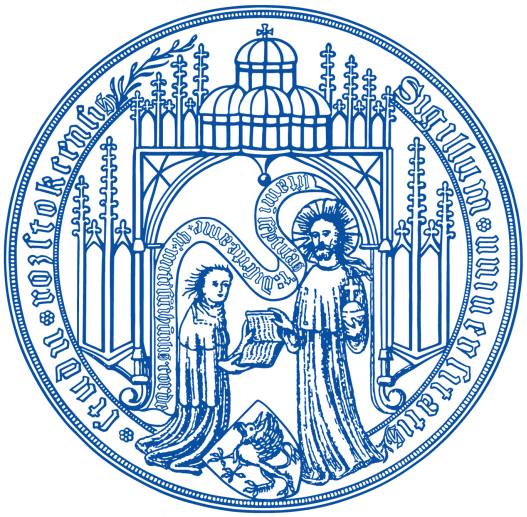
k-Means-Clustering von Textdokumenten mit teilweise fixierten Clustermittelpunkten

Bachelorarbeit

Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Informatik



Vorgelegt von: Abdullatif Akrin
Matrikelnummer: 218202909
geboren am: 09.01.1996 in Aleppo
Erstgutachter: Dr.-Ing. M.Sc. Hannes Grunert
Zweitgutachter: Dr.-Ing. Holger Meyer
Abgabedatum: 02. Oktober 2023

Zusammenfassung

In dieser Arbeit wird einen neuen Ansatz zur Textklassifizierung und -gruppierung mithilfe von Techniken aus den Bereichen maschinelles Lernen und natürliche Sprachverarbeitung (Natural Language Processing, NLP) präsentiert. Die Arbeitsschritte der Textanalyse, von der Vorverarbeitung bis zur Merkmalsextraktion, wurden mit Schwerpunkt auf dem Einsatz des FCK-Means Clustering-Algorithmus detailliert beschrieben. Dieser Algorithmus ermöglicht die Gruppierung ähnlicher Textdokumente auf Basis hochdimensionaler, semantischer Darstellungen, die durch den Einsatz von TF-IDF und BERT-Einbettungen gewonnen werden. In der vorliegenden Arbeit wurde ein System entwickelt und implementiert, das auf Python und mehreren maschinellen Lern- und NLP-Bibliotheken, einschließlich Scikit-Learn, NLTK und Transformers, basiert. Der entwickelte Ansatz wurde auf verschiedenen Anwendungsfällen getestet. Weiterhin wurden praktische Herausforderungen, die bei der Implementierung des Systems auftraten, wie das Management großer Datenmengen und die Sicherstellung der Qualität der Eingabedaten, gründlich diskutiert und Lösungsansätze vorgeschlagen. Zusammenfassend bietet die Arbeit einen tiefgreifenden Einblick in die Methodik und Implementierung eines auf NLP und Clustering basierenden Textklassifikationssystems und unterstreicht dessen Potential in diversen Anwendungsgebieten.

Abstract

In this paper, a new approach to text classification and clustering using machine learning and natural language processing (NLP) techniques is presented. The steps of text analysis, from pre-processing to feature extraction, have been described in detail with emphasis on the use of the FCK-Means clustering algorithm. This algorithm enables the grouping of similar text documents based on high-dimensional semantic representations obtained by using TF-IDF and BERT embeddings. In the present work, a system has been developed and implemented based on Python and several machine learning and NLP libraries, including Scikit-Learn, NLTK and Transformers. The developed approach was tested on several use cases, including recommender systems, information management, knowledge organization, and content aggregation. Furthermore, practical challenges encountered during the implementation of the system, such as managing large data sets and ensuring the quality of input data, were thoroughly discussed and solutions were proposed. In summary, the work provides an in-depth insight into the methodology and implementation of a text classification system based on NLP and clustering, and highlights its potential in diverse application domains.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die exponentielle Zunahme digitaler Informationen in unserer modernen Welt hat zu einer enormen Menge an verfügbaren Textdokumenten geführt. Von sozialen Medien über wissenschaftliche Arbeiten bis hin zu Geschäftsberichten sind Textdokumente ein zentrales Medium für den Austausch von Wissen, Kommunikation und Informationsübertragung. Diese große Menge an Textdaten bietet Möglichkeiten, wertvolle Erkenntnisse und Muster zu gewinnen, stellt aber auch eine Herausforderung dar, da eine strukturierte Organisation und Kategorisierung dieser Daten erforderlich ist [IEA+23]. Eine der bekanntesten und meistverwendeten Clustering-Methoden ist das K-Means-Clustering, das sich als hilfreiches Werkzeug zur Gruppierung ähnlicher Datenpunkte in Cluster herausgestellt hat [IEA+23]. Die Anwendung des K-Means-Algorithmus auf Textdaten ermöglicht es, semantische Zusammenhänge und Themen in den Dokumenten zu identifizieren, die mit herkömmlichen Methoden oft verborgen bleiben [LL19]. Diese Bachelorarbeit untersucht eine spezifische Variation des K-Means-Clustering-Ansatzes, bei dem teilweise fixierte Clustermittelpunkte verwendet werden. Dieser Ansatz erweitert die Vorteile des traditionellen K-Means-Algorithmus durch eine präzisere Steuerung der Clusterbildung. Durch die teilweise Fixierung von Clustermittelpunkten wird die Clusterbildung flexibler gestaltet und es können bestimmte Merkmale oder Kriterien stärker betont werden. Dieser Ansatz ermöglicht eine eingehendere Betrachtung der strukturellen Eigenschaften von Textdokumenten, was zu einer verbesserten Informationsgewinnung führen kann. In der vorliegenden Arbeit werden die theoretischen Grundlagen des K-Means-Clustering und seiner Varianten, insbesondere des teilweise fixierten Ansatzes, detailliert untersucht. Die praktische Umsetzung erfolgt durch die Anwendung eines entwickelten Algorithmus auf eine repräsentative Auswahl von Textdokumenten. Die Ergebnisse werden hinsichtlich der erreichten Clusterqualität, der Interpretierbarkeit der Cluster und der Effizienz des Algorithmus bewertet. Die vorliegende Bachelorarbeit hat das Ziel, nicht nur das Konzept des K-Means-Clustering und seiner Erweiterungen zu erläutern, sondern auch die potenziellen Vorteile der teilweise fixierten Clustermittelpunkte für die Strukturierung und Analyse von Textdaten zu beleuchten. Durch die Anwendung und Evaluation dieses Ansatzes tragen die Ergebnisse zu einem tieferen Verständnis der Clusteranalyse von Textdokumenten bei und liefern wertvolle Einblicke in dessen Anwendungsgebiete und Grenzen.

1.1 Hintergrund und Motivation

Die Menge an digitalen Textdaten wächst exponentiell, was zu einer reichhaltigen Quelle für Informationen, Erkenntnisse und Wissen führt. Die Analyse und Kategorisierung dieser Textdokumente sind jedoch komplexe Aufgaben, da sie häufig unstrukturiert sind und eine Vielzahl von Themen und Kontexten abdecken. In diesem Kontext gewinnt das Clustering von Textdokumenten zunehmend an Bedeutung, um verborgene Muster und Zusammenhänge zu entdecken, die menschlichen Analysten möglicherweise entgehen [KMB+17]. Das K-Means-Clustering hat sich als eine effiziente Methode zur Gruppierung von Datenpunkten in homogene Cluster erwiesen. Es ist jedoch bekannt, dass der klassische K-Means-Algorithmus anfällig für die Initialisierung der Clustermittelpunkte ist und zu unterschiedlichen Ergebnissen führen kann, abhängig von der Anfangskonfiguration. Dies hat zu Forschung und Entwicklung von Modifikationen des K-Means-Verfahrens geführt, um diese Schwächen zu überwinden und seine Anwendbarkeit auf verschiedene Datentypen zu verbessern [Lay23]. In dieser Bachelorarbeit wird auf eine spezielle Modifikation des K-Means-Clustering-Verfahrens konzentriert, bei dem teilweise fixierte Clustermittelpunkte verwendet werden. Diese Erweiterung ermöglicht es, gezielt bestimmte Eigenschaften oder Merkmale in der Clusterbildung zu berücksichtigen. Die Idee dahinter ist, die Flexibilität des K-Means-Algorithmus zu erhöhen, indem man einige Clustermittelpunkte fixiert, während andere frei variieren. Dieser Ansatz bietet die Möglichkeit, eine präzisere und maßgeschneiderte Clusterbildung zu erreichen, was insbesondere in Anwendungsfällen mit spezifischen Anforderungen von Vorteil sein kann. Die Motivation hinter dieser Arbeit liegt in der Notwendigkeit, effektive Ansätze für das Clustering von Textdokumenten zu entwickeln. Die Anwendung des K-Means-Verfahrens mit teilweise fixierten Clustermittelpunkten eröffnet neue Möglichkeiten, die Struktur von Textdaten besser zu erfassen und interpretierbare Cluster zu erzeugen. Dies kann in verschiedenen Bereichen von großem Nutzen sein, von der Informationsextraktion über die Themenmodellierung bis hin zur automatischen Kategorisierung von Textinhalten. Die vorliegende Arbeit zielt darauf ab, nicht nur das Konzept und die Implementierung des K-Means-Clustering mit teilweise fixierten Clustermittelpunkten zu präsentieren, sondern auch die Wirksamkeit dieses Ansatzes auf realen Textdaten zu demonstrieren. Indem sie eine Brücke zwischen theoretischen Konzepten und praktischer Anwendbarkeit schlägt, trägt diese Arbeit dazu bei, das Potenzial dieser Methode für die Analyse und Strukturierung von Textdokumenten auszuloten.

1.2 Problemstellung und Zielsetzung

Die Analyse und Strukturierung von Textdokumenten stehen vor der Herausforderung, unstrukturierte Daten in sinnvolle Gruppen zu organisieren. Der klassische K-Means-Algorithmus, der zur Clusterbildung verwendet wird, kann je nach Anfangskonfiguration der Clustermittelpunkte zu unterschiedlichen Ergebnissen führen. Diese Sensibilität gegenüber der Initialisierung beeinträchtigt die Stabilität und Qualität der Clusterbildung.

Das Ziel dieser Bachelorarbeit ist es, das K-Means-Clustering von Textdokumenten mit teilweise fixierten Clustermittelpunkten zu erforschen. Dabei soll ein Ansatz entwickelt werden, der es ermöglicht, bestimmte Clustermittelpunkte teilweise zu fixieren, um die Clusterbildung zu steuern. Die Arbeit zielt darauf ab, die Vorteile dieses Ansatzes hinsichtlich verbesserter Clusterqualität und Interpretierbarkeit zu untersuchen. Die praktische Implementierung und Evaluation des Ansatzes erfolgt anhand von realen Textdaten, um seine Wirksamkeit in der Praxis zu demonstrieren.

1.3 Aufbau der Arbeit

Die vorliegende Arbeit ist in sechs Hauptkapitel unterteilt. Im zweiten Kapitel 2 werden die theoretischen Fundamente von Natural Language Processing (NLP), Information Retrieval (IR) sowie des K-Means-Verfahrens und dessen erweiterten Anwendungen, insbesondere des FCK-Means-Verfahrens, ausführlich erörtert. Kapitel 3 dient der umfassenden Beschreibung des grundlegenden Konzepts, welches in Kapitel 4 in die Tat umgesetzt wird. Eine umfangreiche Evaluation dieser Implementierung wird schließlich in Kapitel 5 präsentiert.

2 Theoretische Grundlagen

In diesem Kapitel werden die Grundlagen der natürlichen Sprachverarbeitung (NLP) und deren Anwendungen dargelegt. Der Schwerpunkt liegt auf Textmining-Verfahren, insbesondere der Textvorverarbeitung. Ebenso wird eine eingehende Untersuchung des Information Retrieval (IR) durchgeführt, wobei besonderes Augenmerk auf die Textkodierung gelegt wird, welche eine zentrale Rolle bei der effizienten Informationsgewinnung spielt. Darüber hinaus werden Clustering-Methoden sowie deren Anwendungen im Bereich des Text Minings behandelt. Dieses Kapitel bildet die Grundlage für das Verständnis und die praktische Anwendung von NLP- und Textmining-Techniken im Rahmen dieser Arbeit.

2.1 Natural Language Processing

Natural Language Processing (NLP) ist ein multidisziplinäres Forschungsgebiet, das sich mit der Verarbeitung und Analyse von menschlicher Sprache durch Computer befasst.
Es kombiniert Konzepte aus der Semantik, Softwareentwicklung und künstlichen Intelligenz, um Computer in die Lage zu versetzen, natürliche Sprache zu verstehen, zu interpretieren und zu verarbeiten. Das Hauptziel von NLP besteht darin, Computer so zu programmieren, dass sie geschriebene Texte in ihrer natürlichen Form verstehen können, einschließlich des Kontexts und der Bedeutung der verwendeten Sprache. Durch den Einsatz von statistischen Modellen, maschinellem Lernen und sprachwissenschaftlicher Analyse werden verschiedene Techniken und Algorithmen entwickelt, um Sprachdaten zu analysieren, Informationen zu extrahieren und Aufgaben wie maschinelle Übersetzung, Informationsextraktion, Sentiment-Analyse und Textgenerierung durchzuführen [KS21].

