichung erreicht:

$$SIM_{\cos\theta(\vec{q_t},\vec{d_t})} = \frac{\vec{q_t}.\vec{d_t}}{|\vec{q_t}| \times |\vec{d_t}|} \tag{5.4}$$

Die Funktion $\cos\theta$ misst die Ähnlichkeit zwischen den Begriffen der Abfrage (Zielpapier) und den Begriffen des Dokuments (Forschungspapier), wobei $q(t)$ und $d(t)$ m-dimensionale Vektoren über die Menge der Begriffe $T = \{t_1, ..., t_m\}$ sind.

5.2.2 Word Mover's Distance

Durch die Untersuchung des Verhaltens von Worteinbettungen hat Kusner et al. 2015 [34] eine neue Metrik für die Abstandsberechnung zwischen Textdokumenten eingeführt. Dies wurde als Word Mover's Distance bezeichnet.

Word Mover's Distance, oder kurz als WMD, nutzt die Ergebnisse fortschrittlicher Einbettungstechniken wie Word2Vec und Glove [58], die Worteinbettungen von hoher Qualität erzeugen, und auf sehr große Datenmengen skaliert werden. Diese Einbettungstechniken zeigen, dass semantische Beziehungen bei Vektor-Operationen an Wortvektoren häufig erhalten bleiben. Zum Beispiel ist Vektor (Berlin) - Vektor (Deutschland) + Vektor (Frankreich) nahe an Vektor (Paris). Dies legt nahe, dass Abstände zwischen eingebetteten Wortvektoren bis zu einem gewissen Grad semantisch bedeutsam sind. WMD nutzt diese Eigenschaft der Worteinbettungen, insbesondere das Skip-Gram-Modell von Word2Vec, und behandelt Textdokumente als eine gewichtete Punktwolke eingebetteter Wörter. Der Abstand zwischen zwei Textdokumenten A und B ist dann der minimale kumulative Abstand, den Wörter von Dokument A zurücklegen müssen, um genau der Punktwolke von Dokument B zu entsprechen. Abbildung 5.6 zeigt eine schematische Darstellung dieser neuen Metrik.

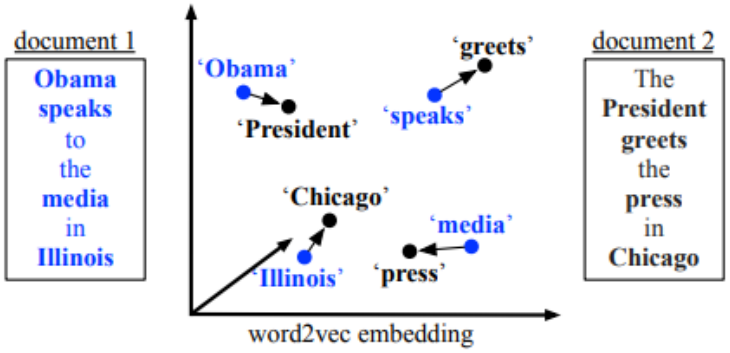


Abbildung 5.6: Eine Illustration von Word Mover's Distance. Alle Non-Stop-Wörter (fett markiert) beider Dokumente werden in einen Word2Vec-Raum eingebettet, und die Entfernung zwischen diesen beiden Dokumenten ist der minimale kumulative Abstand, den alle Wörter in Dokument 1 zurücklegen müssen, um mit Dokument 2 übereinzustimmen [34].

Angenommen, es gibt ein Quelldokument A und ein Zieldokument B. Eine Flussmatrix T ist definiert. Jedes Element in der Flussmatrix T_{ij}, gibt an, wie oft sich das Wort i (in Dokument A) in das Wort j (in Dokument B) umwandelt. Das heißt:

$$\sum_j T_{ij} = d_i \tag{5.5}$$

$$\sum_i T_{ij} = d'_j \tag{5.6}$$

wobei d und d' die nBOW-Darstellungen von zwei Textdokumenten im $(n-1)$-Simplex sind, d_i der ausgehende Fluss von Wort i ist und d_j der eingehende Fluss zum Wort j ist.

Schließlich kann der semantische Abstand zwischen den beiden Dokumenten als der minimale (gewichtete) kumulative Kosten definiert werden, der erforderlich ist, um alle Wörter von d nach d' zu verschieben. Also, wie folgt:

$$distance = min_{T \geq 0} \sum_{i,j=1}^{n} T_{i,j} c(i,j) \tag{5.7}$$

dabei ist $c(i,j)$ der Kostenwert, der mit dem „Zurücklegen" von einem Wort zum anderen verbunden ist.

Die der WMD zugrunde liegende Optimierung reduziert sich auf einen Sonderfall der Abstandsmetrik *Earth Mover's Distance*, kurz EMD [65, 53], ein gut untersuchtes Transportproblem, für das spezielle Löser entwickelt wurden [44, 57].

WMD hat mehrere interessante Eigenschaften:

- Sie ist hyperparameterfrei und einfach zu verstehen und zu verwenden
- Sie ist sehr gut interpretierbar, da der Abstand zwischen zwei Dokumenten aufgeschlüsselt und als spärlicher Abstand zwischen wenigen einzelnen Wörtern erklärt werden kann
- Sie enthält das im Word2Vec/Glove-Raum codierte Wissen und führt zu einer hohen Abrufgenauigkeit

6 Experimente

Basierend auf dem Hauptziel dieser Arbeit, unterschiedliche Techniken zur Messung der Textähnlichkeit zwischen Dokumenten zu bewerten, wird eine Reihe von Experimenten durchgeführt, um die Dokumentähnlichkeit als eines der wichtigsten Probleme von NLP weiter zu untersuchen. Insofern gibt dieses Kapitel einen Überblick über das experimentelle Konzept und Design. Darüber hinaus werden die technischen Details zu den verwendeten Python-Bibliotheken, -Tools und -Modellen sowie die konkrete Umsetzung dieser Experimente näher beschrieben.

6.1 Versuchsaufbau

Alle Experimente zielen darauf ab, Kombinationen verschiedener Feature-Extraktionsmethoden und Ähnlichkeitsmetriken zu vergleichen, um zu untersuchen, wie ein Algorithmus eine bessere Leistung im praktischen Sinne als ein anderer erbringen kann. Daher folgt jeder Test demselben systematischen Workflow, der in Abbildung 6.1 dargestellt ist, und in die folgenden Schritte weiter unterteilt wird.

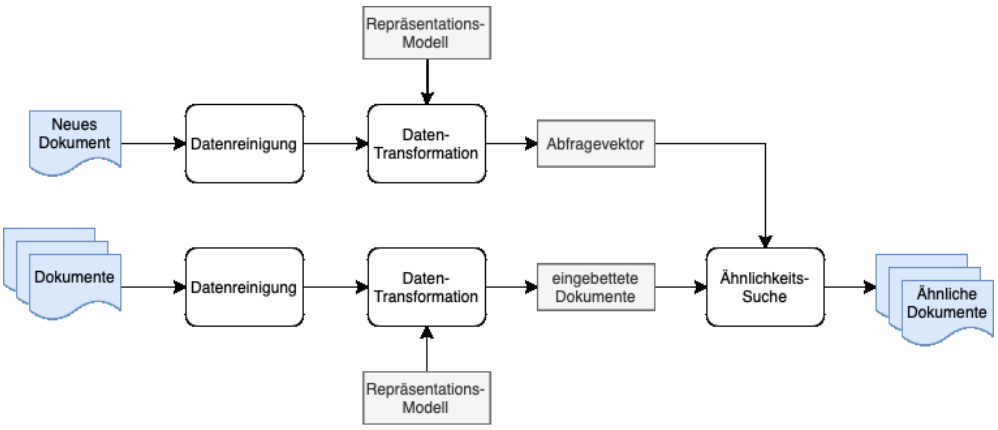


Abbildung 6.1: Experiment-Pipeline.

6.1.1 Datenreinigung

Die Datenvorverarbeitung ist einer der wichtigsten und anspruchsvollsten Schritte. Die Qualität der Daten und die daraus ableitbaren nützlichen Informationen haben einen großen direkten Einfluss auf die Fähigkeit der Modelle zur Verarbeitung dieser Daten. Daher werden in einem ersten Schritt die rohen Abstracts der Artikel (Titel werden dabei ignoriert) einem Datenbereinigungsmodul übergeben, das die Daten vorverarbeitet. Dies umfasst Folgendes:

- Kleinschreibung aller Texte
- Tokenisierung der Textdaten von Texten in voller Länge zu Wörtern und ggf. Sätzen
- Entfernen gängiger Stoppwörtern in Klauseln wie *of*, *a* , *an*
- Entfernen von Interpunktion, Sonderzeichen unter anderen Filtern für reguläre Ausdrücke
- Lemmatisieren, um verschiedene Wörter auf dasselbe Wurzelwort zu reduzieren

6.1.2 Datentransformation

Ab diesem Punkt wird der Begriff 'Repräsentationsmodell' verwendet, um die allgemeine Umwandlung eines Textes in eine geeignete Repräsentation zu beschreiben. Dabei handelt es sich normalerweise um einen Vektor der Merkmale des Textes, also um einen Vektor von Zahlen, damit dieser Text als Eingabedaten verwendet werden kann. Wie in Kapitel 5.1 beschrieben, können Vektordarstellungen von Texten auf viele verschiedene Weisen erstellt werden. Einmal gibt es die BoW-Darstellungsmodelle, die jedoch zu spärlichen Vektoren bzw. spärlichen Darstellungen führen, die beim Modellieren wieder zu mehr Speicher und Rechenressourcen führen. Dann gibt es die Worteinbettungsmodelle, die eine Dimensionsreduktion durch Verwendung dichter Darstellungen und eine ausdrucksstärkere Darstellung durch kontextbezogene Ähnlichkeit erreichen.

Zurück zur Phase der Datentransformation, wo die bereinigten und vorverarbeiteten Textelemente jeweils in numerische Vektoren in einem geeigneten Vektorraum umgewandelt werden, wird einer der folgenden Prozesse ausgeführt:

- Ein TF-IDF-BoW-Repräsentationsmodell wird auf die vorverarbeiteten Textelemente angewendet, um die Wörter in diesen Texten entsprechend zu gewichten
- Das Doc2Vec-Repräsentationsmodell wird auf den bereinigten vorverarbeiteten Textelementen trainiert, um Dokumentdarstellungen zu generieren

- Oder es werden zwei verschiedene vorab trainierte Repräsentationsmodelle (Word2Vec/BERT) verwendet, um Vektordarstellungen von den Textelementen direkt zu erstellen

Auf diese Weise kann sowohl die Syntaktik als auch die Semantik von Texten besser dargestellt, sodass die Textobjekte in den nachfolgenden Schritten analysiert, getestet und miteinander verglichen werden können. Dieser Schritt umfasst daher die verschiedenen Feature-Extraktionsmethoden, die in Kapitel 5.1 diskutiert wurden.