NLP spielt aufgrund seiner wissenschaftlichen, wirtschaftlichen, sozialen und kulturellen Relevanz eine bedeutende Rolle. Es verzeichnet eine rasante Entwicklung, da seine Theorien und Methoden in verschiedenen neuen sprachbezogenen Technologien angewendet werden. Aus diesem Grund ist es von großer Bedeutung, dass eine breite Palette von Menschen über Kenntnisse im Bereich NLP verfügt. In der Industrie betrifft dies insbesondere Personen, die in den Bereichen Mensch-Computer-Interaktion, Geschäftsinformationen und Web-Software-Entwicklung tätig sind. In der akademischen Welt erstreckt sich dies von den Geisteswissenschaften, der Informatik und der Korpuslinguistik bis hin zu den Computerwissenschaften und der künstlichen Intelligenz. Innerhalb der akademischen Gemeinschaft wird NLP oft unter dem Begriff “Computational Linguistics. “diskutiert und erforscht [BKL09].

2.1.1 Anwendungen von NLP

Die Anwendungsbereiche von Natural Language Processing (NLP) umfassen eine Vielzahl von Anwendungen, die darauf abzielen, die menschliche Sprache zu verstehen, zu analysieren und mit ihr zu interagieren. Es existieren verschiedene Beispiele für NLP-Anwendungen, von denen einige im Folgenden aufgeführt werden:

- **Informationsextraktion:** Die Informationsextraktion (IE) gehört zu den zentralen Fachgebieten der Verarbeitung natürlicher Sprache, da sie die geordnete Extraktion relevanter Informationen aus Texten mit unwichtigen oder unbrauchbaren Wörtern, auch als „Rauschen“ bezeichnet, ermöglicht. Dieser Prozess trägt dazu bei, das Verständnis von Maschinen für die menschliche Sprache zu verbessern und spielt eine entscheidende Rolle bei der Extraktion und strukturierten Speicherung von anwendbarem Wissen [LS23]. Die Extraktion von Textinhalten aus Zeitschriftenartikeln spielt eine entscheidende Rolle im Analyseprozess. Typischerweise werden Zeitschriftenartikel im PDF-Format vorliegen, welches halbstrukturierte Daten enthält. Diese Artikel sind in verschiedene Abschnitte, wie beispielsweise Einführung, Methodik, Versuchsaufbau, Ergebnisse und Analyse, unterteilt. Diese Unterteilung ermöglicht es dem Leser, gezielt auf die relevanten Informationen zuzugreifen, abhängig von den individuellen Interessen. Die Hauptaufgabe der Abschnittsextraktion besteht darin, eine repräsentative Teilmenge der Daten zu finden, die alle relevanten Informationen des gesamten Artikels enthält. Es existieren verschiedene Ansätze zur Extraktion von Abschnitten aus Forschungsarbeiten, darunter statistische Methoden, Natural Language Processing (NLP), maschinelles Lernen und andere Techniken [JS17].

  Die Informationsextraktion (IE) umfasst verschiedene Ansätze, darunter die klassische Informationsextraktion und die Open Information Extraction (OpenIE). In der klassischen IE werden spezifische, vorher definierte Beziehungen zwischen Entitäten in Texten extrahiert, basierend auf einer festgelegten Ontologie. Hierbei ist die klare Definition des Verwendungszwecks entscheidend, da sie bestimmt, welche Entitäten und Beziehungen extrahiert werden. Im Gegensatz dazu strebt die OpenIE danach, Beziehungen flexibler zu gestalten. Sie ermöglicht es, Beziehungen zwischen Entitäten in Texten eigenständig zu identifizieren und zu extrahieren, ohne dass im Voraus spezifische Beziehungen festgelegt werden müssen. Dadurch kann sie auf verschiedene Anwendungsfälle angewendet werden [LS23, Lin23].

- **Die Sentimentanalyse** Die Stimmungsanalyse ist ein Prozess zur Bestimmung der hinter einer Aussage liegenden Meinung, des Urteils oder der Emotion. Sie bietet eine effektive Methode zur Bewertung von geschriebenem oder gesprochenem Text, um festzustellen, ob die Ausdrucksweise positiv, negativ oder neutral ist. Diese Analyse ermöglicht es, Schlüsse darüber zu ziehen, wie ein Kunde seine Erfahrung wahrgenommen hat, insbesondere wenn Kunden Online-Bewertungen oder Kommentare zu

Marken oder Produkten abgeben [LGF+22]. Die Sentiment-Analyse ist ein Teilbereich des Text Mining oder der Textanalyse, bei dem die Bedeutung von Textquellen wie Umfragen, Bewertungen, sozialen Medien und Internetartikeln extrahiert wird. Sie bewertet den Text anhand seiner Stimmung und verwendet Werte wie -1 für eine negative Stimmung und +1 für eine positive Stimmung. Dies erfolgt mithilfe von natürlicher Sprachverarbeitung (Natural Language Processing, NLP) [LL12]. Das Hauptziel der Sentiment-Analyse besteht darin, die Einstellung einer Person zu einem bestimmten Thema auf der Grundlage der Verarbeitung natürlicher Sprache zu bestimmen. Dies kann das Urteil, den emotionalen Zustand oder die beabsichtigte emotionale Kommunikation des Textverfassers sein [LGF+22].

- **Textklassifizierung:** Die Textklassifikation ist ein integraler Bestandteil des Natural Language Processing (NLP) und hat zum Ziel, vorherzusagen, in welche Kategorien bestimmte Texte innerhalb eines gegebenen Klassifikationssystems eingeordnet werden können. Bei der Textklassifikation stehen Forschern verschiedene Ansätze zur Merkmalsauswahl und zur Modellierung der Klassifikation zur Verfügung. Oft entscheiden sich Forscher jedoch dafür, vorgefertigte Methoden aus externen Bibliotheken zu nutzen, um ihre Ziele zu erreichen [Zhe19].

- **Spracherkennung:** Die Spracherkennung ist eine Methode zur präzisen Identifizierung von Texten basierend auf ihrem gegebenen Inhalt oder Thema. Durch umfangreiche Studien wird es ermöglicht, jede Sprache zu interpretieren und das Gesagte zu verstehen. Obwohl die Verarbeitung natürlicher Sprache (NLP) eine anspruchsvolle Technik ist, gibt es bemerkenswerte Beispiele wie Siri und Alexa, die darauf basieren. Die Fähigkeit zur Erkennung natürlicher Sprache ermöglicht es uns, die in einem bestimmten Dokument verwendete Sprache zu bestimmen und zu verstehen [RPS23].

- **Sprachassistent:** Ein Sprachassistent, auch unter dem Begriff sprachbasierte Künstliche Intelligenz bekannt, repräsentiert ein Softwareprogramm, das darauf ausgerichtet ist, mündliche Anweisungen von Menschen zu empfangen und entsprechend darauf zu reagieren. Dies ermöglicht eine Interaktion zwischen dem Individuum und dem Rechnersystem. In der heutigen schnelllebigen Welt sind Sprachassistenten weit verbreitet und erweisen sich als äußerst praktisch. Aufgrund ihrer Allgegenwärtigkeit greift nahezu jeder auf Sprachassistenten zurück. Ein prominenter Vertreter ist der Google Assistant, der sogar von Kindern im Alter von fünf Jahren verwendet werden kann. Insbesondere während der aktuellen weltweiten Pandemie hat die Nutzung von Smartphones zugenommen. Ein weiterer populärer Sprachassistent ist Amazon's Alexa, der in der Lage ist, eine Vielzahl von Aufgaben zu bewältigen. Eine bemerkenswerte Funktion besteht darin, dass er Menschen mit körperlichen Beeinträchtigungen, beispielsweise Personen, die nicht in der Lage sind zu laufen, unterstützen kann. Durch die Anwendung von Funktionen des Internet der Dinge (IoT) können sie Haushaltsgeräte steuern und verwalten. In Erwiderung auf diese

Entwicklungen wurde ein benutzerfreundlicher Sprachassistent entwickelt, der den aktuellen Trend im Bereich der Sprachassistenten reflektiert [SJA22].

- **Maschinelle Übersetzung:** Unter maschineller Übersetzung versteht man die automatische Textübertragung von einer Sprache in eine andere mithilfe von Rechnern. Sie ist ein Unterbereich der natürlichen Sprachverarbeitung (NLP), der darauf abzielt, menschenähnliche Übersetzungen zwischen verschiedenen Sprachen zu erzeugen, indem Computerprogramme Texte zu analysieren und sie in eine andere Sprache zu transferieren. Es ist wichtig zu betonen, dass die Übersetzung, vor allem im weiteren Sinne wie die Übertragung von Literatur oder Poesie, ein komplexes, faszinierendes und zutiefst menschliches Unterfangen ist, das ebenso vielfältig ist wie andere Bereiche der menschlichen Kreativität [JM09].

  Die maschinelle Übersetzung stellt einen Prozess dar, bei dem ein Software-System schriftliche Texte von einer Ausgangssprache in eine Zielsprache überträgt. Diese Art von Software-System wird als maschinelles Übersetzungssystem bezeichnet und basiert in der Regel auf grammatikalischen Regelmechanismen [GGGT21].

2.1.2 Text Mining

Die Prozesse der Extraktion und Analyse von Informationen aus Textdaten werden als Text Mining oder Text Analytics bezeichnet. Diese beiden Begriffe werden oft als Synonyme verwendet. Die Hauptaufgabe besteht darin, Texte in eine strukturierte Form zu bringen, um aus ihnen wesentliche Informationen zu extrahieren. Text Mining wird in verschiedenen Anwendungsbereichen eingesetzt, darunter Textklassifizierung, Clustering, Informationsextraktion, Plagiatserkennung und Information Retrieval. Im Bereich des Text Minings werden unterschiedliche Verfahren angewandt, um Informationen aus Texten zu extrahieren. Hierzu zählen typische Abläufe wie die Vorverarbeitung von Text, die Segmentierung von Text und die Kodierung von Text. [Gro22].
Die Textvorverarbeitung ist eine Aufgabe im Bereich der natürlichen Sprachverarbeitung (NLP), bei der der Eingabetext bereinigt wird, um bereinigte Tokens als Ausgabe zu erhalten. Diese Token stellen die Merkmale dar, die für die weitere Analyse verwendet werden können und können einzelne Wörter oder Wortgruppen sein. Die Textvorverarbeitung durchläuft verschiedene Phasen, darunter Tokenisierung, Vereinheitlichung, Bereinigung, Standardisierung, das Entfernen von Stoppwörtern und Stemming oder Lemmatisierung. In jeder dieser Phasen werden irrelevante Informationen aus dem ursprünglichen Text entfernt. Ein Hauptvorteil dieses Prozesses liegt in der Reduzierung des Textumfangs, was die Verarbeitungszeit verkürzt [Dee23, Mel23].

2.2 Information Retrieval

Information Retrieval, abgekürzt als IR, bezeichnet den Forschungsbereich, der sich mit der Suche nach verschiedenen Arten von Medien anhand der Informationsanforderungen der Nutzer befasst. Das daraus resultierende Information-Retrieval-System wird häufig als Suchmaschine bezeichnet [JM09].
Information Retrieval wird im Allgemeinen als ein Fachgebiet der Informatik angesehen, das sich mit der Repräsentation, dem Speichern und dem Abrufen von Informationen beschäftigt [SRM+12]. Information Retrieval (IR) bezieht sich auf den Prozess des Findens von unstrukturiertem Material, normalerweise in Form von Dokumenten, um den Informationsbedarf aus umfangreichen Sammlungen, die in der Regel auf Computern gespeichert sind, zu erfüllen. Ein Information-Retrieval-System informiert den Benutzer nicht über den Inhalt seiner Anfrage, sondern gibt lediglich Auskunft darüber, ob Dokumente vorhanden sind, die sich auf die Anfrage beziehen, und wo sie zu finden sind. Diese Charakterisierung stammt von Lancaster im Jahr 1968 [Lan68] und wird in einer Zeit verwendet, in der die Begriffe „Informationsexplosion“ und „Informationsüberflutung“ weit verbreitet sind [Mog09].

2.2.1 Text-kodierung

Nach der Vorverarbeitung und Segmentierung des Textes müssen die generierten Ergebnisse in numerische Merkmale umgewandelt werden, die dann für die Analyse durch maschinelles Lernen (ML), Deep Learning (DL) oder Techniken zur Ähnlichkeitsmessung verwendet werden können. Darüber hinaus benötigen viele ML- und DL-Algorithmen einen Merkmalsvektor mit fester Länge [Egg22]. Textkodierungstechniken werden zur Umwandlung von Wörtern, Phrasen oder Sätzen in Vektoren verwendet. Eine dieser Methoden, die so genannte Wortvektorisierung (word embedding), wandelt Wörter in Vektoren aus reellen Zahlen um. Diese Vektoren erfassen die semantische Bedeutung der Wörter [BHJ22]. Aus diesem Grund wird erwartet, dass Wörter mit ähnlicher Bedeutung in einem Vektorraum nahe beieinander liegen, was auf eine ähnliche Bedeutung hinweist. Die Vektorisierung von Wörtern findet in vielen Bereichen Anwendung, z. B. bei der Erkennung von Plagiaten und beim Abrufen von Informationen [Egg22].
In Abbildung 2.1 wird der Ablauf einer simplen Satzvektorisierung dargestellt. Dabei werden die Sätze in numerische Vektoren mit fester Länge umgewandelt. Dieser Vektorisierungsprozess trägt dazu bei, die Verarbeitung von Textinformationen, die oft unvollständig und vielfältig sind, im Vektorraum zu erleichtern. Das Resultat dieser Vektorisierung ist ein Vektor für jeden Satz, wobei ähnliche Sätze nahezu ähnliche Werte in den Vektoren aufweisen.

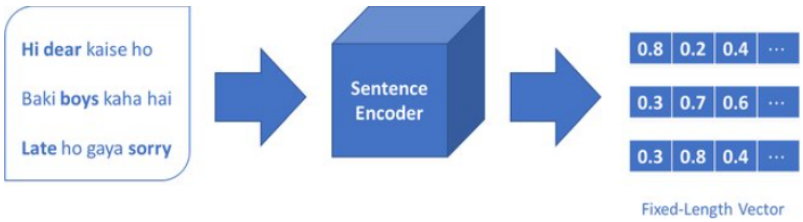


Abbildung 2.1: Der Prozess der Satzeinbettung [PS22]

2.2.2 Vektorsemantik

Die Vektorsemantik stellt die etablierte Methode zur Repräsentation der Wortbedeutung in Bereichen der Information Retrieval und Natural Language Processing (NLP) dar und trägt zur Modellierung zahlreicher Aspekte der Wortsemantik bei. Sie definiert die Bedeutung eines Wortes anhand seiner Verwendung im Sprachkontext, insbesondere in Bezug auf seine benachbarten Wörter oder grammatikalischen Umgebungen. Ähnliche Bedeutungen von zwei Wörtern werden erkannt, wenn sie in sehr ähnlichen Kontexten auftreten, was bedeutet, dass ihre Nachbarwörter oder grammatikalischen Umgebungen ähnlich sind [JM09]. Die Grundidee hinter der Vektorsemantik besteht darin, ein Wort als einen Punkt in einem mehrdimensionalen semantischen Raum zu repräsentieren, der sich aus den Verteilungen der benachbarten Wörter ergibt. Diese Vektoren, die zur Darstellung von Wörtern verwendet werden, werden als Einbettungen bezeichnet [JM09].

Worteinbettung

Die Darstellung von Wörtern in Vektorform erfolgt mithilfe von Einbettungsmodellen. Wort-Einbettung bezeichnet eine Sammlung statistischer Sprachmodellierungs- und Techniken, die im Bereich der natürlichen Sprachverarbeitung (NLP) eingesetzt werden. Sie transformiert Wörter und Ausdrücke in Vektoren von reellen Zahlen und erfasst dabei sowohl semantische als auch syntaktische Informationen von Wörtern. Die Wort-Einbettung kann zur Berechnung der Ähnlichkeit zwischen Wörtern verwendet werden, was in verschiedenen Aufgaben wie der Informationsgewinnung von Bedeutung ist. Es existieren verschiedene Ansätze zur Erzeugung von Wort-Einbettungen, darunter word2vec, doc2vec, tfidf und BERT-Word-Einbettung [Hu22].

- **Bag-of-words:** Rohdaten in Textform weisen eine variable Länge auf, während die meisten maschinellen Lernalgorithmen numerische Merkmalsvektoren als Eingabe erwarten. Um diese Anpassung zu ermöglichen, erfolgt eine Vektorisierung, die den Prozess der Umwandlung von Textdokumenten in numerische Merkmalsvektoren darstellt. Das Bag-of-Words-Modell erreicht dies durch eine Abfolge von Schritten, einschließlich Tokenisierung, Zählung und Normalisierung. Dieses Modell erfasst den gesamten Textkorpus und erzeugt eine effiziente Matrix. In dieser Matrix entspricht

jede Zeile der Häufigkeit von Wörtern in einem Dokument, während jede Spalte oder jedes Merkmal einem bestimmten Token oder einem eindeutigen Begriff entspricht [LM14].

- **Word2vec:** Word2vec ist eine auf NLP (Natural Language Processing) basierende Technologie, bei der ein Textkorpus als Eingabe dient und die Wörter in Form von Vektoren repräsentiert werden. Die räumliche Nähe dieser Vektoren im Vektorraum deutet auf semantische oder funktionale Ähnlichkeiten hin. Die resultierende Wortvektordatei kann in vielfältigen Anwendungen im Bereich der natürlichen Sprachverarbeitung und des maschinellen Lernens als Merkmal verwendet werden [MSC+13].

- **Doc2vec:** Das Doc2vec-Modell transformiert Textdokumente in numerische Repräsentationen. Jede dieser Repräsentationen ist ein Vektor, wobei die Dimensionalität des Vektors vom Nutzer festgelegt werden kann. Sowohl word2vec als auch Doc2vec sind Beispiele für unüberwachte Lernmethoden, wobei Doc2vec als eine Weiterentwicklung von Word2vec entstand. Doc2vec übernimmt die Vorteile von Word2vec, darunter die Berücksichtigung von semantischen Zusammenhängen und der Positionierung von Wörtern im Text [LM14, MSC+13].

- **tf-idf:** Die Termfrequenz-Inverse-Dokumentfrequenz (TF-IDF) ist eine gängige gewichtete Methode in den Bereichen Information Retrieval und Data Mining. In der Scikit-Learn-Bibliothek steht der tf-idf-Vectorizer zur Verfügung, um Text in eine numerische Matrix zu transformieren. Jedes Element dieser Matrix repräsentiert den TF-IDF-Wert des jeweiligen Wortes [SBM96].

- **BERT-Word-Embbedings:** BERT (Bidirectional Encoder Representations from Transformers) ist ein vortrainiertes NLP-Modell, das tiefgreifende bidirektionale Kontextinformationen liefert. Im Gegensatz zu traditionellen Word-Techniken, die nur unidirektionale Kontexte berücksichtigen, erfasst BERT den Kontext eines Wortes in beide Richtungen, was zu einer reichhaltigeren und genaueren Wortrepräsentation führt [McC19].

2.3 K-Means-Clustering und verwandte Verfahren

In diesem Abschnitt erfolgt eine umfassende Darstellung des Clustering-Prozesses, wobei auf die beiden weitverbreiteten Clustering-Algorithmen Hierarchisches Clustering und DBSCAN verzichtet wird. Stattdessen werden die Clustering-Verfahren K-Medoid, K-Means und FCK-Means im Detail und eingehend beleuchtet.

2.3.1 Clustering

Clustering ist eine Methode des maschinellen Lernens, des Data Mining und der Datenanalyse, bei der ähnliche Objekte oder Datenpunkte in Gruppen, sogenannten Clustern, zusammengefasst werden. Das Ziel des Clusterns besteht darin, einen großen Datensatz in kleinere Gruppen aufzuteilen. Jedes Cluster enthält Datenpunkte, die untereinander ähnliche Merkmale aufweisen, während sie sich von den Datenpunkten in anderen Clustern unterscheiden. Es gibt eine Vielzahl von Clustering-Algorithmen, die auf denselben Datensatz angewendet werden können und unterschiedliche Clusterstrukturen erzeugen können. Das Clustering von Daten hat zahlreiche Anwendungen in verschiedenen Bereichen wie Bilderkennung, Information Retrieval, Biologie und Sicherheit. Das Hauptziel des Clusterings besteht darin, latente Muster und Strukturen in den Daten zu entdecken, indem ähnliche Datenpunkte in Cluster zusammengefasst werden [NZT+11, DNHOK20].

Im Bereich des maschinellen Lernens und der Datenanalyse gibt es mehrere Clustering-Algorithmen, die verwendet werden, um Datenpunkte in Gruppen mit ähnlichen Eigenschaften zu organisieren. Zu den bekanntesten gehören Hierarchical Clustering und DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [McG20].

Hierarchisches Clustering ist ein Ansatz, bei dem Datenpunkte in einem hierarchischen Baum oder Dendrogramm organisiert werden. Dieser Algorithmus kann auf verschiedene Weise durchgeführt werden, wobei das agglomerative Clustering und das divisive Clustering die häufigsten Ansätze sind. Das sogenannte agglomerative Clustering beginnt mit einzelnen Datenpunkten als eigenen Clustern und fügt diese nach und nach zusammen, während das divisive Clustering mit einem einzigen großen Cluster beginnt und diesen in kleinere Cluster unterteilt. Das hierarchische Clustering bietet eine hierarchische Darstellung der Daten, so dass verschiedene Abstufungen der Auflösung erforscht werden können [McG20, Gru23].

DBSCAN ist ein dichtebasierter Clustering-Algorithmus, der auf der Erfassung von Gebieten mit hoher Datenpunktdichte basiert. Er identifiziert Cluster, indem er die Punkte in der Nähe eines Mittelpunkts gruppiert und Ausreißer von den Clustern trennt. DBSCAN ist besonders nützlich, wenn die Cluster unterschiedliche Strukturen und Größen haben und wenn es Ausreißer in den Daten gibt [SEKX98].

2.3.2 K-Medoid-Algorithmus

Die K-Medoid-Methode ist eine klassische Clustering-Technik, bei der ein Datensatz mit n Objekten in k a priori bekannte Cluster aufgeteilt wird. Im Vergleich zu k-Mittelwerten ist die Silhouette ein robusteres Werkzeug zur Bestimmung von k, da es weniger empfindlich gegenüber Rauschen und Ausreißern ist. Ein Medoid wird als das Objekt eines Clusters

definiert, dessen durchschnittliche Unähnlichkeit zu allen anderen Objekten im Cluster minimal ist. Dieses Objekt repräsentiert den zentralsten Punkt im gegebenen Datensatz [CY10].
Der K-Means-Algorithmus ist anfällig für Ausreißer, da ein Objekt mit einem extrem hohen Wert die Verteilung der Daten stark verzerren kann. Um diese Empfindlichkeit zu verringern, kann der Algorithmus modifiziert werden, indem statt des Durchschnittswerts der Objekte innerhalb eines Clusters ein Medoid als Referenzpunkt verwendet wird. Die Partitionierungsmethode basiert weiterhin auf dem Prinzip der Minimierung der Unähnlichkeiten zwischen jedem Objekt und seinem entsprechenden Referenzpunkt. Dies bildet die Grundlage für die K-Medoids-Methode. Die grundlegende Strategie der K-Medoids-Clustering-Algorithmen besteht darin, k Cluster in einer Menge von n Objekten zu identifizieren. Dazu werden zunächst für jeden Cluster willkürlich repräsentative Objekte als Medoide ausgewählt. Anschließend werden die verbleibenden Objekte dem Medoid zugeordnet, zu dem sie die höchste Ähnlichkeit aufweisen. Die K-Medoids-Methode verwendet somit repräsentative Objekte als Referenzpunkte anstelle des Durchschnittswerts der Objekte in jedem Cluster, um robustere Clustering-Ergebnisse zu erzielen. Der Algorithmus erfordert den Eingabeparameter k, der die Anzahl der Cluster angibt, in die die gegebene Menge von n Objekten partitioniert werden soll [EJYHG20, CY10]. Partitioning Around Medoids (PAM) gehört zu den frühesten k-Medoids-Algorithmen, die entwickelt wurden. Das Ziel besteht darin, k Partitionen für n Objekte zu finden. Der Algorithmus startet mit einer zufälligen Auswahl von k Medoiden und versucht dann wiederholt, eine verbesserte Auswahl an Medoiden zu finden [CY10].

2.3.3 K-Means-Algorithmus

Der k-Means-Algorithmus ist ein weit verbreiteter Clustering-Algorithmus im Bereich des Data Mining, der häufig zur Gruppierung großer Datenmengen eingesetzt wird. Er wurde erstmals 1967 von MacQueen [M+67] vorgeschlagen und zählt zu den einfachsten nicht-überwachten Lernalgorithmen, die zur Bewältigung des Problems des bekannten Clusters angewendet werden [Sun08]. Der besagte Algorithmus ist ein Clustering-Algorithmus auf Partitionsbasis. Diese Methode besteht darin, die gegebenen Datenobjekte durch wiederholte Iteration in k verschiedene Cluster zu klassifizieren und auf ein lokales Minimum zu konvergieren. Die resultierenden Cluster weisen eine kompakte und unabhängige Struktur auf. Der Algorithmus umfasst zwei diskrete Phasen. In der ersten Phase erfolgt eine zufällige Auswahl von k Zentren, wobei der Wert von k im Voraus festgelegt wird. Die nachfolgende Phase besteht darin, jedes Datenobjekt dem nächstgelegenen Zentrum zuzuordnen [Fah21]. Im K-Means-Clustering werden verschiedene Distanzfunktionen genutzt, um den Abstand zwischen den einzelnen Datenobjekten und den Clusterzentren zu berechnen. Zu den vom Algorithmus verwendeten Distanzmetriken gehört die euklidischen-Entfernungsmetrik, einschließlich der Minkowski Entfernungsmetrik. Sobald alle Datenobjekte in bestimmten Clustern enthalten sind, ist der erste Schritt abgeschlossen und eine vorzeitige Gruppierung wird durchgeführt. Anschließend erfolgt eine Neuberechnung des

Durchschnitts der vorzeitig gebildeten Cluster. Dieser iterative Prozess wird fortgesetzt, bis die Kriteriumsfunktion ihr Minimum erreicht. Obwohl K-Means einfach ist und auf eine breite Palette von Datentypen angewendet werden kann, hat er einen bestimmten Nachteil, nämlich dass der K-Means-Algorithmus rechenintensiv ist und mehr Zeit in Anspruch nimmt, insbesondere im Verhältnis zur Anzahl der Datenelemente, der Clusteranzahl und der Iterationen. Des Weiteren muss die Anzahl der Cluster im Voraus festgelegt werden [MB00]. Die Abbildung 2.2 zeigt die Schritte des K-Means-Clustering-Algorithmus.