6.1.3 Suchabfragen

Um die Qualität der Ähnlichkeitsvorhersage tatsächlich zu bewerten, müssen die Repräsentationsmodelle an ungesehenen Stichproben ausgeführt werden. Somit werden drei neue Artikel-Abstracts wieder aus der *New York Times*-Tageszeitung als Basis-Suchabfragen ausgewählt. Jeder Artikel repräsentiert eine eigene Kategorie, wie in Tabelle 6.1 gezeigt.

Nr.	Title	Abstract	Newsdesk	Section	Keywords
1.	"*China's Climate Ambitions Collide with its Coal Addiction*"	"Beijing's new development blueprint is meant to steer the country to carbon neutrality before 2060, but companies and regions dependent on the fossil fuel aren't making it easy."	World	Climate	[Greenhouse Gas, Coal, Carbon Dioxide, National Energy Administration, Xi Jinping, China, Economy, Renewable Energy]
2.	"*NASA's Mars Helicopter Prepares for Its First Flight*"	"The experimental vehicle named Ingenuity traveled to the red planet with the Perseverance rover, which is also preparing for its main science mission."	Science	Science	[Mars, Helicopter, NASA, Perseverance Mars Rover, Space]
3.	"*Fear of Inflation Finds a Foothold in the Bond Market*"	"There is little evidence for a big jump in prices, but some economists and bond investors fear President Biden's policies could lead to inflation."	Business	Business	[US Economy, Inflation, Interest Rate, Deflation, Debt, Recession and Depression, American Rescue Plan, US Politics, Price, Stocks, Bonds, Government Bond, Federal Reserve, Treasury Department, Joe Biden, US]

Tabelle 6.1: Basisartikel.

Dementsprechend müssen die folgenden Schritte wiederholt werden, um die Suchabfragen für die abschließende Ähnlichkeitssuche vorzubereiten:

- Die Abfrage bereinigen
- Sie dann in einen Abfragevektor mit denselben Repräsentationsmodellen transformieren, die für die vorherigen Textelemente verwendet wurden

6.1.4 Ähnlichkeitssuche

Wenn ein Abfragevektor q eines Abfrageartikels a angegeben wird, werden die ähnlichsten Artikel abgerufen, je nachdem, wie nahe sie an a liegen. Diese werden unter Verwendung der verschiedenen in Kapitel 5.2 dargestellten Metriken abgerufen, die die Ähnlichkeit von gewichteten vektorisierten Textobjekten berechnen. D.h. sie messen, wie relevant ein Artikel für eine bestimmte gegebene Abfrage ist.

Dabei wird jede Ähnlichkeitssuche für die gesamten 16.787 Artikel ausgeführt, um die ähnlichsten Artikel mit den höchsten Werten zu finden. Dieser Vorgang wird für jeden der drei Basisartikel wiederholt.

6.2 Technische Implementierung

Python gilt unter Forschern als die beste Wahl für die Analyse von Big-Data. Sie ist für ihre Einfachheit und Effizienz bei der Integration von Systemen bekannt, und wird für viele Anwendungsbereiche verwendet. Zudem enthält sie viele eingebaute Bibliotheken, die einfache, dennoch sehr fortgeschrittene wissenschaftliche Berechnungen bieten, womit verschiedene NLP-Aufgaben ausgeführt bzw. gelöst werden können. Auf dieser Grundlage wurden alle Experimente in Python 3 mithilfe mehrerer Bibliotheken implementiert, die allgemein über den pip-Befehl auf der Shell installiert werden können.

6.2.1 Python-Bibliotheken

NLP benötigt einige wesentliche Tools für die Analyse von Big-Data und die Verarbeitung unstrukturierter Daten. Basierend darauf wurden verschiedene Programmiersprachen, Software, Tools und Bibliotheken entwickelt, um die Leistung von NLP zu verbessern. Heutzutage ist Python sicherlich die führende Open-Source-Sprache, die eine breite Palette benutzerfreundlicher Bibliotheken bietet. Zum Aufbauen der geplanten Ähnlichkeitsanalysen werden die folgenden Standardbibliotheken von Python verwendet:

NLTK

Das *Natural Language Toolkit* [5], kurz als *NLTK* bezeichnet, gehört zu den bekanntesten und leistungsstärksten Bibliotheken, die Python bei der Verarbeitung von Textdaten unterstützen. Dieses Toolkit bietet über 50 Korpora, einschließlich einer Vielzahl von Funktionen für die Textverarbeitung, wie Tokenisierung, Klassifizierung, Stemming, Lemmatisieren, Parts-of-Speech, Tagging, Parsing und semantisches Denken.

NumPy

Numerical Python, *NumPy* [21], ist eine Python-Bibliothek, die eine einfache, aber leistungsstarke Datenstruktur bietet: das n-dimensionale Array, das bei der wissenschaftlichen Berechnung großer mehrdimensionaler Arrays/Matrizen hilft.

Pandas

Pandas [48] ist ein Open-Source-Python-Paket, das auf NumPy basiert und sehr nützliche Datenstrukturen für die Verwaltung großer Datenmengen bietet. Es vereinfacht viele der zeitaufwändigen, sich wiederholenden Aufgaben, die mit der Datenverarbeitung verbunden sind, einschließlich: Importieren und Speichern von Daten, Bereinigen, Normalisieren, Visualisieren usw.

Scikit-learn

Für die meisten Algorithmen in dieser Arbeit wird die *Scikit-Learn*-Bibliothek [56] verwendet, eine Bibliothek in Python, die viele unüberwachte und überwachte Lernalgorithmen bereitstellt. Sie bietet eine Auswahl effizienter Tools für maschinelles Lernen und statistische Modellierung, einschließlich Klassifizierung, Regression, Clustering und Dimensionsreduktion. Im Gegensatz zu NumPy oder Pandas konzentriert sich diese Bibliothek auf die Modellierung von Daten und nicht auf das Laden, Bearbeiten oder Zusammenfassen von denen.

Scikit-learn bietet also eine Toolbox mit soliden Implementierungen einiger weit verbreiteter Algorithmen u. a. eine sofort einsatzbereite Implementierung von TF-IDF sowie eine praktische Aufrufmethode zur Berechnung der Kosinusähnlichkeit zwischen den resultierenden Vektoren, die während dieser Arbeit gründlich verwendet werden.

Gensim

Gensim [84] steht für *Generate Similarity*. Es ist eine kostenlose Python-Bibliothek, die Themenmodellierung und semantische Ähnlichkeit zwischen Dokumenten unterstützt. Deren Schwerpunkt liegt auf der statistischen Semantikanalyse von Dokumenten und dann der Bewertung anderer Dokumente anhand ihrer Ähnlichkeit. Gensim kann mit sehr großen Textmengen arbeiten, indem Dokumente an seine Analyse-Engine gestreamt und inkrementell unüberwacht gelernt werden. Damit können mehrere Modelltypen erstellt werden, die jeweils für unterschiedliche Szenarien geeignet sind: u. a. Word2Vec und Doc2Vec.

6.2.2 Repräsentationsmodelle

Im Folgenden wird auf die technischen Aspekte eingegangen, die bei der Umsetzung der Repräsentationsmodelle eingerichtet wurden:

TF-IDF

Wie bereits erwähnt, bietet die Scikit-learn-Bibliothek eine sehr einfache Implementierung von TF-IDF. Unter Verwendung der *TfidfVectorizer*-Klasse werden die Wortanzahl-, IDF- und die TF-IDF-Werte gleichzeitig berechnet, sodass die Eingabe einer Sammlung von Rohtexten in eine Matrix von TF-IDF-Merkmalsvektoren konvertiert wird.

Word2Vec

Für die Word2Vec-Einbettung wird ein vorab trainiertes Modell von Google verwendet, um die Ähnlichkeit zwischen Dokumenten zu bewerten. Dabei wird die Gensim-Bibliothek zum Importieren gebraucht wird, indem der Pfad der Modelldatei übergeben wird. Dieses Modell enthält 300-dimensionale Vektoren für 3 Millionen Wörter und Phrasen, die auf 100 Milliarden Wörtern aus dem *GoogleNews*-Datensatz [54] trainiert wurden, einem kostenlosen Open-Source-Datensatz von Google.

Doc2Vec

Leider wurde für Doc2Vec kein geeignetes vorab trainiertes Modell veröffentlicht. Aus diesem Grund wird Gensim verwendet, um ein Doc2Vec-Modell auf dem für diese Arbeit verfügbaren Korpus zu trainieren und daher Vektordarstellungen von den Textobjekten zu erstellen. Dabei wird das Distributed Memory-Modell als Trainingsalgorithmus verwendet, weil damit die Wortreihenfolge in einem Dokument beibehalten wird, während bei dem Distributed Bag of Words-Modell nur der Bag of Words-Ansatz verwendet wird, bei dem keine Wortreihenfolge beibehalten wird.

Dieses Doc2Vec-Modell wird mit 300 *Dimensionen* über eine Anzahl von 100 *Iterationen* getestet. Diese Durchläufe zeigen, wie oft das Modell auf dem Datensatz trainiert wird. Das wird besonders bevorzugt, da mehr Iterationen das Modell in Richtung Konvergenz treiben. Zusätzlich zu den vorherigen Parametern verfügt Doc2Vec über die Variable *Minimum Count*, mit der Wörter entfernt werden, die kleiner als der angegebene Wert sind (in diesem Fall gleich 1 ist). Dies hilft, verrauschte Wörter herauszufiltern.

BERT

Damit Bert die Dateneingaben einbetten kann, wird das vorab trainierte BERT-Modell von *Huggingface* [55] aus dem sentence-transformers-Repository verwendet. Dieses Repository ermöglicht das Trainieren und Verwenden von Transformer-Modellen zum Generieren von Satz- und Texteinbettungen. Somit wird ein Basismodell erhalten, das 12 Schichten (Transformer Blocks), 12 Aufmerksamkeitsköpfe (Attention Heads), 110 Millionen Parameter und eine versteckte Größe von 768 besitzt. Infolgedessen werden die semantisch bedeutsamen abgeleiteten Satz- und Texteinbettungen unter Verwendung von Kosinusähnlichkeit verglichen.

7 Evaluation

Das folgende Kapitel befasst sich mit dem Problem der Bewertung der Textähnlichkeitswerte, die aus den vorherigen Experimenten erhalten wurden. Zunächst wird der vorgeschlagene Bewertungsansatz beschrieben. Hierzu wird eine Liste von vier Kriterien bereitgestellt, die als Maße für die weitere Evaluation verwendet wird. Im Anschluss wird ein Überblick über die Methodenleistungen gegeben, indem die ersten Ergebnisse aller Experimente vorgestellt werden. Zudem werden weitere Bewertungsanalysen durch Hypothesentests durchgeführt, um die vorliegenden Ergebnisse zusätzlich auf Plausibilität zu prüfen. Abschließend werden die finalen Erkenntnisse aus den Experimenten zusammengefasst.