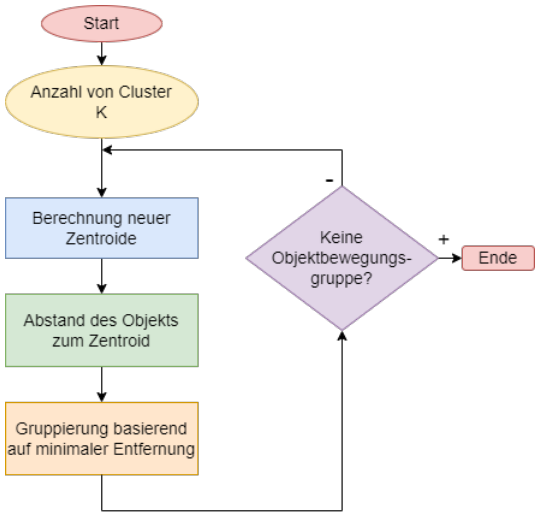


Abbildung 2.2: Schritte des K-Means-Clusterings-Algorithmus [KCA16]

Der k-Means-Algorithmus besteht aus den folgenden Schritten, die wiederholt durchgeführt werden [1]:

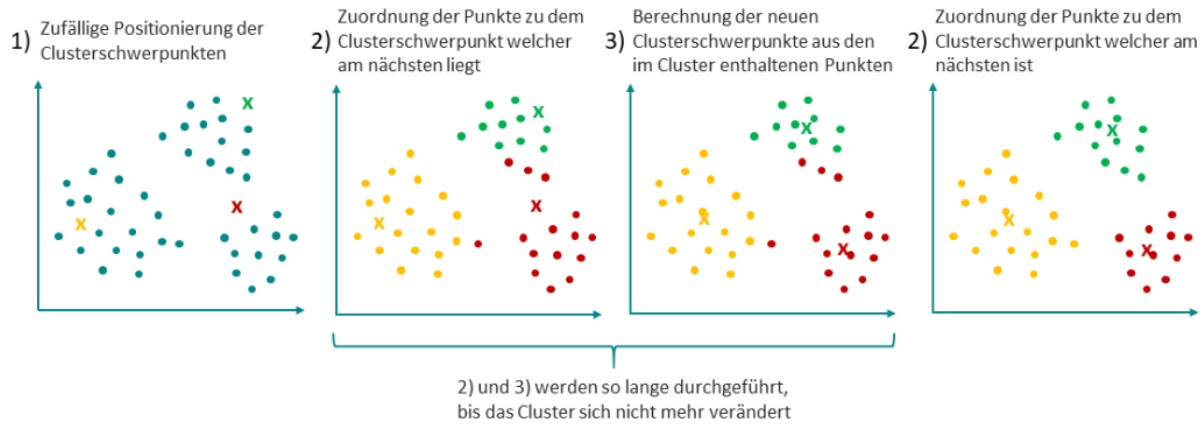


Abbildung 2.3: Schritte der Initialisierung des K-Means-Clusterings-Algorithmus

[1] Quelle: https://datatab.de/tutorial/k-means-clusteranalyse

2.3.4 FCK-Means-Algorithmus

Mit Hilfe des K-Means-Algorithmus werden die anfänglichen Clusterzentren durch ein Zufallsverfahren ermittelt. Im ersten Schritt des Prozesses werden einige Datenpunkte nach dem Zufallsprinzip ausgewählt, die als anfängliche Clusterzentren dienen. Im zweiten Schritt des Prozesses werden die verbleibenden Datenpunkte auf der Grundlage eines Kriteriums, das als Mindestabstand bezeichnet wird, auf die Zentren verteilt. Danach werden die neuen Clusterzentren für die Punkte, die den Clustern zugeordnet wurden, bestimmt. Danach werden die Abstände zwischen den neuen Clusterzentren und allen Punkten neu berechnet, und wenn es potenzielle Punkte mit einem minimalen Abstand gibt, die den neuen Zentren zugeordnet werden können, dann werden diese potenziellen Punkte den neuen Zentren zugewiesen. Der Clustering-Prozess ist abgeschlossen, wenn sich entweder die Zentren der Cluster nicht mehr verschieben oder die maximale Anzahl der Iterationen erreicht ist. Bei der Betrachtung der Prozesse des Algorithmus wird deutlich, dass dieser Algorithmus sehr abhängig von den ersten Clusterzentren ist und dass die Auswahl der Anfangszentren zu unterschiedlichen Clustern und Clusterzentren führen kann. Es ist wichtig zu bedenken, dass bei den Algorithmen K-Means und K-Means++ die Zentren der einzelnen Cluster nach jeder Iteration geändert werden. Der wichtigste Beitrag dieser Forschungsarbeit ist die Möglichkeit, eine Clusteranalyse durchzuführen, indem ein Teil der Clusterzentren fixiert und nicht aktualisiert wird, anstatt alle Clusterzentren zu aktualisieren [A+23].

Der FCK-Means-Algorithmus kann in zwei Phasen unterteilt werden. K-Means++ wird in Phase I verwendet, um die ersten Kandidaten-Clusterzentren für „k“ zu finden. Das Clustering wird sowohl mit festen als auch mit nicht festen Clusterzentren in Phase II des Prozesses durchgeführt. Der K-Means-Algorithmus wird ausgeführt, nachdem die nicht festen Zentren von K-Means++ in Phase I identifiziert wurden, und liefert dann „k“ Clusterzentren und „k“ Cluster. Es werden die „k-m“ nicht-fixierten Clusterzentren ausgewählt, die am weitesten von den fixierten Zentren entfernt sind [A+23].

Im ersten Schritt des Prozesses wird ein Durchschnitt der Gesamtentfernungen zwischen jedem Kandidatenzentrum und allen festen Zentren ermittelt. Danach werden die „k-m“-Zentren mit dem höchsten Wert entsprechend diesen durchschnittlichen Entfernungen ausgewählt. In Phase II werden die ausgewählten Zentren und die festen Clusterzentren zu den so genannten anfänglichen Clusterzentren zusammengefasst. Danach werden die Punkte den Zentren zugewiesen, die am nächsten liegen. Während des gesamten vorgeschlagenen Prozesses erhalten die festen Clusterzentren keine Aktualisierungen. Im Rahmen der Methode werden nur die Zentren von Clustern, die nicht fest bleiben, neu berechnet und aktualisiert [A+23].

Der letzte Schritt des Algorithmus besteht in der Bildung von „k“ Clustern mit festen und nicht festen Clusterzentren. Dieser Schritt erfolgt entweder, wenn der Algorithmus die ma-

ximale Anzahl von Iterationen für Phase II erreicht hat oder wenn sich die nicht fixierten Mittelpunkten nicht mehr ändern. Der wichtigste Unterschied zwischen diesem Ansatz und dem K-Means-Algorithmus besteht darin, dass bei dieser Methode nicht alle Zentren aktualisiert werden, sondern nur die Zentren der nicht fixierten Cluster. Die Werte der festen Clusterzentren ändern sich nicht, da sie in diesem Aktualisierungsschritt des Algorithmus nicht berücksichtigt werden. Ein weiterer Vorteil dieser Methode besteht darin, dass die Ergebnisse des K-Means++ Algorithmus als Anfangswerte für die nicht fixierten Clusterzentren verwendet werden können. Diese anfänglichen, nicht fixierten Clusterzentren ermöglichen ein besseres Clustering mit weniger Iterationen in Phase II zu erreichen [A+23].

Die Berechnung des Abstands zwischen den in Phase I ermittelten Clusterzentren und den für Phase II festgelegten Clusterzentren ist einer der wichtigsten zu berücksichtigenden Punkte. Folglich kann das Ergebnis der FCK-Means-Phase-I verwendet werden, um die „k-m“ nicht fixierten Clusterzentren auszuwählen, die am weitesten von den fixierten Clusterzentren für Phase-II entfernt sind. Die Neuberechnung der Clusterzentren in Phase II ist ein weiterer wichtiger Aspekt, der berücksichtigt werden muss. Während der Iterationen werden nur die nicht fixierten Clusterzentren neu berechnet und aktualisiert, während die fixierten Clusterzentren keine Änderungen erfahren. Die Ergebnisse der Phase I sind in Form von Clusterzentren zu sehen. Der FCK-Means-Algorithmus verwirft von diesen Punkten diejenigen, die sich in der Nähe der festen Clusterzentren befinden, und wählt stattdessen die Zentren aus, die am weitesten entfernt sind. Ein Vorteil der geringeren Anzahl von Iterationen in Phase II ist die anfängliche Auswahl von nicht fixierten Clusterzentren, die aus dem Ergebnis von Phase I stammen [A+23].

3 Methodik

In der vorliegenden Bachelorarbeit wurde eine quantitative Forschungsart mit Schwerpunkt auf dem maschinellen Lernen angewendet.

3.1 Zweck der Forschung

Der Hauptzweck der Forschung besteht darin, ein optimiertes Clustering-Verfahren für Textdokumente zu entwickeln und zu implementieren. Speziell wird eine Variante des bekannten K-Means Clustering Algorithmus, nämlich das Fixed Centroids K-Means Clustering (FCK-Means), verwendet. Durch die Kombination verschiedener Informationsquellen aus den Texten und der Anwendung von FCK-Means sollte eine sinnvolle Gruppierung der Dokumente erzielt werden.

3.2 Datenerhebung

Die Datenerhebung erfolgte durch das Sammeln von Textdokumenten. Diese Textdokumente repräsentieren das Hauptdatenset, auf dem die gesamte Analyse basierte. Es ist zu beachten, dass in dieser Forschung eine ausreichende Menge an Daten erhoben wurde, um statistisch signifikante Ergebnisse zu erzielen.

3.3 Merkmale der erhobenen Daten

Die erhobenen Daten bestanden aus unstrukturierten, textbasierten Dokumenten. Diese variierten in ihrer Länge und ihrem Inhalt, repräsentierten jedoch alle eine Form von Textinformationen. Die Daten erforderten eine sorgfältige Vorverarbeitung und Transformation, bevor sie für maschinelles Lernen und Clustering geeignet waren.

3.4 Durchführung der Forschung

Die Forschung wurde in vier Hauptphasen durchgeführt:

3.4.1 Datenaufbereitung

Die Datenaufbereitung (Abb. 3.1) für die Textdokumente besteht aus mehreren wichtigen Schritten. Zunächst werden alle Kontraktionen (wie z.B. „don't“ zu „do not“) erweitert und den Text in einzelne Wörter, einen Prozess, der als Tokenisierung bekannt ist zerteilt. Danach werden alle Wörter in Kleinbuchstaben umgewandelt, um Konsistenz zu gewährleisten. Im nächsten Schritt werden sogenannte Stoppwörter entfernt, also Wörter, die häufig vorkommen, aber wenig semantische Information tragen, wie „the“, „is“, „in“ usw. Anschließend werden die verbleibenden Wörter lemmatisiert, also auf ihre Grundform reduziert, um verschiedene Formen desselben Wortes gleich zu behandeln. Nichtalphanumerische Zeichen und Satzzeichen werden ebenfalls entfernt. Die endgültigen aufbereiteten Textdaten werden dann wieder zu einem einzigen String zusammengefügt und sind bereit für nachfolgende Verarbeitungsschritte im Bereich des Natural Language Processing.

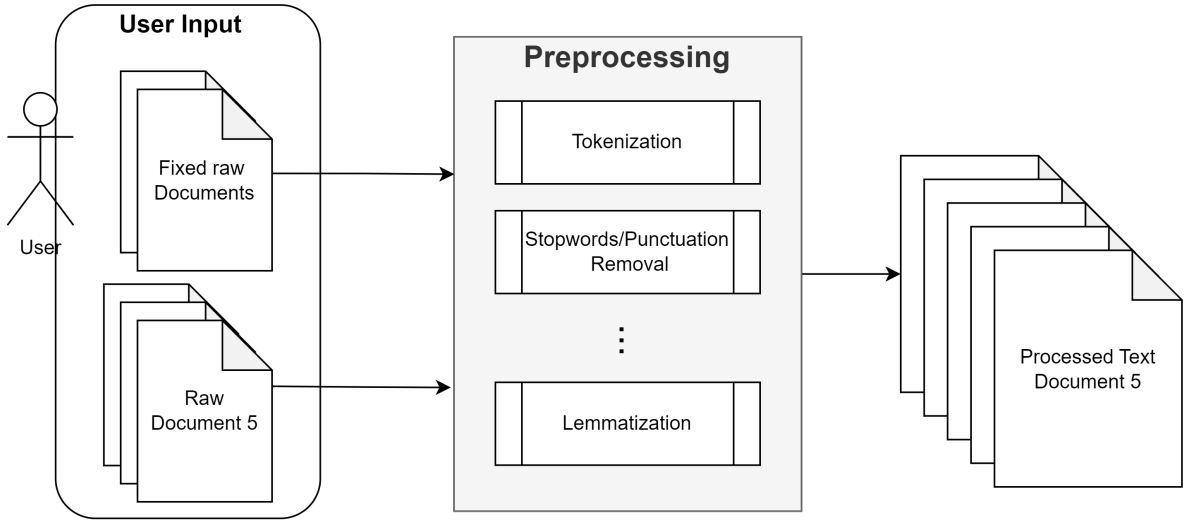


Abbildung 3.1: Eingabe und Ausgabe des Datenaufbereitung-Schrittes.

Am folgenden steht ein kleines Beispiel für die Erklärung der oben erwähnten Schritten für die Anwendung vom Preprocessing auf einem kleinen Text:
Ursprünglicher Text: „Hallo! Wie geht es Ihnen? Ich hoffe, Sie haben einen grossartigen Tag.“
Die Schritte:

1. Tokenisierung:
   Verarbeiteter Text: [Hallo, wie, geht, es, Ihnen, Ich, hoffe, Sie, haben, einen, grossartigen, Tag]

2. Kleinschreibung:
   Verarbeiteter Text: [hallo, wie, geht, es, ihnen, ich , hoffe, sie, haben, einen, grossartigen, tag]

3. Stoppwortentfernung:
   Verarbeiteter Text: [hallo, geht, hoffe, grossartigen, tag]

4. Stammwortreduktion:
   Verarbeiteter Text: [hallo, geh, hoff, grossartig, tag]

3.4.2 Merkmalsextraktion

In der Phase der Merkmalsextraktion (Abb. 3.2) wird jedes Dokument auf mehrere Arten repräsentiert: Bag of Words (BoW), TF-IDF, Word Embeddings mittels BERT und mittels Word2Vec und Sentiment Scores via NLTK Sentiment Intensity Analyzer. BoW und TF-IDF modellieren Dokumente basierend auf Wortfrequenz, wobei TF-IDF zusätzlich die Inverse Dokumentenhäufigkeit berücksichtigt, um seltene Wörter stärker zu gewichten. BERT-basierte Word Embeddings erfassen semantische Ähnlichkeiten und Kontextinformationen, während Sentiment Scores die in den Dokumenten ausgedrückte Stimmung abbilden. Diese Kombination von Repräsentationen liefert eine umfassende Charakterisierung der Textdaten und bereichert die folgenden Modellierungsschritte.

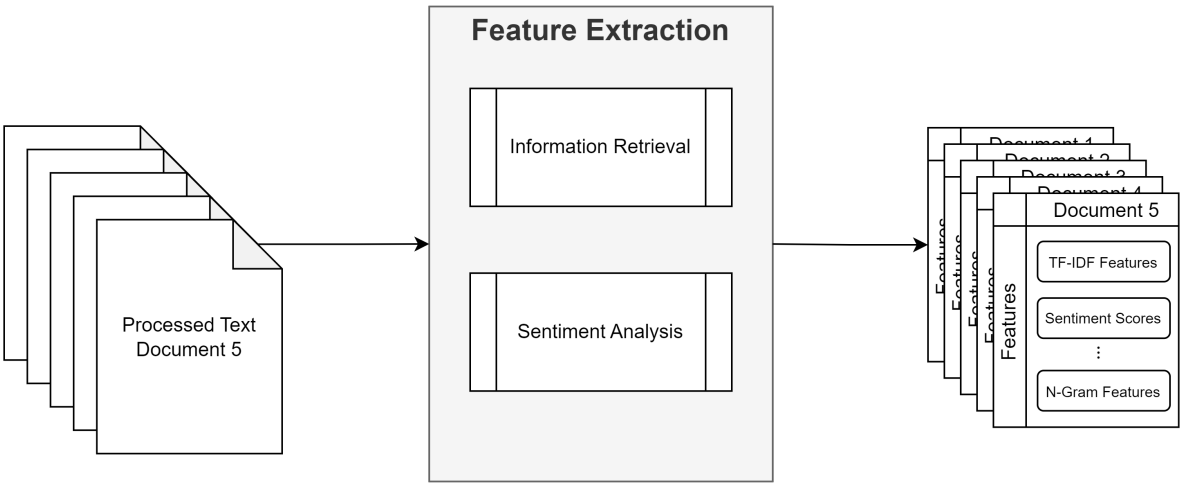


Abbildung 3.2: Eingabe und Ausgabe des Merkmalsextraktion-Schrittes.

Im Folgenden steht ein kleines Beispiel für die Erklärung der Schritte, wie das am letzten Schnitt erwähnten Beispieltext mit TF-IDF, eine der mehreren Arten der Repräsentation der Dokumente, berechnet wird:
Annahme: Es gibt zwei Dokumente und jedes davon enthält einen Satz.

Ursprünglicher Text 1: „Hallo! Wie geht es Ihnen? Ich hoffe, Sie haben einen grossartigen Tag.“
Ursprünglicher Text 2: „Geht es Ihnen gut? Ich hoffe, Sie haben einen wundervollen Tag.“

1. Termfrequenz (TF):

$$TF(w) = \frac{\text{Anzahl des Vorkommens von Wort } w \text{ im Dokument}}{\text{Gesamtanzahl der Wörter im Dokument}} \tag{3.1}$$

2. Inverse Dokumentfrequenz (IDF):

$$IDF(w) = \log\left(\frac{\text{Gesamtanzahl der Dokumente}}{\text{Anzahl der Dokumente, die Wort } w \text{ enthalten}}\right) \tag{3.2}$$

3. TF-IDF-Gewichtung:

$$TF - IDF(w) = TF(w) \cdot IDF(w) \tag{3.3}$$

Wort	TF im Satz 1	TF im Satz 2	IDF	TF-IDF im Satz 1	TF-IDF im Satz 2
hall	1/5	0	$\log\left(\frac{2}{1}\right)$	$1/5 \cdot \log\left(\frac{2}{1}\right)$	0
geht	1/5	1/4	$\log\left(\frac{2}{2}\right)$	$1/5 \cdot \log\left(\frac{2}{2}\right)$	$1/4 \cdot \log\left(\frac{2}{2}\right)$
hoff	1/5	1/4	$\log\left(\frac{2}{2}\right)$	$1/5 \cdot \log\left(\frac{2}{2}\right)$	$1/4 \cdot \log\left(\frac{2}{2}\right)$
grossartig	1/5	0	$\log\left(\frac{2}{1}\right)$	$1/5 \cdot \log\left(\frac{2}{1}\right)$	0
tag	1/5	1/4	$\log\left(\frac{2}{2}\right)$	$1/5 \cdot \log\left(\frac{2}{2}\right)$	$1/4 \cdot \log\left(\frac{2}{2}\right)$
wundervoll	0	1/4	$\log\left(\frac{2}{1}\right)$	0	$1/4 \cdot \log\left(\frac{2}{1}\right)$

Tabelle 3.1: TF-IDF-Werte für beide Sätze

3.4.3 Clustering

In der Clustering-Phase (Abb. 3.3) werden die Dokumente basierend auf ihren Merkmalen gruppiert. Anfangs werden einige Datenpunkte mittels eines zufälligen Auswahlverfahrens ausgewählt, um die anfänglichen Zentren der Cluster zu repräsentieren. Im Anschluss werden die übrigen Datenpunkte basierend auf dem Kriterium des minimalen Abstands den bereits festgelegten Zentren zugeordnet. Um die Abstände zwischen diesen Datenpunkten zu berechnen, stehen zur Verfügung mehrere Distanzmaße, wie Euclidean, Minkowski, Jaccard und in dieser Arbeit werden die Euklidische (3.4)- und Minkowski (3.5)-Distanz als Distanzmetrik verwendet. Hierbei wird eine modifizierte Version des K-Means-Algorithmus genutzt, genannt FCK-Means (Fixed Centroids K-Means) [A+23]. Im Gegensatz zu herkömmlichem K-Means erlaubt FCK-Means das Festlegen bestimmter Dokumente als „feste Zentren“. Diese festen Zentren dienen als Startpunkte für das Clustering und bleiben während des gesamten Iterationsprozesses unverändert. So können Nutzer das Clustering gezielt beeinflussen und sicherstellen, dass bestimmte Dokumente immer in spezifischen Clustern landen. Dieser Ansatz erhöht die Flexibilität des Clustering-Prozesses und ermöglicht eine anwendungsspezifische Feinabstimmung.

Die euklidische-Distanz wird durch folgende Formel dargestellt [A+23]:

$$\text{euklidische Distanz} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.4}$$

Die Minkowski-Distanz wird durch folgende Formel dargestellt [Min20]:

$$\text{Minkowski-Distanz} = \left(\sum_{i=1}^{n} |a_i - b_i|^p \right)^{1/p} \tag{3.5}$$

3.4.4 Visualisierung der Ergebnisse

Die Visualisierung der Ergebnisse spielt eine entscheidende Rolle, um die Relevanz und die Auswirkungen des Clustering-Prozesses zu verstehen. Nach der Durchführung des Clustering wurden die Ergebnisse in Form eines 2D-Streudiagramms dargestellt. In diesem Diagramm wurden die Cluster-Etiketten jedes Dokuments angezeigt, wodurch eine klare Unterscheidung zwischen den einzelnen Clustern ermöglicht wurde. Da die Merkmale in einem hochdimensionalen Raum vorliegen, musste die Dimensionalität dieses Merkmalsraums zur Erstellung des Streudiagramms reduziert werden. Dies wurde durch die Anwendung der Methode der Hauptkomponentenanalyse (PCA) erreicht, einer gängigen Methode zur Dimensionsreduktion. Der reduzierte Merkmalsraum behält dabei die wesentlichen strukturellen Informationen des hochdimensionalen Raums bei, wodurch das

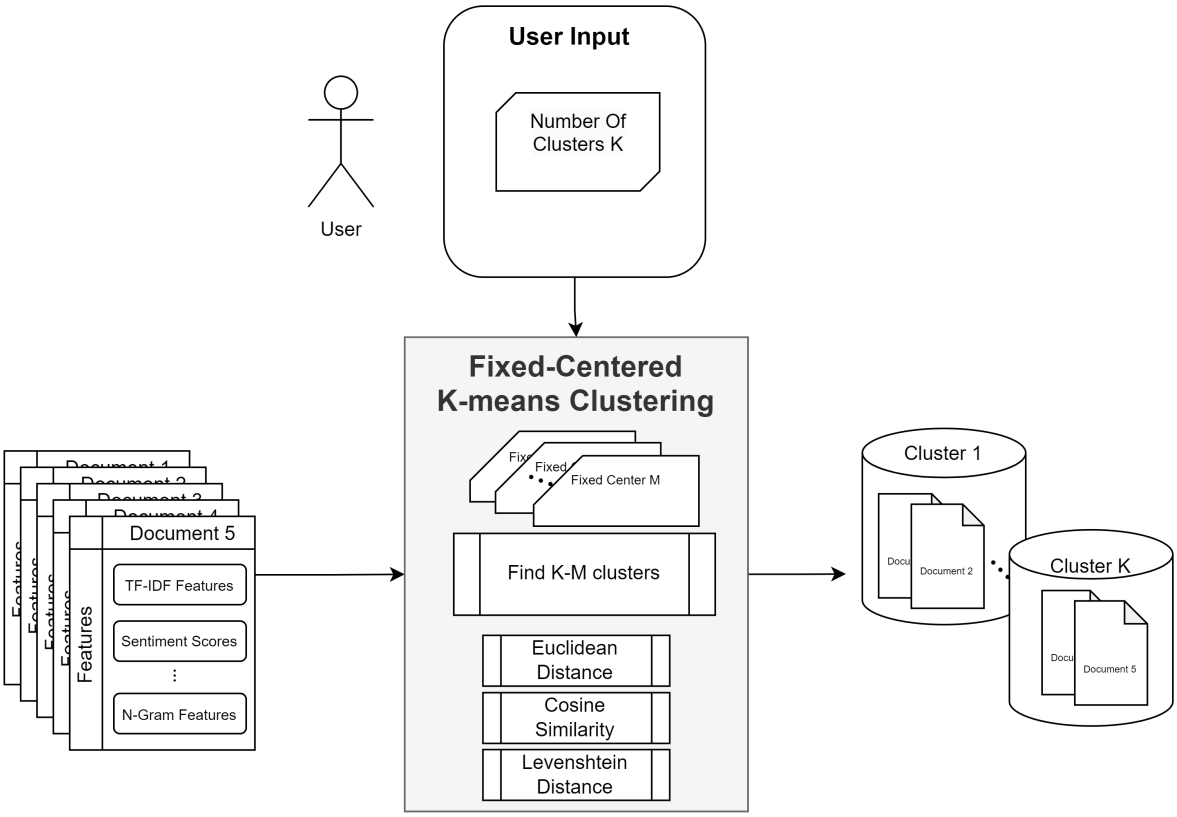


Abbildung 3.3: Eingabe und Ausgabe des FCK Means-Clustering-Schrittes.

Clustering-Ergebnis auf eine visuell zugängliche Weise dargestellt werden konnte. Weitere Details zu diesem Vorgang, einschließlich Implementierungsdetails und Evaluation der Ergebnisse, werden in den entsprechenden Kapiteln behandelt.

3.5 Validität und Reliabilität

Bei maschinellem Lernen und Textclustering hängen Validität und Reliabilität von verschiedenen Faktoren ab, darunter die Qualität und Repräsentativität der Daten, die Robustheit der verwendeten Algorithmen und die Sorgfalt bei der Datenverarbeitung und -analyse. In dieser Forschungsarbeit wurden etablierte Verfahren und Algorithmen verwendet und die Daten gründlich aufbereitet und analysiert, um eine hohe Validität und Reliabilität zu gewährleisten. Das Verfahren, das in dieser Arbeit verwendet wird, um eine hohe Validität und Reliabilität zu gewährleisten, ist Silhouette Score. Dazu kann dieses Verfahren die Clustering-Ergebnisse bewerten [MK20].
Der Silhouette Score wird mit folgender Formel berechnet [Bha21]:

$$\text{Silhouette Score} = \frac{b - a}{\max(a, b)}$$

Wobei a steht für die durchschnittliche Distanz innerhalb eines Clusters, das heißt, es repräsentiert den mittleren Abstand zwischen den individuellen Punkten innerhalb desselben Clusters. b repräsentiert die durchschnittliche Distanz zwischen den Clustern. Das bedeutet, dass es sich um die mittlere Distanz zwischen allen Clustern handelt.