7.1 Bewertungskriterien

Soweit bekannt, gibt es keinen festen Ansatz oder eine feste Methode zur Bewertung von Ähnlichkeitsmaßen. Solange die Entwicklung und das Trainieren von Repräsentationsmodellen fortgesetzt wird, wird jede Methodik dadurch modernisiert und modifiziert. Daher wird keine Methodik hier vorgegeben, sondern ein einziger Ansatz vorgeschlagen, der auf einer Reihe von Kriterien basiert. Dieser soll die Eignung, die Unterschiede und den Erfolg der Textähnlichkeitsmethoden demonstrieren können.

Die folgenden Parameter reichen von der Erfassung qualitativer Aspekte bis hin zu messbaren und vergleichbaren Werten, anhand derer die Ergebnisse ausgewertet werden können. Für jeden dieser Parameter wird eine detaillierte und genaue Definition bereitgestellt.

Tag Overlap

Tags sind der engste Vertreter menschlicher Urteile in Bezug auf inhaltliche Ähnlichkeit. Die Journalisten selbst schreiben diese von Hand auf. Sie sind in der Spalte *keywords* in Abbildung 4.1 angezeigt. Das Beste an der Verwendung von Tags besteht darin, objektiv zu messen, wie stark sich zwei Inhalte überschneiden. Jedes Tag hat eine Größe von 1 bis 12 Schlüsselwörtern (Keywords), so dass je mehr sich zwei Artikel überschneiden, desto ähnlicher sie sind.

Newsdesk Overlap

Zweitens wird das Feature *newsdesk* eines Artikels berücksichtigt. Die New York Times kategorisiert ihre Artikel auf diese Weise auf höchster Ebene: Wissenschaft, Politik, Sport usw. Jedes Feature erfasst eine sogenannte Obergruppe, die sich auf ein bestimmtes Thema bezieht.

Section Overlap

Das dritte qualitative Kriterium ist das *section*-Feature. Beispielsweise kann ein Style-Newsdesk in Self-Care oder Welt in Europe unterteilt werden. Allerdings wird dieses Feature meistens wie das Newsdesk eingetragen, weshalb es keine entscheidende Rolle wie die beiden vorherigen Kriterien spielt.

Leistung

Der Hauptzweck der Leistungsmessung besteht darin, eine allgemeine Vorstellung von der Effizienz jedes Experiments in Bezug auf die Implementierungszeit und den Verbrauch verfügbarer Ressourcen zu vermitteln. Insbesondere, weil diese Experimente unterschiedliche Methodenkombinationen repräsentieren. Der im Rahmen dieser Arbeit vorgeschlagene Ansatz umfasst daher zwei Faktoren für die Leistungsstudie:

- **Ausführungszeit:** Die Zeit, die benötigt wird, um ein Experiment abzuschließen. Diese kann durch das Betriebssystem, mit dem ein bestimmter Prozess ausgeführt wird, und seine Speicherzuweisungsstrategie beeinflusst werden. Hier wird sie in Minuten gemessen bzw. gegeben.
- **Speichernutzung:** Dies ist die von der Anwendung verwendete Speichermenge, um einen bestimmten Prozess durchzuführen. Sie wird in Megabytes gemessen.

7.2 Initiale Ergebnisse

Als erster Schritt zur Bewertung der Methodenleistungen werden die Ergebnisse aller Experimente für jede Suchabfrage in den Tabellenabbildungen 7.1 bis 7.3 dargestellt.

Die folgenden Tabellenbeschreibungen bieten jeweils detaillierte Einblicke in die Performanz jeder Methodenkombination anhand der oben festgelegten Bewertungskriterien:

Inhaltsbeschreibung der Tabelle 7.1

Die erste Tabelle 7.1 handelt sich um eine Suchabfrage, die die globale Erwärmung beleuchtet, indem sie über Beijins Plan zur CO2-Neutralität vor 2060 spricht. Somit enthält diese Artikelabfrage viele spezifische Begriffe wie *CO2 Neutrality* und *Fossil Fuel*.

Hierbei hat TF-IDF keine gute Leistung mit nur **23%** erbracht, und sich jedoch mehr auf den Begriff *Companies* konzentriert hat, was zu einem Off-Topic-Ergebnis führte. Ein ähnliches Verhalten ist bei Doc2Vec zu sehen. Diese Methode hat einen Artikel als Ergebnis aufgegriffen, der mit dem Thema der globalen Erwärmung gar nicht zusammenhängt. Auf der anderen Seite gelang es BERT und Word2Vec, sich auf denselben verwandten Kontext zu konzentrieren und Artikel zur globalen Erwärmung im Hinblick auf das aktuelle Problem der Reduzierung fossiler Brennstoffe zu liefern. Diese Methoden haben jeweils einen Kosinus-Ähnlichkeitsprozentsatz [1] von **84%** und Word Mover's Abstandswert [2] von **2.6**, der relativ kurz ist, getroffen. Allerdings litten die Ergebnisse unter einem Mangel an ausreichenden Daten, weshalb keine der Methoden ein Ergebnis zu demselben Thema erfassen konnte. Dies spricht für die folgende entscheidende Bedeutung: Datensätze müssen nach besten Kräften gesammelt, diversifiziert und erweitert werden, um das beste Ergebnis für die Ähnlichkeitsübereinstimmung zu erzielen.

Inhaltsbeschreibung der Tabelle 7.2

Die zweite Abfrage in der Tabelle 7.2 befasst sich mit einem reinen wissenschaftlichen Ansatz, wo es viele Terminologien wie *Ingenuity*, *Red Planet* und *Perseverance Rover* enthalten sind. Solche Begriffe tauchen selten auf, und sind daher leicht zu erlernen. Deshalb haben sich die meisten Methoden hier relativ gut bewährt. Insbesondere haben TF-IDF und Word2Vec, jeweils mit einem Kosinus-Prozentsatz von **39%** und geringen Abstandswert von **2.6**, die Begriffe richtig getroffen bzw. wahrgenommen, die hauptsächlich über die spezielle wissenschaftliche Mission *Ingenuity* auf dem Mars gesprochen haben. BERT, mit **80%**, war auch ein guter Anwärter jedoch mit dem Fokus auf der *Chang'e-4*-Mission auf dem Mond anstelle des Mars. Während Doc2Vec einen Artikel im Lernrahmen aufgegriffen hat, was gar nicht zum Thema gehört.

[1] Die Kosinusähnlichkeit stellt einen Wert dar, der durch einen eingeschränkten Bereich von 0 und 1 begrenzt ist. Hierbei wurde dieser Wert mit 100 multipliziert, um den entsprechenden Prozentsatz anzuzeigen.

[2] Die Word Mover's Distance ist ein Maß für den Abstand zwischen eingebetteten Wortvektoren. Sie stellt daher einen Wert dar, der von 0 bis unendlich reicht.

Inhaltsbeschreibung der Tabelle 7.3

Im Vergleich zu den beiden vorherigen Artikelabfragen ist die letzte Abfrage in Tabelle 7.3 viel ereignisgesteuerter, zeitkritischer und trägt ein bestimmtes Substantiv, nämlich *Joe Biden*.

Da sich Wirtschaftsnachrichten in Bezug auf z.B. Börsenkurse oder Bons ständig ändern und in der Regel weit davon entfernt sind, einen statischen Stand beizubehalten, ändern sich dabei alle deren relevanten Nachrichten meistens im Verlauf der Ereignisse sehr schnell. Es ist also fair zu sagen, dass sich die erworbenen Ergebnisartikel inhaltlich sehr stark variieren, was mit den rapiden Veränderungen im Wirtschaft und somit an den Aktienmärkten zusammenhängt. In diesem Fall haben TF-IDF und BERT, mit jeweils **86%** und **30%**, die schnelle Entwicklung der wirtschaftlichen Situation anhand der eingegebenen Suchabfrage erfasst. Dabei haben sie den Rückgang der Aktienmärkte dargestellt. Allerdings hat Word2Vec einen Abstecher in ein wirtschaftliches Wohnproblem wie Immobilien und Wohnen gemacht. Doc2Vec hat sich anderseits auf das Wort *President* in einem wirtschaftlichen aber auch politischen Kontext konzentriert, was wieder zu einem ungenauen Ergebnis geführt hat.

1. Suchabfrage: *"China's Climate Ambitions Collide with its Coal Addiction"*					
Experiment	Ähnlichster Artikel	Ähnlichkeitswert	Tag Overlap	Newsdesk Overlap	Section Overlap
TF-IDF + Kosinusähnlichkeit	"Tech companies aren't going to dismantle the systems that are making them billions."	Kosinuswert = 0.23	0	OpEd (N)	Opinion (N)
Word2Vec + Word Mover's Distance	"Efforts to limit global warming often focus on emissions from fossil fuels, but food is crucial, too, according to new research."	WMD-Wert = 2.6	2	Climate (N)	Climate (Y)
Doc2Vec + Kosinusähnlichkeit	"Companies, unions and trade groups are making pitches that their workers should be prioritized."	Kosinuswert = 0.56	0	Metro (N)	New York (N)
BERT + Kosinusähnlichkeit	"Efforts to limit global warming often focus on emissions from fossil fuels, but food is crucial, too, according to new research."	Kosinuswert = 0.84	2	Climate (N)	Climate (Y)

Tabelle 7.1: Ergebnisse der Experimente zur ersten Suchabfrage.

2. Suchabfrage: *"NASA's Mars Helicopter Prepares for Its First Flight"*					
Experiment	Ähnlichster Artikel	Ähnlichkeitswert	Tag Overlap	Newsdesk Overlap	Section Overlap
TF-IDF + Kosinusähnlichkeit	"The third and final mission to the red planet of the month lifted off on Thursday."	Kosinuswert = 0.39	4	Science (Y)	Science (Y)
Word2Vec + Word Mover's Distance	"As part of its next Mars mission, NASA is sending an experimental helicopter to fly through the red planet's thin atmosphere."	WMD-Wert = 2.6	5	Science (Y)	Science (Y)
Doc2Vec + Kosinusähnlichkeit	"Look closely at this image, stripped of its caption, and join the moderated conversation about what you and other students see."	Kosinuswert = 0.68	0	Learning (N)	The Learning Network (N)
BERT + Kosinusähnlichkeit	"The Chang'e-4 mission, the first to land on the lunar far side, is demonstrating the promise and peril of using ground-penetrating radar in planetary science."	Kosinuswert = 0.80	1	Science (Y)	Science (Y)

Tabelle 7.2: Ergebnisse der Experimente zur zweiten Suchabfrage.

3. Suchabfrage: *"Fear of Inflation Finds a Foothold in the Bond Market"*					
Experiment	Ähnlichster Artikel	Ähnlichkeitswert	Tag Overlap	Newsdesk Overlap	Section Overlap
TF-IDF + Kosinusähnlichkeit	"Stocks fell and investors rushed to the safety of government bonds."	Kosinuswert = 0.30	4	Business (Y)	Business (Y)
Word2Vec + Word Mover's Distance	"Economists say rents would fall. Neighbours fear rents would rise. New research tries to find the answer."	WMD-Wert = 2.7	0	Business (Y)	Business (Y)
Doc2Vec + Kosinusähnlichkeit	"The president leads a transition to sickness."	Kosinuswert = 0.58	3	OpEd (N)	Opinion (N)
BERT + Kosinusähnlichkeit	"The drop in stocks and bond yields suggests investors think the Federal Reserve's interest-rate cut won't contain the economic impact of the coronavirus."	Kosinuswert = 0.86	4	Business (Y)	Business (Y)

Tabelle 7.3: Ergebnisse der Experimente zur dritten Suchabfrage.