4 Implementierung

In der vorliegenden Arbeit wird ein neuartiger Ansatz zur Klassifizierung und Gruppierung von Textdaten mittels maschinellen Lernens vorgestellt. Übergeordnetes Ziel dieses Projekts ist die Entwicklung einer effektiven sowie effizienten Methode zur Verwaltung und Kuratierung umfangreicher Mengen unstrukturierter Textdaten. Der präsentierte Ansatz stützt sich auf eine Kombination verschiedener Techniken zur Vorverarbeitung und Merkmalsextraktion von Textdaten. Hierbei ist insbesondere der FCK-Means-Clustering-Algorithmus hervorzuheben, der eine Gruppierung ähnlicher Dokumente ermöglicht. Ergänzt wird dieser durch Methoden zur Reduktion der Dimensionalität und Visualisierung, um die erzielten Resultate ansprechend und verständlich darzustellen. Der gesamte Ansatz nutzt bewährte Techniken und Bibliotheken im Bereich des maschinellen Lernens und der natürlichen Sprachverarbeitung, wie beispielsweise Scikit-Learn, NLTK und Transformers. Implementiert wurde das System in Python, welches sich in der maschinellen Lerngemeinschaft als weitverbreitete und akzeptierte Programmiersprache etabliert hat. Im Kontext von Empfehlungssystemen, Informationsmanagement und Wissensorganisation sowie Content Aggregation und Kuratierung bietet der hier vorgestellte Ansatz wertvolle Unterstützung für Nutzer, die mit einer Flut von Informationen konfrontiert sind. Es erlaubt ihnen, relevante Inhalte effizient zu finden und zu entdecken. In der vorliegenden Arbeit werden die spezifischen Schritte und Entscheidungen, die bei der Implementierung des Systems getroffen wurden, eingehend diskutiert und analysiert, um einen klaren Einblick in die angewandte Methodik zu gewähren. Dabei werden auch Herausforderungen und Hindernisse, die während des Prozesses aufgetreten sind, sowie die Strategien zu ihrer Bewältigung thematisiert.

4.1 Technologien und Werkzeuge

Die Wahl der Programmiersprache und der verwendeten Bibliotheken spielen eine entscheidende Rolle bei der Entwicklung von maschinellem Lernen und der Verarbeitung natürlicher Sprache. Sie beeinflusst nicht nur die Leistungsfähigkeit und Effizienz des erstellten Systems, sondern auch die Einfachheit und Klarheit des Entwicklungsprozesses. Für dieses Projekt wird Python Version 3.11 als Programmiersprache ausgewählt. Python hat sich aufgrund seiner Einfachheit und seiner starken Unterstützung für wissenschaftliche Berechnungen als die dominante Sprache in den Bereichen maschinelles Lernen und Datenwissenschaft etabliert. Darüber hinaus bietet Python eine Vielzahl von Bibliotheken und Frameworks, die speziell für Aufgaben im Bereich des maschinellen Lernens und der Verar-

beitung natürlicher Sprache entwickelt wurden. Eine dieser Bibliotheken ist Scikit-Learn Versionsnummer 1.3.0, eine allgemeine Bibliothek für maschinelles Lernen, die eine breite Palette von Algorithmen und Hilfsfunktionen für Aufgaben wie Klassifikation, Regression, Clustering und Dimensionsreduktion bietet [1]. In diesem Projekt wird Scikit-Learn für die Implementierung des FCK-Means-Clustering-Algorithmus sowie für die Dimensionsreduktion und Visualisierung der Daten genutzt. Die Verarbeitung und Analyse von Textdaten werden durch die Verwendung der NLTK (Natural Language Toolkit) Versionsnummer 3.8.1 Bibliothek unterstützt. NLTK bietet eine Vielzahl von Funktionen und Ressourcen für Aufgaben wie Tokenisierung, Stoppwortentfernung, Stemming und Sentimentanalyse [2]. In Kombination mit dem VADER (Valence Aware Dictionary and Sentiment Reasoner) Sentimentanalyse-Tool, das auch in NLTK integriert ist, konnte eine zuverlässige Bewertung der Stimmung innerhalb der Textdaten durchgeführt werden. Schließlich wird für die Erzeugung der BERT-Word-Embeddings die Transformers-Bibliothek von Hugging Face genutzt. Diese Bibliothek bietet eine einfache und effiziente Möglichkeit, vortrainierte Modelle wie BERT für spezifische NLP-Aufgaben einzusetzen. Durch die Erzeugung von BERT-Word-Embeddings konnten detaillierte semantische Darstellungen der Textdaten erzeugt werden, die wesentlich zur Genauigkeit des Clustering-Prozesses beitrugen. Zusammengefasst haben Python und die ausgewählten Bibliotheken eine effektive und effiziente Implementierung des vorgeschlagenen Systems ermöglicht, während gleichzeitig ein hoher Grad an Flexibilität und Anpassungsfähigkeit für zukünftige Erweiterungen und Verbesserungen gewährleistet wurde.

4.2 Vorverarbeitung der Daten

Die Vorverarbeitung von Textdaten ist ein kritischer Schritt bei der Anwendung von maschinellem Lernen und NLP-Techniken. Sie zielt darauf ab, die Daten in ein geeignetes Format für die anschließende Analyse zu bringen und gleichzeitig Rauschen und irrelevante Informationen zu reduzieren. Bei diesem Projekt wird eine Kombination verschiedener Vorverarbeitungstechniken angewendet. Die vorliegenden Textdaten dienen als Grundlage für die darauf folgenden Clusteringschritte. Diese Daten bestehen aus insgesamt 256 Textelementen.

[1] https://scikit-learn.org/stable/tutorial/basic/tutorial.html
[2] https://realpython.com/nltk-nlp-python/

4.2.1 Textreinigung und Normalisierung

Die erste Stufe der Vorverarbeitung besteht in der Reinigung und Normalisierung des Texts. Das umfasst die Entfernung von Zeichenformatierung, die Umwandlung aller Texte in Kleinbuchstaben sowie das Entfernen von Zahlen und Sonderzeichen. Diese Schritte sind wichtig, um sicherzustellen, dass ähnliche Worte, die in verschiedenen Formaten vorliegen, als gleich erkannt werden. Zum Beispiel sollten „Email“, „email“ und „e-mail“ als dasselbe Wort behandelt werden. Darüber hinaus werden während der Reinigungsphase URLs und HTML-Tags entfernt, da sie in der Regel keine relevanten Informationen für die Textklassifizierung liefern. Stattdessen könnten sie das Modell ablenken und die Qualität der Ergebnisse beeinträchtigen.

4.2.2 Tokenisierung, Stoppwortentfernung und Stemming

Nach der Reinigung und Normalisierung wird der Text tokenisiert, d.h. in einzelne Wörter (Tokens) aufgeteilt. Dies ist eine grundlegende Voraussetzung für die meisten NLP-Techniken und maschinellen Lernverfahren. Anschließend werden Stoppwörter entfernt. Stoppwörter sind häufig vorkommende Wörter wie „und“, „ist“, „ein“, die in der Regel keine nützlichen Informationen für Textklassifizierungsaufgaben liefern. Durch die Entfernung von Stoppwörtern kann das Modell sich auf die relevantesten Wörter konzentrieren. Schließlich wird Stemming auf die verbleibenden Wörter angewendet. Stemming reduziert Wörter auf ihren Stamm oder Wurzel, indem es Endungen entfernt. Dies kann dazu beitragen, die Dimensionalität des Feature-Raums zu reduzieren und ähnliche Wörter zusammenzufassen.

Im folgenden Codeausschnitt 4.1 werden zunächst verschiedene Bibliotheken, die für die Verarbeitung und Analyse von Textdaten wesentlich sind, importiert. Anschließend werden die Stoppwörter sowie die Spacy Lemmatisierung initialisiert.
Die `preprocess text`-Funktion führt eine Reihe von Schritten aus, um den Eingabetext für NLP-Aufgaben vorzubereiten, einschließlich der Expansion von Kontraktionen, der Tokenisierung, der Konvertierung in Kleinbuchstaben, der Entfernung von Stoppwörtern, der Lemmatisierung und der Filterung von Nicht-Alphanumerik.

```
import nltk
import contractions
import spacy
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)

stop_words = set(stopwords.words('english'))
nlp = spacy.load('en_core_web_sm', disable=['parser', 'ner'])
return " ".join(processed_tokens)

def preprocess_text(text):

text = contractions.fix(text)
 tokens = word_tokenize(text)

  processed_tokens = [token.lemma_ for token in nlp(' '.join(tokens))
                      if (token.lower_ not in stop_words) and token.
   text.isalnum()]
return " ".join(processed_tokens)
```

Listing 4.1: Textverarbeitung

4.3 Merkmalsextraktion

Nach der Vorverarbeitung der Daten folgte die Merkmalsextraktion, die darauf abzielt, repräsentative Merkmale aus dem Text zu extrahieren, die für das maschinelle Lernen verwendet werden können. Bei diesem Projekt wurden drei Haupttechniken zur Merkmalsextraktion angewendet: Bag-of-Words (BoW) und TF-IDF für die Gewichtung von Wörtern, Sentiment-Analyse zur Bewertung der Stimmung in den Texten und BERT-Word-Embeddings zur Erfassung komplexer semantischer Beziehungen.

4.3.1 TF-IDF

Eine weit verbreitete Methode zur Darstellung von Textdaten in maschinellem Lernen und Natural Language Processing (NLP) ist TF-IDF (Term Frequency-Inverse Document Frequency). Diese Methode berücksichtigt sowohl die Häufigkeit eines Wortes in einem bestimmten Dokument (Term Frequency) als auch dessen seltene Verwendung im gesamten Korpus (Inverse Document Frequency). Durch diese Kombination werden Wörter, die häufig in einem bestimmten Dokument, aber selten im gesamten Korpus vorkommen, höher gewichtet. Dies spiegelt die Bedeutung des Wortes in einem bestimmten Kontext besser wider und ist oft informativer als reine Wortzähler [R23].
In der TF-IDF-Darstellung wird jedes Dokument durch einen Vektor repräsentiert, wobei jede Dimension dieses Vektors einem einzigartigen Wort in dem Korpus entspricht. Im

Gegensatz zur „Bag-of-Words“-Methode, die die Wörter unabhängig von ihrer Häufigkeit im Korpus betrachtet, liefert der Wert an einer Dimension in TF-IDF eine metrische Darstellung der Relevanz des Wortes im jeweiligen Dokument, die durch dessen Häufigkeit und die inverse Dokumenthäufigkeit beeinflusst wird [Lan22, R23].
Trotz seiner Einfachheit hat TF-IDF den Vorteil, dass es semantische Informationen über die Worte in Bezug auf den gesamten Korpus behält. Deshalb wird es häufig in Kombination mit anderen, komplexeren Techniken zur Merkmalsextraktion verwendet. TF-IDF hat sich insbesondere bei Textklassifizierungs- und Clustering-Aufgaben als wirksam erwiesen. Es trägt dazu bei, dass jedes Dokument auf eindeutige Weise durch seinen Vektor repräsentiert wird, was wiederum eine effektive Klassifizierung und Gruppierung der Dokumente ermöglicht. Durch die Betrachtung der Häufigkeit eines Wortes in einem Dokument im Vergleich zu seiner Verbreitung im gesamten Korpus kann TF-IDF auch helfen, Wörter zu identifizieren, die für ein bestimmtes Thema oder einen bestimmten Kontext besonders charakteristisch sind [R23].

```
def information_retrieval(corpus, ngram_range=(1, 2), vector_size=100):
    # TF-IDF representation
    vectorizer = TfidfVectorizer(ngram_range=ngram_range)
    tfidf = vectorizer.fit_transform(corpus)

    # Generate word embeddings
    sentences = [doc.split() for doc in corpus]
    model = Word2Vec(sentences=sentences, vector_size=vector_size,
   epochs=100, min_count=1, seed=42)
    word_embeddings = []
    for doc in sentences:
        vectors = [model.wv[word] for word in doc if word in model.wv]
        if vectors:
            doc_vector = np.mean(vectors, axis=0)
        else:
            doc_vector = np.zeros(vector_size)
        word_embeddings.append(doc_vector)
    word_embeddings = csr_matrix(np.vstack(word_embeddings))

    # Generate ngram features
    vectorizer = CountVectorizer(ngram_range=ngram_range)
    ngrams = vectorizer.fit_transform(corpus)

    # Generate document length feature
    doc_lengths = np.array([len(doc) for doc in corpus]).reshape(-1, 1)
    doc_lengths = csr_matrix(doc_lengths)

    return tfidf, word_embeddings, ngrams, doc_lengths
```

Listing 4.2: TF-IDF

Der oben gegebene Code führt Information Retrieval (IR) auf einem Textkorpus durch. Dabei werden verschiedene Textrepräsentationen und Features erzeugt, die in IR-Analysen

oder -Aufgaben nützlich sind. Zuerst wird die Bedeutung von Wörtern mittels TF-IDF erfasst. Dann werden Word Embeddings generiert, um semantische Beziehungen zwischen Wörtern und Dokumenten festzuhalten. Des Weiteren werden n-Gramm-Features verwendet, um Kontextinformationen zu erfassen. Schließlich wird die Dokumentenlänge als weiteres Merkmal hinzugefügt. Diese verschiedenen Features dienen als Grundlage für weitere Information-Retrieval-Analysen oder -Aufgaben in dieser Arbeit.

4.3.2 Sentiment-Analyse mit VADER

Um die Stimmung in den Textdaten zu bewerten, wird die Sentiment-Analyse-Technik VADER (Valence Aware Dictionary for Sentiment Reasoning) angewendet. VADER ist ein lexikon- und regelbasiertes Sentiment-Analyse-Tool, das speziell für Texte aus sozialen Medien entwickelt wurde. Es ist in der Lage, sowohl die Polarität (positiv, negativ, neutral) als auch die Intensität (Stärke) der Emotion in einem Text zu bewerten. VADER verwendet eine Kombination aus einer Liste von lexikalischen Merkmalen (bekannt als Lexikon), die manuell mit ihrer Polarität (positiv, negativ, neutral) und Intensität bewertet wurden, und einer Reihe von heuristischen Regeln, um die Gesamtstimmung eines Textes zu berechnen. Diese Merkmale werden zusammen mit den BoW- und TF-IDF-Merkmalen zur weiteren Verbesserung der Textrepräsentation genutzt [Sin21].

```
1 def analyze_sentiment(documents):
2
3     nltk.download(['vader_lexicon'], quiet=True)
4     sia = SentimentIntensityAnalyzer()
5     sentiment_scores = [sia.polarity_scores(text)['compound'] for text
      in documents]
6     return sentiment_scores
```

Listing 4.3: Sentiment-Analyse mit VADER

Dieser Codeausschnitt führt eine Sentimentanalyse Mithilfe des VADER-Lexikons aus der NLTK-Bibliothek und werden Sentimentbewertungen für jedes Dokument berechnet. die Stimmung jedes Dokument wird repräsentiert, ob positiv, nigativ oder neutral durch Sentimentscores, die als Ergebnis der Berechnung der Sentimentbewertungen entstehen. Die Funktion verwendet das VADER-SentimentIntensityAnalyzer-Werkzeug, um die Sentiment-Scores basierend auf dem Wortgebrauch und der Satzstruktur in den Dokumenten zu ermitteln.

4.3.3 BERT-Word-Embeddings

Das letzte Element in der Merkmalsextraktion waren die BERT-Word-Embeddings. Word Embeddings sind eine Art von Wortdarstellungen, die komplexe semantische Beziehungen zwischen Wörtern einfangen können. Sie stellen Wörter als dichte Vektoren dar, bei denen ähnliche Wörter nahe beieinander im Vektorraum liegen.

Für dieses Projekt wurden BERT-Embeddings mit der Transformers-Bibliothek von Hugging Face erzeugt. Diese Embeddings wurden dann zusammen mit den BoW-, TF-IDF- und Sentiment-Merkmalen verwendet, um eine umfassende Darstellung der Textdaten zu erstellen, die als Eingabe für den Clustering-Algorithmus dient [McC19].
Gemäß dieser Struktur wird eine Vertiefung in die detaillierten Aspekte und Feinheiten jedes Schrittes vorgenommen, wobei gleichzeitig der Projektfortschritt dargelegt wird.

```
class BertEmbedding:
    def __init__(self, model_name='bert-base-uncased'):
        self.tokenizer = BertTokenizer.from_pretrained(model_name)
        self.model = BertModel.from_pretrained(model_name)

    def get_embeddings(self, documents):
        embeddings = []
        for text in documents:
            # Tokenize the text
            inputs = self.tokenizer(text, return_tensors='pt',
   truncation=True, padding=True)

            # Get the output from the BERT model
            with torch.no_grad():
                outputs = self.model(**inputs)

            # The embedding for the text is the last hidden state of the
    BERT model
            last_hidden_state = outputs.last_hidden_state

            # To get a single vector for the entire text, we take the
   mean of all token vectors
            text_embedding = last_hidden_state.mean(dim=1).squeeze()

            # Convert the torch.Tensor to a numpy array
            text_embedding_np = text_embedding.numpy()

            embeddings.append(text_embedding_np)

        # Stack all embeddings along a new axis to create a 2D numpy
   array
        return np.stack(embeddings)
```

Listing 4.4: BERT-Word-Embeddings

Dieser Code definiert eine Klasse namens „BertEmbedding“. Die Klasse bietet eine Methode „get_embeddings“, um Word-Embedding zu erstellen. Während dieser Erstellung wird der Text mithilfe des BERT-Modells in numerische Vektoren umgewandelt. Die Methode erzeugt eine 2D-Numpy-Matrix, in der jede Zeile ein Embedding für ein Dokument darstellt. Dies ermöglicht die Nutzung von BERT-Textrepräsentationen.

4.4 FCK-Means-Clustering-Algorithmus

Die Hauptkomponente des Verfahrens liegt in der Gruppierung von Textdokumenten durch Clustering. Der Clustering-Prozess organisiert ähnliche Dokumente in Gruppen oder „Cluster“ auf der Grundlage ihrer Merkmale. In diesem Projekt wird den FCK-Means-Clustering-Algorithmus implementiert, der für Textdaten optimiert wurde.

Um den FCK-Means Clustering Algorithmus anzuwenden, werden zunächst die Clusterzentren initialisiert.

```
def _initialize_centers(data, fixed_centers, n_clusters, metric='euclidean'):

    # Run standard KMeans++ to initialize
    kmeans = KMeans(n_clusters=n_clusters, init='k-means++', n_init=10).fit(data)
    centers = kmeans.cluster_centers_

    # If there are no fixed centers, return the results of the normal clustering approach.
    if len(fixed_centers) == 0:
        return centers

    # Calculate the average distance to all fixed centers for each non-fixed center
    distances = np.array([calculate_average_distance(center, data[fixed_centers], metric) for center in centers])

    # Select the k-m non-fixed centers with the maximum distance
    # Step 1: Sort the distances and get their indices
    sorted_indices = np.argsort(distances)

    # Step 2: Define the number of non-fixed centers we need
    num_non_fixed_centers = n_clusters - len(fixed_centers)

    # Step 3: Select the indices for the non-fixed centers from the sorted array
    non_fixed_centers_indices = sorted_indices[-num_non_fixed_centers:][::-1]

    non_fixed_centers = centers[non_fixed_centers_indices]

    return np.vstack((data[fixed_centers], non_fixed_centers))
```

Listing 4.5: Initialisierung der Clusterzentren

Dieser oben gegebene Code implementiert die Initialisierung der Clusterzentren für den FCK-Means Clustering-Algorithmus. Der Algorithmus verwendet eine Kombination aus dem K-Means++-Initialisierungsverfahren und einer Anpassung, um die Clusterzentren

zu finden. Zuerst wird das Standard-K-Means++-Verfahren angewendet, um eine erste Gruppe von Clusterzentren zu finden. Falls fixierte Zentren vorhanden sind, werden die Durchschnittsdistanzen von den nicht-festgelegten Zentren zu den festgelegten Zentren berechnet. Die nicht-festgelegten Zentren mit den größten Durchschnittsdistanzen zu den festgelegten Zentren werden ausgewählt. Die ausgewählten Zentren, sowohl die festgelegten als auch die nicht-festgelegten, werden zurückgegeben.

```
def fck_means_clustering(data, n_clusters, fixed_centers, metric='euclidean', max_iter=300, tol=1e-4):
    # Check if data is not already a numpy array
    if not isinstance(data, np.ndarray):
        data = data.toarray()

    # Initialize centers
    centers = _initialize_centers(data, fixed_centers, n_clusters, metric=metric)

    # If there are no fixed centers, assign the points to the normal clustering centers.
    if not len(fixed_centers):
        labels, _ = pairwise_distances_argmin_min(data, centers, metric=metric)
        return labels, centers

    labels = None
    for _ in range(max_iter):
        # Assign each sample to the closest center
        labels, _ = pairwise_distances_argmin_min(data, centers, metric=metric)

        # Calculate new centers as the mean of the samples in each cluster
        new_centers = np.array([data[labels == i].mean(axis=0) for i in range(n_clusters)])

        # Keep the fixed centers
        new_centers[:len(fixed_centers)] = data[fixed_centers]

        # Check for convergence
        if np.all(np.abs(new_centers[len(fixed_centers):] - centers[len(fixed_centers):]) <= tol):
            break
        plot_intermediate_clusters(data, labels, centers, new_centers, data[fixed_centers])

        centers = new_centers

    return labels, centers
```

Listing 4.6: FCK-Means-Clustering-Algorithmus

Dieser Codeausschnitt implementiert eine Version des K-Means-Clustering-Algorithmus, deren Aufgabe ist, feste Zentren im Datensatz zu berücksichtigen. Es führt eine Clusteranalyse auf gegebenen Daten durch, indem es die Datenpunkte in vordefinierte Anzahl von Clustern gruppiert. Der Algorithmus ermöglicht Festlegung einiger Clusterzentren (falls gewünscht), während er die restlichen Zentren anhand der Daten durch die Anwendung von der euklidischen Distanz berechnet. Dabei werden die Datenpunkte in Clustern gruppiert, und die Zentren werden schrittweise angepasst, bis ein Konvergenzkriterium erreicht ist. Die zurückgegebenen Ergebnisse sind die zugewiesenen Cluster-Labels für jeden Datenpunkt und die berechneten Zentren der Cluster.

Der unten aufgeführten Code führt eine umfassende Analyse des Clustering-Prozesses auf Textdokumenten durch, indem der FCK-Means-Algorithmus eingesetzt wird. Dieser Prozess umfasst verschiedene Schritte. Zu Beginn werden Daten aus zwei unterschiedlichen Quellen (rohe und fixierte Dokumente) geladen und vorverarbeitet. Anschließend erfolgt das Kombinieren dieser Dokumente aus beiden Quellen zu einem umfassenden Datensatz. Als nächstes werden Merkmale, repräsentiert durch Feature-Vektoren, aus dem Textkorpus extrahiert. Hierbei kommt optional die BERT-Verarbeitung zum Einsatz, um eine tiefergehende Textrepräsentation zu erzielen. Die Kernkomponente des Prozesses besteht im Anwenden des FCK-Means-Clustering-Algorithmus auf die extrahierten Merkmale. Dabei werden sowohl unbearbeitete als auch fixierte Clusterzentren berücksichtigt. Die Qualität der erzeugten Cluster wird mithilfe des Silhouetten-Scores bewertet und ausgegeben. Die Ergebnisse der Clusterung, inklusive der Dokumente und den Zuordnungen zu den Clustern, werden gespeichert. Zusätzlich erfolgt eine Visualisierung der Clusterergebnisse durch Diagramme, die helfen, die Struktur und Verteilung der Cluster zu verstehen. Insgesamt ermöglicht dieser Code eine hochwirksame Gruppierung von Textdokumenten unter Verwendung des FCK-Means-Algorithmus und bietet umfangreiche Möglichkeiten zur Analyse sowie zur ansprechenden Visualisierung der erhaltenen Ergebnisse.

```
1  import argparse
2  import numpy as np
3  from sklearn.metrics import silhouette_score
4
5  from src.utils.data_processing import load_and_preprocess_data,
       save_file, plot_clusters, plot_labels
6  from src.feature_extraction import get_vector_representation
7  from src.fck_means import fck_means_clustering
8
9  def main(k_clusters=5):
10     # Load and preprocess data
11     raw_documents, raw_corpus = load_and_preprocess_data('./data/raw/')
12     fixed_documents, fixed_corpus = load_and_preprocess_data('./data/
       fixed/')
13     documents = fixed_documents + raw_documents
14     corpus = fixed_corpus + raw_corpus
15
```

```
    fixed_docs_len = len(fixed_documents)

    if k_clusters < fixed_docs_len:
        print(
            f"Error: Number of clusters ({k_clusters}) cannot be less than the number of fixed centers ({fixed_docs_len})")
        print("Press any key to exit.")
        input()
        return

    print(f"{fixed_docs_len} fixed centers were detected.")

    combined_features = get_vector_representation(corpus, use_bert=0)
    # Run FCK-Means clustering
    fixed_centers = np.arange(fixed_docs_len)
    labels, centers = fck_means_clustering(combined_features, k_clusters, fixed_centers, metric='minkowski')
    # Compute and print the silhouette score
    silhouette_avg = silhouette_score(combined_features, labels)
    title = f"Silhouette Score: {silhouette_avg:.2f}, Fixed centers count: {fixed_docs_len}"

    plot_labels(labels, 'FCK-Means CLustering Results')
    # Save the clustering results
    save_file(documents, corpus, labels)
    # Plot and save the cluster results
    plot_clusters(combined_features, labels, centers, 'output/', title, fixed_centers)

    print("Clustering results have been saved to the 'output/' directory.")
```