In der nachstehenden Abbildung 7.1 sind die vier Messergebnisse zur Speichernutzung aus Tabelle 7.4 in einem Säulendiagramm aufgeführt. Unter Verwendung des MacOS Catalina-Betriebssystems zum Ausführen der Implementierungen, kann man direkt sehen, dass die Methodenkombination aus Word2Vec und WMD ein einschränkendes Verhalten für den steigenden Speicherbedarf darstellt und einen Schwellenwert von etwa 5,300 MB beibehält. Das ist nämlich auf den großen GoogleNews-Datensatz zurückzuführen, der allein mehr als 3,000 MB zum Laden benötigt, noch bevor die Word Mover's Distance-Methode ihre Ähnlichkeitsberechnungen ausführt. Darauf folgt die 2,600-MB-Speichernutzung bei Doc2Vec. Dieser Wert ergibt sich aus der Kombination von 300 Vektordimensionen mit 100 Iterationsrunden auf demselben Datensatz.

Darüber hinaus zeigt die zweitstehende Abbildung 7.2 die Ausführungszeit der vier Szenarien einschließlich aller Schritte (Tokenisierung, Vektorisierung und Vergleich) unter MacOS Catalina. Es ist klar, dass die von dem Betriebssystem verfolgte Speicher-Zuweisungsstrategie keinen spürbaren Einfluss auf die Ausführungszeit hat. Man kann das besonders bei Word2Vec und WMD feststellen, dessen durchschnittlichen Ausführungszeit nur etwa 3 Minuten dauerte. Im Vergleich zu BERT zusammen mit dem Kosinus-Ähnlichkeitsalgorithmus, dessen Ausführung die längste Zeit, durchschnittlich 45 Minuten, in Anspruch nahm, aber eine vergleichsweise geringere Speichernutzung als beim vorherigen Ansatz. Ein weiteres auffälliges Verhalten ist bei der TF IDF-Methode zu sehen, die nicht mal eine Minute bei allen Experimenten brauchte. Das zeigt allerdings, wie sich ihre schnelle Implementierung auf die Laufzeit auswirkte.

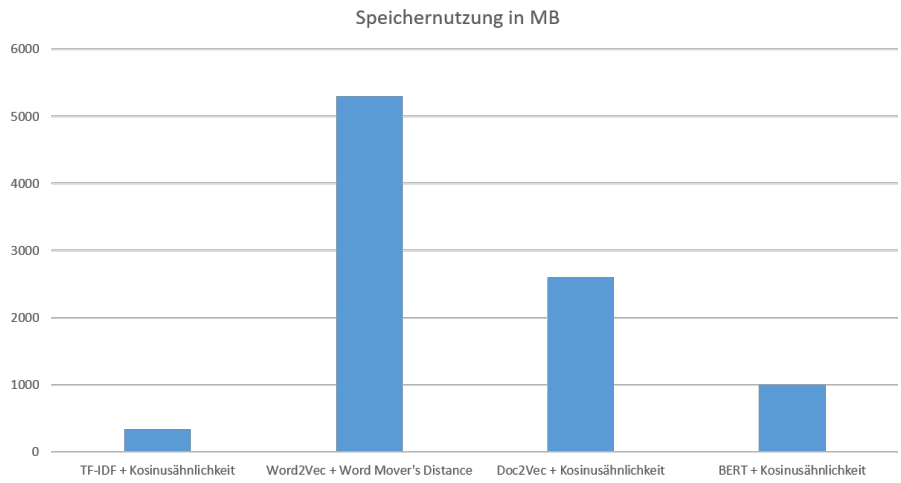


Abbildung 7.1: Speichernutzung der Experimente in Megabyte.

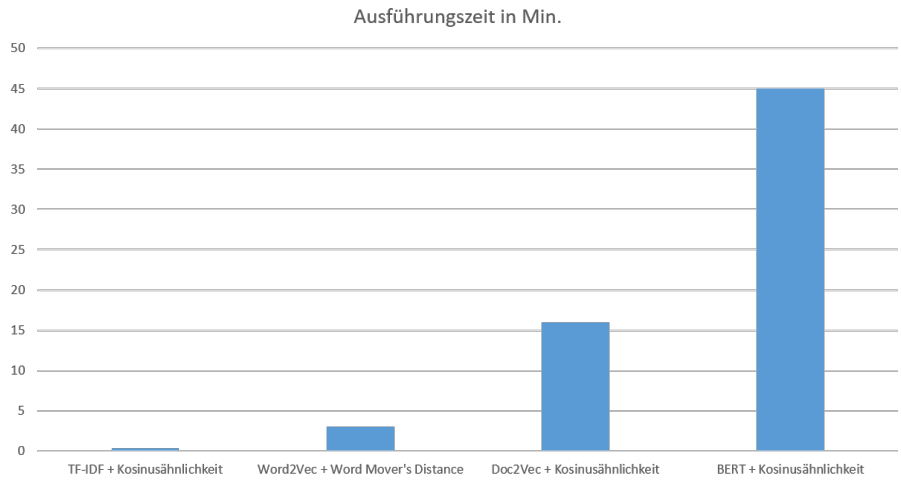


Abbildung 7.2: Ausführungszeit der durchgeführten Experimente in Minuten.

Experiment	Speichernutzung	Ausführungszeit
TF-IDF + Kosinus-Ähnlichkeit	347 MB	0.37 Min.
Word2Vec + Word Mover's Distance	5,300 MB	3 Min.
Doc2Vec + Kosinusähnlichkeit	2,600 MB	16 Min.
BERT + Kosinus-Ähnlichkeit	1,014 MB	45 Min.

Tabelle 7.4: Messergebnisse der Speichernutzung und Ausführungszeit der vier durchgeführten Experimenten.

7.3 Experimentelle Ergebnisse

Beim Hypothesentest ist eine Möglichkeit definiert, die Ergebnisse einer Umfrage oder eines Experiments zu testen, um festzustellen, ob ein Analyst aussagekräftige Ergebnisse hat. Dabei testet man im Grunde genommen, ob seine Ergebnisse gültig sind. Wenn sie jedoch zufällig entstanden sind, kann das Experiment somit nicht wiederholt werden und ist daher wenig nützlich. Das Testen von Hypothesen ist für fast jeden Sektor erforderlich und beschränkt sich nicht auf Statistiker oder Datenwissenschaftler. Wenn man beispielsweise einen Code entwickelt, führt man auch Tests durch. Ebenso muss für jedes Produkt oder Problem, das eine Organisation zeigt, dieses durch die Bereitstellung von Annahmen gelöst werden. Das kann mit Hypothesentests erfolgen.

Im Folgenden werden drei Hypothesen vorgestellt. Diese beziehen sich auf die gemessene Ähnlichkeit (in diesem Fall als abhängige Variable bekannt) zwischen dem in Kapitel 4 erhaltenen Datensatz und künstlichen bzw. manipulierten Versionen der Artikelabfragen (unabhängige Variablen) aus Kapitel 6.1.3. Während des Testens der Hypothesen wird die Methode Doc2Vec ausgeschlossen und nicht weiter verwendet, da sie soweit keine plausiblen oder präzisen Ergebnisse geliefert hat, und folglich keine gute Leistung in den vorherigen Experimenten erbracht hat.

Erste Hypothese - Negation

Wenn die zweite Artikelabfrage aus Kapitel 6.1.3 mit der Negation der Verbalphrase durch Einfügen des Kurzforms *n't* negiert wird, dann werden die Kosinuswerte der Experimente in Tabelle 7.2 dadurch erheblich abnehmen, und die WMD-Werte ansteigen.

Diese Hypothese konzentriert sich hauptsächlich auf die Identifizierung der Negation. Im Allgemeinen ist diese keine einfache Aufgabe und ihre Komplexität nimmt zu, da Negationswörter wie *not*, *nor* usw. (syntaktische Negation) nicht das einzige Kriterium zur Negationsberechnung repräsentieren. Die sprachlichen Muster - Präfixe (z. B. *un-*, *dis-* usw.), oder Suffixe (z. B. *-less*) führen auch den Kontext der Negation in Textdaten ein [11]. Allerdings wird sich der Test hier ausschließlich mit dem Umfang der syntaktischen Negation befassen, die sehr häufig im Text verwendet wird und die Bedeutung von Wörtern vollständig ändert. Dies kann daher zu einer Änderung der Versuchsergebnisse führen.

Analyseplan: Der nächste Schritt besteht darin, einen Analyseplan zu formulieren, in dem beschrieben wird, wie die Daten ausgewertet werden. Basierend auf dem in Abbildung 7.3 dargestellten Negationsprozess wird das Folgende vorangestellt: In einem ersten Schritt wird die negative Aussage mit jedem Ergebnisartikel in der Spalte *Ähnlichster Artikel* in Tabelle 7.2 verglichen, und zwar unter Verwendung der entsprechenden Methodenkombination, mit der diese Ergebnisse ursprünglich erzielt wurden. Anschließend wird diese negierte Aussage mit dem gesamten Datensatz verglichen, um mögliche neue Ähnlichkeitsergebnisse zu entdecken.

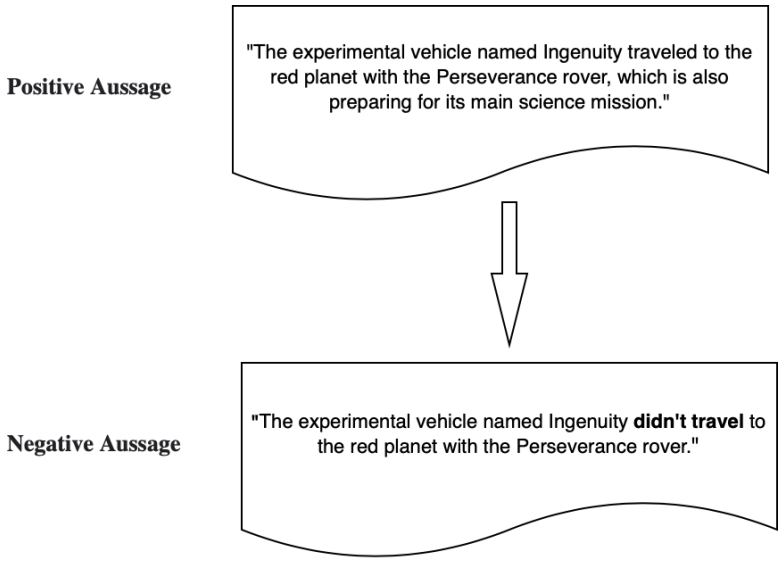


Abbildung 7.3: Syntaktische Negation der zweiten Suchabfrage aus Kapitel 6.1.3.

Erkenntnisse: Anhand der aufgenommenen Werte in Tabelle 7.5 ist es offensichtlich, dass die negative Aussage weniger mit den initialen Ergebnissen übereinstimmt, die ursprünglich unter Verwendung der positiven Aussage als Eingabe erhalten wurden. Eine weitere Erkenntnis ergibt sich aus der Tabelle 7.6. Diese Tabelle zeigt die neuen Ergebnisse, die durch Vergleich der negativen Aussage mit den einzelnen Artikeln in dem Datensatz erhalten wurden. Im Allgemeinen waren alle Methoden nicht in der Lage, ähnliche Artikel aufzunehmen, die die negierte Aussage als Tatsache beweisen können. Dies bestätigt jedoch, dass der Datensatz keine Artikel enthält, die über eine gescheiterte wissenschaftliche Mission auf dem roten Planeten sprechen, was zu den Ausgaben in der Tabelle führte. Somit helfen diese Erkenntnisse, die ursprüngliche Hypothese zu unterstützen. Allerdings implizieren sie keine vollständige Gewissheit.

2. Suchabfrage: *"NASA's Mars Helicopter Prepares for Its First Flight"*		
Experiment	Artikel	Ähnlichkeitswert
TF-IDF + Kosinus-Ähnlichkeit	"The third and final mission to the red planet of the month lifted off on Thursday."	Kosinuswert = 0.15
Word2Vec + Word Mover's Distance	"As part of its next Mars mission, NASA is sending an experimental helicopter to fly through the red planet's thin atmosphere."	WMD-Wert = 2.8
BERT + Kosinus-Ähnlichkeit	"The Chang'e-4 mission, the first to land on the lunar far side, is demonstrating the promise and peril of using ground-penetrating radar in planetary science."	Kosinuswert = 0.54

Tabelle 7.5: Ähnlichkeitswerte der Experimente zwischen der negativen Suchabfrage und den Textelementen in der Spalte *Artikel*.

2. Suchabfrage: *"NASA's Mars Helicopter Prepares for Its First Flight"*		
Experiment	Artikel	Ähnlichkeitswert
TF-IDF + Kosinus-Ähnlichkeit	"The combined orbiter, lander and rover will reach the red planet in February, if all goes well. NASA plans a Mars launch of its own next week."	Kosinuswert = 0.33
Word2Vec + Word Mover's Distance	"As part of its next Mars mission, NASA is sending an experimental helicopter to fly through the red planet's thin atmosphere."	WMD-Wert = 2.8
BERT + Kosinus-Ähnlichkeit	"As part of its next Mars mission, NASA is sending an experimental helicopter to fly through the red planet's thin atmosphere."	Kosinuswert = 0.76

Tabelle 7.6: Ergebnisse der Experimente zwischen der negativen Suchabfrage und den gesamten Artikeln im Datensatz.

Zweite Hypothese - Kontextergänzung

Wenn der Begriff *Fossil Fuel* durch *Coal, Petroleum und Natural Gas* in der ersten Artikelabfrage aus Kapitel 6.1.3 ersetzt wird, ohne den Kontext der originalen Aussage zu ändern, dann werden

sich die Ergebnisartikel bei Word2Vec und BERT von denen in der Tabelle 7.1 nicht ändern. Allerdings wird das Ergebnis bei TF-IDF davon beeinflusst, und sich dadurch ändern.

Hierbei wird auf der Verteilungshypothese aus [22] basiert. Diese besagt, dass Wörter, die in denselben Kontexten vorkommen, tendenziell ähnliche Bedeutungen haben, was die Basis für die semantische Analyse von Text ist. Die Idee dahinter ist, dass man die Bedeutung von Wörtern lernen kann, indem man den Kontext betrachtet, in dem sie erscheinen. Wie zuvor erläutert, wurden im Laufe der Jahre viele Ansätze zum Erlernen von Wörtern aus dem Kontext entwickelt, darunter Word2Vec und BERT. Wenn während der Trainingszeit dieser Methoden zwei Zielwörter einen bestimmten Kontext teilen würden, wäre das Gewicht des Netzwerks für diese beiden Zielwörter intuitiv nahe beieinander und somit ihre übereinstimmenden Vektoren. Dadurch wird eine Verteilungsdarstellung für jedes Wort im Korpus erhalten, im Gegensatz zu zählbasierten Ansätzen (wie BOW und TF-IDF), die die semantische Bedeutung von Wörtern nicht erfassen können.

Analyseplan: Dieser Plan besteht nur aus einem Schritt. Der geänderte bzw. ergänzte Text in Abbildung 7.4 wird über jeden Dateneintrag ausgeführt. Dadurch wird untersucht, ob am Ende des Tests dieselben Ergebnisse wie in Tabelle 7.1 erzielt werden.

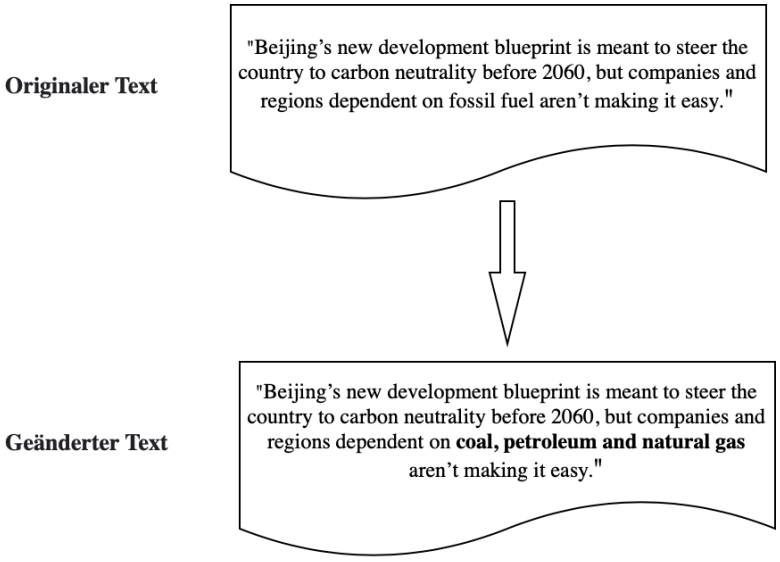


Abbildung 7.4: Kontextergänzung der ersten Suchabfrage aus Kapitel 6.1.3.

Erkenntnisse: Die Tabelle 7.7 unten zeigt, dass die Annahme bzw. zweite Hypothese nur bei BERT richtig wahr. Diese Methode hat trotz der Wortänderung denselben Artikel mit einem gleichen Kosinus-Ähnlichkeitsprozentsatz von **83%** erfasst, wie bei der ersten originalen

Suchabfrage. Allerdings hat die Annahme bei Word2Vec und TF-IDF eindeutig gescheitert. Obwohl die Ausgabe der Methode Word2Vec anders bzw. nicht wie erwartet war, ging sie jedoch nicht aus dem Kontext. D.h. das Hauptthema des erhaltenen Artikels blieb *blieb das Streben nach einer Reduzierung des Einsatzes von Kohle und ihren Derivaten.* Bei TF-IDF hat sich das Ergebnis nicht geändert. Diese Methode wurde daher von dem ergänzten Kontext nicht beeinflusst, und hat sich weiterhin lediglich auf den Begriff *Companies* konzentriert.

1. Suchabfrage: *"China's Climate Ambitions Collide with its Coal Addiction"*		
Experiment	Artikel	Ähnlichkeitswert
TF-IDF + Kosinus-Ähnlichkeit	"Tech companies aren't going to dismantle the systems that are making them billions."	Kosinuswert = 0.21
Word2Vec + Word Mover's Distance	"As coal declines and wind and solar energy rise, some are pushing to limit the use of natural gas, but utilities say they are not ready to do so."	WMD-Wert = 2.7
BERT + Kosinus-Ähnlichkeit	"Efforts to limit global warming often focus on emissions from fossil fuels, but food is crucial, too, according to new research."	Kosinuswert = 0.83

Tabelle 7.7: Ergebnisse der Experimente zwischen der ergänzten Suchabfrage und den gesamten Artikeln im Datensatz.

Dritte Hypothese - Kontextänderung

Wenn der Kontext der dritten Artikelabfrage aus Kapitel 6.1.3 vollständig geändert wird, sodass sie nicht mehr dieselbe Bedeutung besitzt, dann werden sich die zuvor erhaltenen Ausgaben bei Word2Vec und BERT in Tabelle 7.3 ändern. Allerdings wird das Ergebnis bei TF-IDF nicht davon beeinflusst.

Analyseplan: Der Plan folgt dem gleichen Schritt wie bei der zweiten Hypothese und kann daher genauso formuliert werden. Allerdings unter Verwendung des geänderten Textes in Abbildung 7.5 als Eingabe.

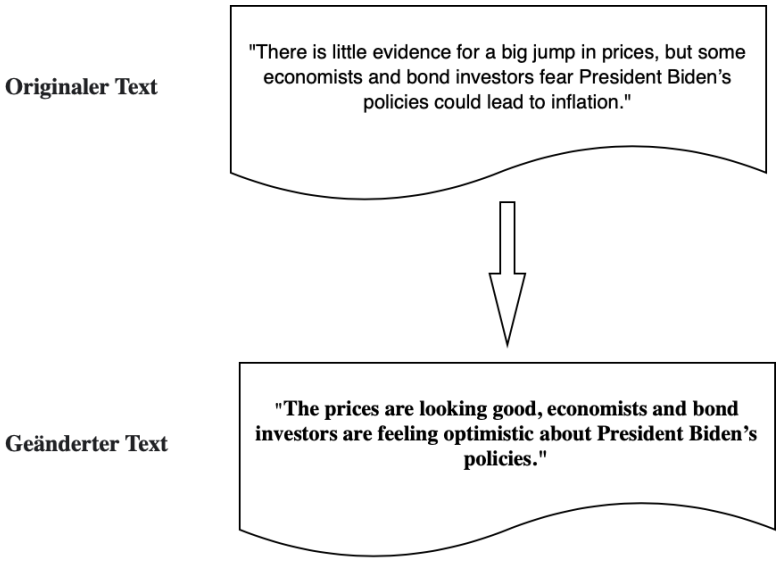


Abbildung 7.5: Kontextänderung der dritten Suchabfrage aus Kapitel 6.1.3.

Erkenntnisse: Durch das Betrachten der Ergebnisse in Tabelle 7.8 und deren Vergleich mit denen in Tabelle 7.3, sieht man keine Änderung in der Ausgabe bei TF-IDF, wie zuvor angenommen wurde. Auf der anderen Seite wurden unterschiedliche Ergebnisse bei Word2Vec und BERT erzielt, was darauf hinweist, dass sich die Änderung des Kontexts tatsächlich die Ähnlichkeitswerte beeinflusst hat. Hierbei haben die neu herausgefundenen Artikel eine leichte Änderung in der Stimmung und wirken nun positiver als die Artikel in der ursprünglichen Tabelle 7.3, die für sie eine negative Bedeutung haben. Diese beiden Erkenntnisse tragen also dazu bei, die getroffene Annahme weiter zu stützen.

3. Suchabfrage: *"Fear of Inflation Finds a Foothold in the Bond Market"*		
Experiment	Artikel	Ähnlichkeitswert
TF-IDF + Kosinus-Ähnlichkeit	"Stocks fell and investors rushed to the safety of government bonds."	Kosinuswert = 0.35
Word2Vec + Word Mover's Distance	"Investors are tripping over one another to give hot start-ups money. DoorDash and Airbnb are going public. The good times are baaack."	WMD-Wert = 2.8
BERT + Kosinus-Ähnlichkeit	"The country's swift and successful response to the pandemic has enabled its real estate market to stay open, encouraging an influx of domestic buyers."	Kosinuswert = 0.77

Tabelle 7.8: Ergebnisse der Experimente zwischen der geänderten Suchabfrage und den gesamten Artikeln im Datensatz.

7.4 Abschließende Erkenntnisse

Im allgemeinen stellt man anhand der oben dargestellten Evaluationen und Hypothesentests fest, dass die Experimente tatsächlich verlässliche Ergebnisse erzielt haben. Allerdings gibt es keine eindeutige Gewinnerkombination. Insgesamt waren BERT, Word2Vec und TF-IDF die leistungsstärksten Methoden. Es ist also überraschend für TF-IDF, da es der zweitälteste erfundene Algorithmus und daher altmodisch im Vergleich zu den anderen Methoden ist. Vielmehr kann man hier enttäuscht sein, dass das hochmoderne KI-Deep-Learning bei dieser Aufgabe keine große Bedeutung hat. Auf der anderen Seite kann jede Deep-Learning-Technik verbessert werden, indem man sein eigenes Modell trainiert und die Daten besser vorverarbeitet, aber es sind alle mit Entwicklungskosten verbunden. Man möchte sich also überlegen, wie besser sich diese Bemühungen im Vergleich zur traditionellen TF-IDF-Methode auswirken werden.

Schließlich ist es fair zu sagen, dass man die Doc2Vec-Methode in Bezug auf die Ähnlichkeit der Dokumente insgesamt vergessen sollte, weil sie keinen Nutzen gegenüber der heutigen Alternative bringt. Entgegen den Erwartungen hat das Hinzufügen von Stimmungsmerkmalen in diesem Fall die Leistung nicht wesentlich erhöht.

8 Zusammenfassung und Ausblick

In diesem abschließenden Kapitel wird eine Zusammenfassung der wichtigsten Erkenntnisse dieser Arbeit gegeben. Darüber hinaus wird ein Ausblick auf absehbare Forschungsmöglichkeiten ermittelt. Dabei werden einige Gedanken und offene Fragen diskutiert, die für zukünftige Entwicklungen untersucht werden können.

8.1 Zusammenfassung

In dieser Forschungsarbeit wurde die Leistung verschiedener Ähnlichkeitsansätze für Textanalyse bewertet. Sie wurden anhand eines Datensatzes verglichen, der aus insgesamt 16,786 Artikeln der New York Times bestand, die im gesamten Jahr 2020 veröffentlicht wurden.

Insgesamt wurden vier verschiedene Kombinationen von Text-Repräsentationsmethoden und Ähnlichkeitsmetriken wie Kosinusähnlichkeit und Word Mover's Distance experimentiert. Diese reichten von den altmodischen und traditionellen Bag-of-Words (mit TF-IDF) bis zu den effizienteren Worteinbettungsmodellen (mit einem tief lernenden neuronalen Netzwerk) wie Word2Vec und Doc2Vec und schließlich dem neuesten Sprachmodell BERT (verwendet beim Transferlernen von *attention*-basierten Transformatoren), das die NLP-Landschaft vollständig revolutioniert hat.

Dabei haben die durchgeführten Experimente einen iterativen Ansatz verfolgt, um ähnliche Texte in vier Hauptschritten zu erkennen: (1) Textbereinigung und Vorverarbeitung durch Konvertierung in Kleinbuchstaben, Stemming und Tokenisierung; (2) Umwandlung von Texten in numerische Ausdrücke bzw. Vektoren unter Verwendung von Repräsentationsmodellen; (3) Sammlung mehrerer Textabfragen als Eingaben; (4) Berechnung der Ähnlichkeitswerten. Die experimentellen Ergebnisse haben dann gezeigt, dass Doc2Vec keine gute Leistung im Vergleich zu den anderen Methoden erbracht hat. Zudem wurde eine Reihe von weiteren Experimenten durchgeführt, die insgesamt drei von der Arbeit aufgestellten Hypothesen getestet haben. Diese haben bewiesen, wie sich die Manipulation verschiedener Merkmale der Texte tatsächlich auf die Methodenergebnisse auswirken kann, was bei der Bewertung der Genauigkeit der erhaltenen Ergebnisse geholfen hat.

Schließlich hat die Arbeit veranschaulicht, dass die präzisesten Ähnlichkeitswerte mit BERT und Word2Vec, gefolgt von TF-IDF, erzielt wurden. Dabei wurde außerdem bestätigt, dass Doc2Vec die Feature-Extraktionsmethode mit der schlechtesten Leistung ist.

8.2 Ausblick

Weitere Forschungsideen und Tests werden in diesem Abschnitt aufgeführt, die aus zeitlichen Gründen nicht untersucht werden konnten. Anschließende Arbeiten schlagen neue Experimente vor. Diese sollen die Auswirkungen verschiedener Vorverarbeitungsideen und Modifikationen der Repräsentationsmodelle auf die Genauigkeit der Ergebnisse untersuchen.

Wie in Kapitel 5.1 angegeben, extrahieren Word2Vec und Doc2Vec semantische Bedeutungen basierend auf Word-Co-Occurrences im Datensatz. D.h. je mehr Word-Co-Occurrences im selben Kontext im gesamten Korpus auftreten, desto bessere Wortdarstellungen werden im Vektorraum erzielt. Allerdings gibt es in der deutschen Sprache viele zusammengesetzte Wörter, zum Beispiel *Diplomsozialpädagogik*, die in *Diplom*, *Sozial* und *Pädagogik* unterteilt werden kann, sodass die im Rahmen dieser Arbeit verwendeten Modelle das zusammengesetzte Wort als eigenständiges Wort in das Vokabular aufnehmen würden.

Zusammengesetzte Wörter sind weniger häufig als ihre einzelnen Unterkomponenten, was schwächere Darstellungen solcher Wörter impliziert. Daher soll der Effekt der Aufteilung dieser Wörter untersucht werden, wodurch der Wortschatz und die Trainingszeit verringert und die Anzahl der Word-Co-Occurrences erhöht wird, um die Unterkomponenten besser darzustellen. Hierzu können Compound-Splitting-Methoden wie SECOS (SEmantic COmpound Splitter) [63, 31] eingesetzt werden.

Daneben ist es auch möglich, mehr als einen Ähnlichkeitsansatz zu kombinieren, um die Gesamtleistung der Methoden zu erhöhen. Dieser Schritt könnte besonders nützlich sein, indem beispielsweise TF-IDF und Doc2Vec kombiniert und ihre Ergebnisse zusammengeführt werden, da Doc2Vec allein nicht zu interessanten Ausgaben geführt hat.

Zudem ist es wichtig zu beachten, dass die erhaltenen Ergebnisse in Kapitel 7.2 und 7.3 in Bezug auf den verwendeten Datensatz interpretiert wurden, weshalb weitere Experimente mit anderen ähnlichen Datensätzen durchgeführt werden sollen, um die Machbarkeit der untersuchten Methoden besser zu beweisen.

Schließlich bleibt das Ziel dieser Arbeit, die Reproduzierbarkeit experimenteller Ergebnisse zu fördern und zuverlässige dauerhafte experimentelle Bedingungen bereitzustellen, die zukünftigen Studien und Entwicklungen im Bereich der Textähnlichkeit zugute kommen sowie die Wiederverwendung bestimmter experimenteller Schritte unterstützen können.

Literaturverzeichnis

[1] Banerjee, S. ; Naskar, S. ; Rosso, P. ; Bandyopadhyay, S. ; Chakma, K. ; Das, A. ; Choudhury, M.: MSIR@FIRE: Overview of the Mixed Script Information Retrieval. In: Working Notes of FIRE 2016 - Forum for Information Retrieval Evaluation. (2016)

[2] Barto, Andrew G. ; Sutton, Richard S.: Reinforcement Learning: An Introduction. In: *MIT Press* (2018)

[3] Bhattacharya, A.: On a Measure of Divergence between Two Multinomial Populations. In: *The Indian Journal of Statistics (1933-1960)* (1946)

[4] Bird, Steven ; Klein, Ewan ; Loper, Edward: *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit.* O'Reilly Media, Inc., 2009

[5] Bird, Steven ; Loper, Edward: *NLTK: the Natural Language Toolkit.* In Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions, 2004. – 31 S

[6] Bonaccorso, Giuseppe: *Machine Learning Algorithms.* Brimingham, England : Packt Publishing Ltd., 2017

[7] Bustos, B. ; Keim, D. ; Saupe, D. ; Schreck, T. ; Vranic, D. V.: Feature-based Similarity Search in 3D Object Databases. In: *ACM Comput. Surv.* 37 (2005), S. 345–387

[8] Bär, Daniel ; Gurevych, Iryna ; Dagan, Ido ; Zesch, Torsten: *A Composite Model for Computing Similarity Between Texts.* Darmstadt, Germany : Technische Universität, 2013

[9] Chase, Zach ; Genain, Nicolas ; Karniol-Tambour, Orren: Learning Multi-Label Topic Classification of News Articles. (2014)

[10] Chen, M.: Efficient Vector Representation for Documents through Corruption. (2017)

[11] Councill, I. G. ; Mcdonald, R. ; Velikovich, L.: *What's Great and What's Not: Learning to Classify the Scope of Negation for Improved Sentiment Analysis.* Uppsala, Sweden : In: The Workshop on Negation and Speculation in Natural Language Processing, 2010. – 51–59 S

[12] Dai, A. M. ; Olah, C. ; Le, Q. V.: *Document Embedding with Paragraph Vectors.* Montreal, Canada : In Proceedings of Deep Learning and Representation Learning Workshop at NIPS, 2014

[13] Devlin, J. ; Chang, M. W. ; Lee, K. ; Toutanova, K. B.: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018)

[14] Dong, Yihong ; Wang, Jiapeng: Measurement of Text Similarity: A Survey. (2020)

[15] Dornel, Benjamin: New York Times Articles & Comments (2020). In: *International Journal on DigitalLibraries* (2020). – URL https://www.kaggle.com/benjaminawd/new-york-times-articles-comments-2020

[16] Eugene, F. K.: Taxicab Geometry. (1987)

[17] Feldman, R. ; Sanger, J.: *The Text Mining Handbook. Advanced Approaches in Analyzing Unstructured Data.* New York, USA : Cambridge University Press, 2007. – 13–19 S

[18] Gabrilovich, E. ; Markovitch, S.: *Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis.* 2007. – 1606–1611 S

[19] Ganesh, H. B. B. ; Kumar, M. A. ; Soman, K. P.: *From Vector Space Models to Vector Space Models of Semantics.* 2018

[20] Gomaa, W. H. ; Fahmy, A. A.: A Survey of Text Similarity Approaches. In: *International Journal of Computer Applications* (2013), S. 13–18

[21] Harris, Charles R. ; Millman, K. J. ; Walt, St'efan J. van der ; Gommers, Ralf ; Virtanen, Pauli ; Cournapeau, David ; Wieser, Eric ; Taylor, Julian ; Berg, Sebastian ; Smith, Nathaniel J. ; Kern, Robert ; Picus, Matti ; Hoyer, Stephan ; Kerkwijk, Marten H. van ; Brett, Matthew ; Haldane, Allan ; R'io, Jaime F. del ; Wiebe, Mark ; Peterson, Pearu ; G'erard-Marchant, Pierre ; Sheppard, Kevin ; Reddy, Tyler ; Weckesser, Warren ; Abbasi, Hameer ; Gohlke, Christoph ; Oliphant, Travis E.: Array programming with NumPy. In: *Nature* 585 (2020), S. 357–362. – URL https://doi.org/10.1038/s41586-020-2649-2

[22] Harris, Zellig S.: *Distributional Structure.* In: arXiv preprint arXiv:1301.3781, 1954. – 146–162 S

[23] Hartmann, Mark: *Machine Learning und IT-Security. Datenschutz Datensich 42.* 2018. – 231–235 S

[24] Hatzivassiloglou, V. ; Klavans, J. ; Eskin, E.: *Detecting Text Similarity over Short Passages: Exploring Linguistic Feature Combinations via Machine Learning. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.* 1999. – 203–212 S

[25] H.Li ; Xu, J.: *Semantic Matching in Search, Foundations and Trends in Information Retrieval.* 2014. – 343–469 S

[26] Huang, A.: *Similarity Measures for Text Document Clustering.* Christchurch, New Zealand : In Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), 2008. – 49–56 S

[27] Indurkhya, Nitin ; Damerau, Fred J.: *Handbook of Natural Language Processing.* CRC Press, 2010

[28] Irving, Robert W. ; Fraser, Campbell B.: *Two Algorithms for the Longest Common Subsequence of Three (or More) Strings.* Tucson, AZ, USA : Conference: Combinatorial Pattern Matching, Third Annual Symposium, 1992. – 214–229 S

[29] Jomsri, P. ; Sanguansintukul, S. ; Choochaiwattana, W.: A Comparison of Search Engine Using "Tag Title and Abstract" with CiteULike—An Initial Evaluation in Internet Technology and Secured Transactions. (2009), S. 1–5

[30] Klenin, Julius ; Botov, Dmitry: Comparison of Vector Space Representations of Documents for the Task of Matching Contents of Educational Course Programmes. (2017)

[31] Koehn, P. ; Knight, K.: *Empirical Methods for Compound Splitting.* In: Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 1, 2003. – 187–193 S

[32] Kondrak, G.: *N-gram Similarity and Distance: In International Symposium on String Processing and Information Retrieval.* 2015. – 115–126 S

[33] Krause, E. F.: *Taxicab Geometry: An Adventure in non-Euclidean Geometry.* Courier Corporation, 1975

[34] Kusner, Matt ; Sun, Yu ; Kolkin, Nicholas ; Weinberger, Kilian: *From Word Embeddings to Document Distances.* In International Conference on Machine Learning, 2015. – 957–966 S

[35] Landauer, T. K. ; Dumais, S. T.: *A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge.* 1997. – 211–240 S

[36] Le, Q. ; Mikolov, T.: *Distributed Representations of Sentences and Documents.* In Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014. – 1188–1196 S

[37] Leacock, C. ; Chodorow, M.: *WordNet: An Electronic Lexical Database. Chapter Combining Local Context and WordNet Similarity for Word Sense Identification.* MIT Press, 1998. – 265–283 S

[38] Lee, Sungjick ; Kim, Hanjoon: *News Keyword Extraction for Topic Tracking*. Bd. 2. In: Networked Computing and Advanced Information Management, 2008. – 554–559 S

[39] Levenshtein, V. I.: Binary Codes Capable of Correcting Spurious Insertions and Deletions of Ones. In: *Problems of Information Transmission* (1965), S. 8–17

[40] Li, Xue ; Wang, Shuliang ; Dong, Zhao Y.: *Advanced Data Mining and Applications.* Wuhan, China : Springer, 2005

[41] Li, Y. ; McLean, D. ; Bandar, Z. A. ; O'Shea, J. D. ; Crockett, K.: Sentence Similarity Based on Semantic Nets and Corpus Statistics. In: *IEEE Transactions on Knowledge and Data Engineering* 18 (2006), S. 1138–1150

[42] Liddy, Elizabeth D.: Natural Language Processing. In: *Syracuse University* (2001)

[43] Lin, D.: *Extracting Collocations from Text Corpora.* Montreal, Canada : In Workshop on Computational Terminology, 1995. – 57–63 S

[44] Ling, Haibin ; Okada, Kazunori: *An Efficient Earth Mover's Distance Algorithm for Robust Histogram Comparison.* Pattern Analysis and Machine Intelligence, 2007

[45] Lund, K. ; Burgess, C.: *Producing High-Dimensional Semantic Spaces from Lexical Co-occurrence.* Behavior Research Methods, Instruments & Computers, 1996. – 203–208 S

[46] Lund, K. ; Burgess, C. ; Atchley, R. A.: *Semantic and Associative Priming in a High-Dimensional Semantic Space.* Cognitive Science Proceedings (LEA), 1995. – 660–665 S

[47] Manning, Christopher D. ; Hirschberg, Julia: *Advances in Natural Language Processing.* American Association for the Advancement of Science, 2015

[48] McKinney, Wes: *Data Structures for Statistical Computing in Python.* In Stefan van der Walt and Jarrod Millman, editors, Proceedings of the 9th Python in Science Conference, 2010. – 51–56 S

[49] Meadow, C. ; Boyce, B. ; Kraft, D.: *Text Information Retrieval Systems.* second ed. Academic Press, 2000

[50] Mihalcea, Rada ; Corley, Courtney ; Strapparava, Carlo: *Corpus-based and Knowledge-based Measures of Text Semantic Similarity.* Boston, USA : Proceedings of the 21st National Conference on Artificial Intelligence, 2006

[51] Mikolov, Tomas ; Chen, Kai ; Corrado, Greg ; Dean, Jeffrey: *Efficient Estimation of Word Representations in Vector Space.* In: arXiv preprint arXiv:1301.3781, 2013

[52] Miller, George A. ; Beckwith, Richard ; Fellbaum, Christiane ; Gross, Derek ; Miller, Katherine: *Introduction to WordNet: An On-line Lexical Database.* Princeton, NJ, USA : Princeton University, 1993. – 235–244 S

[53] Nemhauser, G. L. ; Wolsey, L. A.: *Integer and Combinatorial Optimization.* Bd. 18. Wiley New York, 1988

[54] Online: GoogleNews Dataset. (2020). – URL https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[55] Online: HuggingFace's Transformers: State-of-the-Art Natural Language Processing. (2020). – URL https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens

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

[57] Pele, O. ; Werman, M.: *Fast and Robust Earth Mover's Distances.* In ICCV, 2009. – 460–467 S

[58] Pennington, Jeffrey ; Socher, Richard ; Manning, Christopher: *Glove: Global Vectors for Word Representation.* In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. – 1532–1543 S

[59] Peters, Jan ; Bagnell, J. A. ; Kober, Jens: Reinforcement Learning in Robotics: A Survey, Bielefeld: CoR-Lab Research Institute for Cognition and Robotics. (2013)

[60] Pradhan, Nitesh ; Gyanchandani, Manasi ; Wadhvani, Rajesh: A Review on Text Similarity Technique used in IR and its Application. (2015)

[61] Rensch, Calvin R.: Calculating lexical similarity. In: *Summer Institute of Linguistics and the University of Texas at Arlington* (1992)

[62] Resnik, R.: *Using Information Content to Evaluate Semantic Similarity.* Montreal, Canada : In Proceedings of the 14th International Joint Conference on Artificial Intelligence, 1995

[63] Riedl, M. ; Biemann, C.: *Unsupervised Compound Splitting with Distributional Semantics Rivals Supervised Methods.* In: HLT-NAACL, 2016. – 617–622 S

[64] Rozeva, Anna ; Zerkova, Silvia: Assessing Semantic Similarity of Texts - Methods and Algorithms. In: *AIP Conference Proceedings* (2017)

[65] Rubner, Y. ; Tomasi, C. ; Guibas, L. J.: *A Metric for Distributions with Applications to Image Databases.* In ICCV, 1998. – 59–66 S

[66] Salton, G. ; Lesk, M.: Computer Evaluation of Indexing and Text Processing. In: *Prentice Hall, Inc. Englewood Cliffs, NJ* (1971)

[67] Schnabel, Tobias: *Evaluation Methods for Unsupervised Word Embeddings.* In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015. – 298–307 S

[68] Sengupta, S. ; Wang, H. ; Blackburn, W. ; Ojha, P.: *Feature Based Similarity Measure.* 2014

[69] Singh, V. ; Kumar, B. ; Patnaik, T.: *Feature Extraction Techniques for Handwritten Text in Various Scripts: a Survey.* Bd. 3. International Journal of Soft Computing & Engineering, 2013. – 238–241 S

[70] Sugiyama, K. ; Kan, M. Y.: *A Comprehensive Evaluation of Scholarly Paper Recommendation Using Potential Citation Papers.* Bd. 16. International Journal on DigitalLibraries, 2015. – 91–109 S

[71] Taylor, Wilson L.: *Cloze procedure: A New Tool for Measuring Readability.* Journalism Bulletin, 2008. – 415–433 S

[72] Tian, Y. ; Zhang, J.: *Improvement of Linked Data Fusion Algorithm Based on Bag of Words.* Bd. 35. Library Journal, 2016. – 17–22 S

[73] Turney, P.: Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In Proceedings of the 12th. European Conference on Machine Learning. (2001)

[74] Vijaymeena1, M. K. ; Kavitha, K.: A Survey on Similarity Measures in Text Mining. In: *Machine Learning and Applications: An International Journal (MLAIJ)* (2016), S. 19–28

[75] Vrbanec, Tedo ; Meštrović, Ana: Corpus-Based Paraphrase Detection Experiments and Review. In: *Information* 11 (2020)

[76] Wan, X.: *A Novel Document Similarity Measure based on Earth Movers Distance.* Information Sciences, 2007. – 3718–3730 S

[77] Wang, Qiang ; Li, Bei ; Xiao, Tong ; Zhu, Jingbo ; Li, Changliang ; Wong, Derek F. ; Chao, Lidia S.: Learning Deep Transformer Models for Machine Translation. (2019)

[78] Wennker, Phil: *Künstliche Intelligenz in der Praxis. Anwendung in Unternehmen und Branchen: KI wettbewerbs- und zukunftsorientiert einsetzen.* Wiesbaden, Germany : Springer Gabler, 2020

[79] Wu, Yonghui ; Schuster, Mike ; Chen, Zhifeng ; Le, Quoc V. ; Norouzi, Mohammad ; Macherey, Wolfgang ; Krikun, Maxim ; Cao, Yuan ; Gao, Qin ; Macherey, Klaus: *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* 2016

[80] Wu, Z. ; Palmer, M.: Verb semantics and Lexical Selection. In Proceedings of the Annual Meeting Association for Computational Linguistics. (1994)

[81] Yu, M. ; Li, G. ; Deng, D. ; Feng, J.: *String Similarity Search and Join: A Survey.* 2016. – 399–417 S

[82] Zhang, W. ; Yoshida, T. ; Tang, X.: *A Comparative Study of TF*IDF, LSI and Multi-Words for Text Classification.* Bd. 38. Expert Systems with Applications, 2010. – 2758–2765 S

[83] Zhu, Yukun ; Kiros, Ryan ; Zemel, Rich ; Salakhutdinov, Ruslan ; Urtasun, Raquel ; Torralba, Antonio ; Fidler, Sanja: *Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books.* In Proceedings of the IEEE International Conference on Computer Vision, 2015. – 19–27 S

[84] Řehůřek, Radim ; Sojka, Petr: *Software Framework for Topic Modelling with Large Corpora.* Valletta, Malta : In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, 2010. – 45–50 S

Quellcode

1 tf-idf+cosine-similarity.py

```
import nltk
import string
import numpy as np
# import pandas as pd
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel

nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')

new_document = "This is an example sentence for the document to be compared"
documents = ["This is the documents collection to be compared against the new_document"]

# ____________________________________________________________________________________________________


# The tokenize function preprocesses the dataset before it gets vectorized
def tokenize(text):
    stem = nltk.stem.SnowballStemmer('english')
    stem = nltk.stem.PorterStemmer()
    lemmatizer = WordNetLemmatizer()
    text = text.lower()

    for token in nltk.word_tokenize(text):
        if token in string.punctuation:
            continue
        yield lemmatizer.lemmatize(token)
        yield stem.stem(token)

# ____________________________________________________________________________________________________

```

```
34
35  def process_tfidf_similarity():
36      vectorizer = TfidfVectorizer(tokenizer=tokenize, stop_words='english')
37
38      # To make uniformed vectors, new_document needs to be combined with the documents collection first.
39      documents.insert(0, new_document)
40      embeddings = vectorizer.fit_transform(documents)
41
42      cosine_similarities = linear_kernel(embeddings[0:1], embeddings).flatten()
43      cosine_sim = np.delete(cosine_similarities, 0)
44
45      # The below iteration finds the highest similarity value in the object cosine_sim and its corresponding document
46      # index
47      highest_score = 0
48      highest_score_index = 0
49      for i, score in enumerate(cosine_sim):
50          if highest_score < score:
51              highest_score = score
52              highest_score_index = i
53
54      most_similar_document = documents[highest_score_index + 1]
55
56      print("Most similar document with the score [", highest_score, "] is: " + most_similar_document)
57
58  # ____________________________________________________________________________________________________
59
60
61  process_tfidf_similarity()
```

2 word2vec+wmd.py

```
1   import gensim
2   import nltk
3   import string
4   # import pandas as pd
5   from nltk.tokenize import word_tokenize
6   from nltk.corpus import stopwords
7   from nltk.stem import WordNetLemmatizer
8
9   nltk.download('stopwords')
10
11  # Loading the pre-trained embeddings. Each word is represented as a 300 dimensional vector
12  model = gensim.models.KeyedVectors.load_word2vec_format(
13      'GoogleNews-vectors-negative300.bin.gz', binary=True)
14
15  new_document = "This is an example sentence for the document to be compared"
16  documents = [
17      "This is the documents collection to be compared against the new_document"]
18
19  # ____________________________________________________________________________________________
20
21
22  # The preprocess function cleans the dataset before it gets tokenized
23  def preprocess(text):
24      lowered = str.lower(text)
25      lemmatizer = WordNetLemmatizer()
26
27      stop_words = set(stopwords.words('english'))
28      word_tokens = word_tokenize(lowered)
29
30      words = []
31      for w in word_tokens:
32          if w not in stop_words:
33              if w not in string.punctuation:
34                  if len(w) > 1:
35                      lemmatized = lemmatizer.lemmatize(w)
36                      words.append(lemmatized)
37
38      return words
39
```

```
40
41  # The dataset is then tokenized
42  tokenized_doc = []
43  for d in documents:
44      tokenized_doc.append(preprocess(d))
45
46  # ____________________________________________________________________________________________
47
48
49  # The word mover's distances are calculated between the new_document and each one of the documents in the
50  # collection
51  distances = []
52  for d in tokenized_doc:
53      distance = model.wmdistance(preprocess(new_document), d)
54      distances.append(distance)
55
56  # ____________________________________________________________________________________________
57
58  # The distance between two text documents A and B is the minimum cumulative distance that words from
59  # the text document A need to travel to match exactly the point cloud of text document B
60  most_similar_document = documents[distances.index(min(distances))]
61  print("Most similar document with the score [", min(
62      distances), "] is: " + most_similar_document)
```

3 doc2vec+cosine-similarity.py

```
1   import numpy as np
2   import pandas as pd
3   import nltk
4   import string
5   from gensim.models.doc2vec import Doc2Vec, TaggedDocument
6   from nltk import word_tokenize
7   from nltk.stem import WordNetLemmatizer
8   from nltk.corpus import stopwords
9   from sklearn.metrics.pairwise import cosine_similarity
10
11  nltk.download('stopwords')
12  nltk.download('wordnet')
13  nltk.download('punkt')
14
15  new_document = "This is an example sentence for the document to be compared"
16  documents = ["This is the documents collection to be compared against the new_document"]
17
18  # First new_document needs to be combined with the documents collection.
19  documents = np.insert(documents, 0, new_document)
20
21  # ____________________________________________________________________________________________
22
23  # The preprocess function cleans the dataset before it gets vectorized
24  def preprocess(text):
25      lowered = str.lower(text)
26      lemmatizer = WordNetLemmatizer()
27
28      stop_words = set(stopwords.words('english'))
29      word_tokens = word_tokenize(lowered)
30
31      words = []
32      for w in word_tokens:
33          if w not in stop_words:
34              if w not in string.punctuation:
35                  if len(w) > 1:
36                      lemmatized = lemmatizer.lemmatize(w)
37                      words.append(lemmatized)
38
39      return words
40
41  # ____________________________________________________________________________________________
```

```
42
43  # Before training the doc2Vec model, the cleaned data is tagged first.
44  tagged_data = [TaggedDocument(words=preprocess(doc), tags=[i]) for i, doc in enumerate(documents)]
45
46  # The doc2Vec model is tested with 300 dimensions over a number of 100 iterations using the 'distributed memory'
47  # (PV-DM) training algorithm
48  model_d2v = Doc2Vec(vector_size=300, alpha=0.025, min_alpha=0.00025, min_count=1, dm=1)
49  model_d2v.build_vocab(tagged_data)
50
51  for epoch in range(100):
52      print('iteration {0}'.format(epoch))
53      model_d2v.train(tagged_data,
54                      total_examples=model_d2v.corpus_count,
55                      epochs=model_d2v.epochs)
56
57  # document-word embeddings are created based on the doc2Vec model results
58  document_embeddings = np.zeros((documents.shape[0], 300))
59
60  for i in range(len(document_embeddings)):
61      document_embeddings[i] = model_d2v.docvecs[i]
62
63  # ____________________________________________________________________________________________
64
65  documents = pd.DataFrame(documents, columns=['documents'])
66
67  # The cosine similarity is calculated between all the document embeddings
68  pairwise_similarities = cosine_similarity(document_embeddings)
69
70
71  # The most_similar function returns the cosine similarity between a given document index and each one of the
72  # documents in the collection
73  def most_similar(doc_id, similarity_matrix, matrix):
74      print(f'Document: {documents.iloc[doc_id]["documents"]}')
75      print('\n')
76      print('Similar Documents:')
77      if matrix == 'Cosine Similarity':
78          similar_ix = np.argsort(similarity_matrix[doc_id])[::-1]
79      for ix in similar_ix:
80          if ix == doc_id:
81              continue
82          print('\n')
83          print(f'Document: {documents.iloc[ix]["documents"]}')
84          print(f'{matrix} : {similarity_matrix[doc_id][ix]}')
85
86
87  # ____________________________________________________________________________________________
88
89  most_similar(0, pairwise_similarities, 'Cosine Similarity')
```

4 bert+cosine-similarity.py

```python
import numpy as np
# import pandas as pd
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
from nltk import sent_tokenize

new_document = "This is an example sentence for the document to be compared"
documents = ["This is the documents collection to be compared against the new_document"]

# ____________________________________________________________________________________________


def process_bert_similarity():
    # This will download and load the pretrained model offered by UKPLab.
    model = SentenceTransformer('bert-base-nli-mean-tokens')

    # Although it is not explicitly stated in the official document of sentence transformer
    # the original BERT is meant for shorter sentences.
    # Therefore the model is fed by sentences instead of the whole documents.
    sentences = sent_tokenize(new_document)
    base_embeddings_sentences = model.encode(sentences)
    base_embeddings = np.mean(np.array(base_embeddings_sentences), axis=0)

    vectors = []
    for i, document in enumerate(documents):
        sentences = sent_tokenize(document)
        embeddings_sentences = model.encode(sentences)
        embeddings = np.mean(np.array(embeddings_sentences), axis=0)

        vectors.append(embeddings)

        print("making vector at index:", i)

# ____________________________________________________________________________________________

    # The cosine similarity values are calculated between the new_document embeddings and each of the vectorized
    # documents in the collection
    scores = cosine_similarity([base_embeddings], vectors).flatten()

    highest_score = 0
    highest_score_index = 0
    for i, score in enumerate(scores):
        if highest_score < score:
            highest_score = score
            highest_score_index = i

    most_similar_document = documents[highest_score_index]
    print("Most similar document by BERT with the score:", most_similar_document, highest_score)

# ____________________________________________________________________________________________


process_bert_similarity()
```

Erklärung zur selbstständigen Bearbeitung einer Abschlussarbeit

Gemäß der Allgemeinen Prüfungs- und Studienordnung ist zusammen mit der Abschlussarbeit eine schriftliche Erklärung abzugeben, in der der Studierende bestätigt, dass die Abschlussarbeit „— bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit [(§ 18 Abs. 1 APSO-TI-BM bzw. § 21 Abs. 1 APSO-INGI)] — ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Quelle: § 16 Abs. 5 APSO-TI-BM bzw. § 15 Abs. 6 APSO-INGI

Erklärung zur selbstständigen Bearbeitung der Arbeit

Hiermit versichere ich,

Name: _____

Vorname: _____

dass ich die vorliegende Bachelorarbeit – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema:

Textähnlichkeit unter Verwendung von Ansätzen des maschinellen Lernens

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

_____	_____	_____
Ort	Datum	Unterschrift im Original