Listing 4.7: Generierung der Ergebnisse

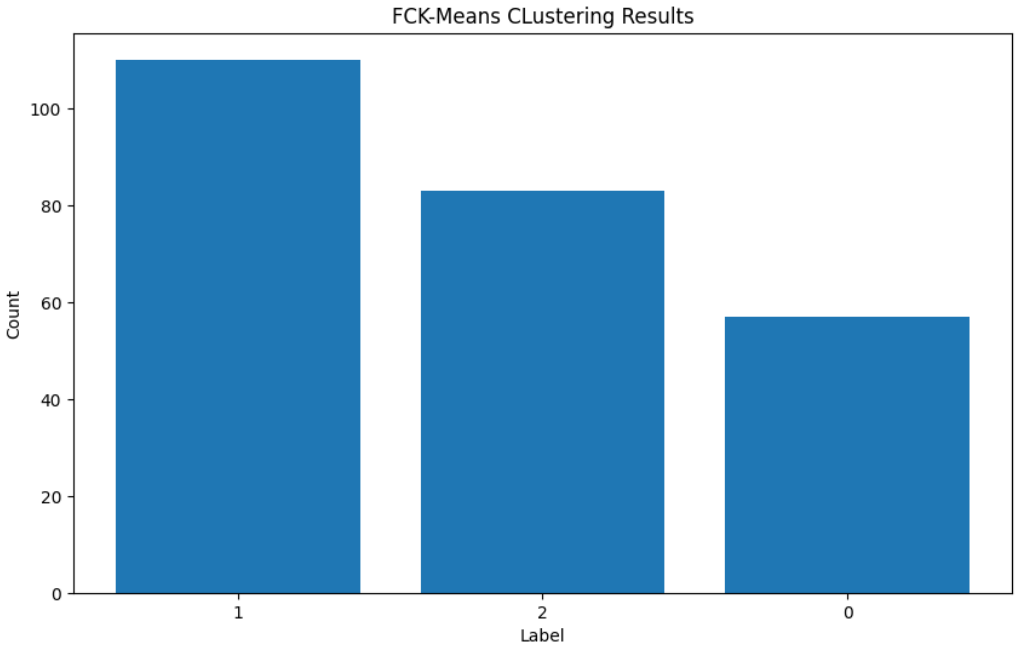


Abbildung 4.1: Ergebnisse des FCK-Means-Clustering

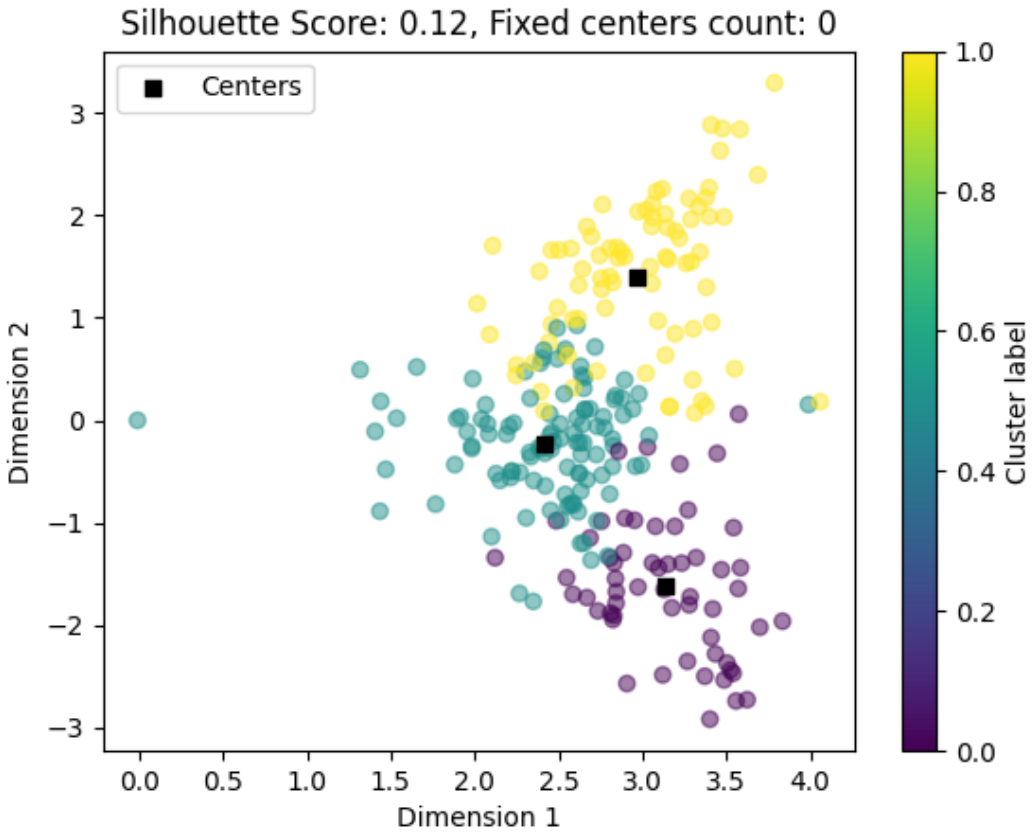
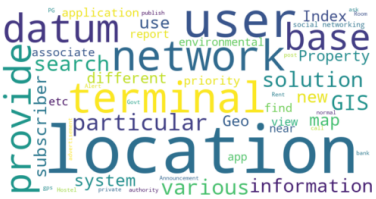
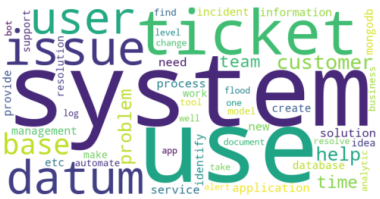


Abbildung 4.2: Clustering ohne fixierte Mittelpunkte

Die Abbildung 4.1 zeigt die ersten Ergebnisse von FCK-Means-Clustering ohne fixierte Mittelpunkte. Da stehen drei Clustern, die die Ergebnisse der Bearbeitung der Textuellen Eingaben enthalten, wobei jedes Cluster die miteinander ähnlichsten Dokumente kategorisiert. Der zweite Cluster, der Nummer 1 trägt, hat den größten Wert von den Dokumenten, die zueinander ähnlich sind. Der dritte Cluster hat einen wenigeren Wert als der vorherigen, da die Menge der darin enthaltenen Dokumente kleiner ist als die Menge der Dokumente, die bei dem zweiten Cluster sind. Der erste Cluster, der die Nummer 0 trägt, hat den wenigsten Wert der eingegebenen Dokumente, weil diese Dokumente sehr unterschiedlich von den Dokumenten sind, die in den anderen Clustern kategorisiert sind. Die Abbildung 4.2 zeigt die Initialisierung der Zenter aller Clustern und deren Punkten auf eine zweidimensionale Karte, wobei jedes Cluster mit seiner Farbe und seiner Repräsentation von den anderen Clustern unterschieden werden kann.



(a) Wordcloud Cluster 0



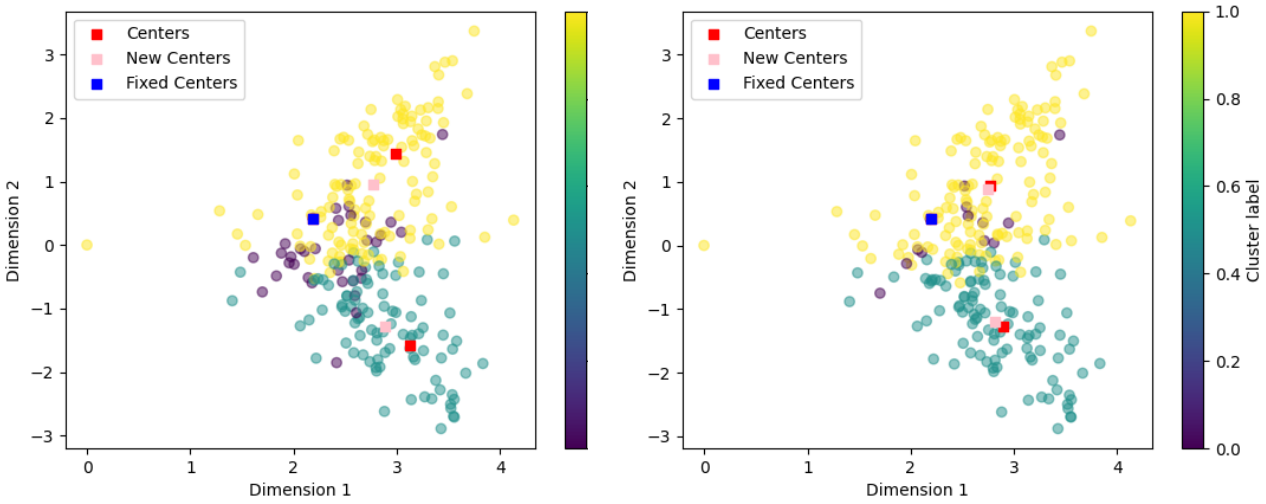
(b) Wordcloud Cluster 1

(c) Wordcloud Cluster 2

Abbildung 4.3: Wordclouds der erstellten Cluster

Die Abbildung 4.3 zeigt drei Wordclouds, die die am meisten in jedem Cluster vorkommenden Wörter zeigen. Jede Wordcloud zeigt verschiedene Wörter mit unterschiedlichen Größen, wobei die Größe des Wortes, ob kleiner oder größer ist, von der Häufigkeit des Wortes im Cluster abhängt. Daraus wird abgeleitet, was der allgemeine Kontext der Themen der Dokumente in jedem Cluster ist.

Nach der Durchführung des FCK-Means-Clustering-Prozesses ohne die Anwendung von fixierten Mittelpunkten und unter Berücksichtigung der daraus resultierenden Ergebnisse, wie sie in den Abbildungen 4.1 und 4.2 veranschaulicht werden, wird eine anschließende Ausführung des FCK-Means-Clustering-Prozesses mit fixierten Mittelpunkten vorgenommen. Die resultierenden Resultate dieses Vorgehens werden durch die Abbildungen 4.4, 4.5, 4.6 und 4.7 dargestellt. Diese Abbildungen visualisieren die fortlaufende Anpassung der Darstellung der Cluster nach der Festlegung der fixierten Mittelpunkte. Diese kontinuierliche Anpassung wird durch Veränderungen in den Positionen der Clusterzentren deutlich. Die Neuberechnung der Clusterzentren erfolgt durch die Bildung eines Durchschnitts der Datenpunkte in jedem einzelnen Cluster. Dabei bleibt das festgesetzte Dokument als Clusterzentrum unverändert, da es fixiert ist. Dieser Prozess der Neuberechnung wiederholt sich solange, bis eine Konvergenz erreicht ist oder eine vordefinierte Anzahl von Iterationen durchgeführt wurde. In jeder Iteration werden die Zuordnungen der Datenpunkte zu den Zentren überprüft und die Clusterzentren entsprechend aktualisiert. Nach Erreichen der Konvergenz oder dem Abschluss der maximalen Anzahl von Iterationen werden die erzielten Resultate analysiert. Dies schließt die Evaluierung der Zuordnungen der Datenpunkte zu den Clustern sowie die Ermittlung der endgültigen Clusterzentren ein. Die Clusterergebnisse werden sodann visualisiert, um die Güte der Clusterbildung zu beurteilen. Diese Visualisierung kann durch die Darstellung der Dokumente innerhalb der Cluster oder mithilfe von Metriken wie beispielsweise dem Silhouetten-Score erfolgen.



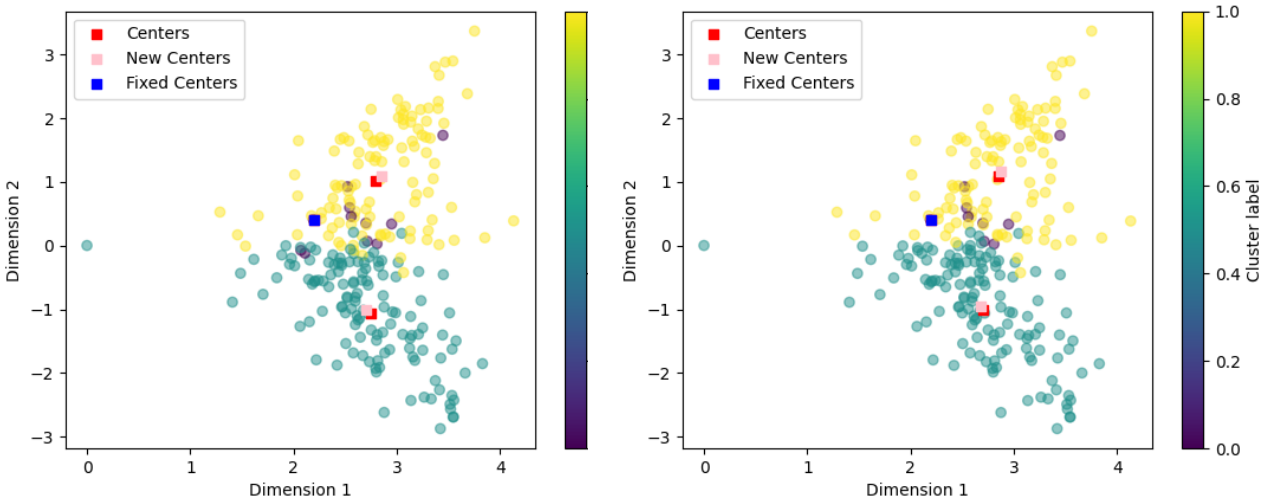
(a) Clustering Ausführung N1

(b) Clustering Ausführung N2

(c) Clustering Ausführung N3

(d) Clustering Ausführung N4

Abbildung 4.4: Ablauf der Clustering Ausführung 1-4



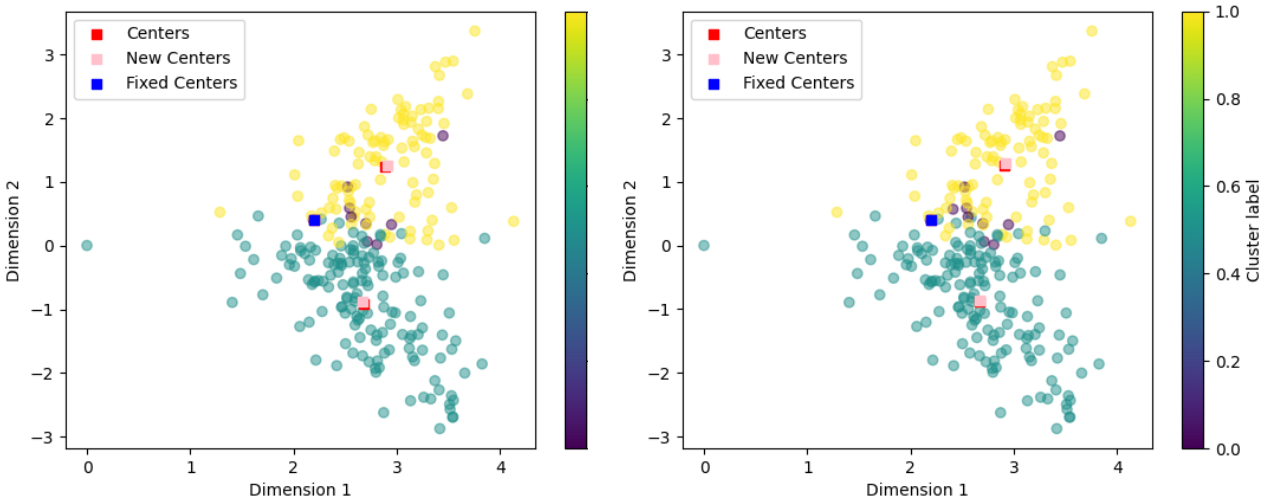
(a) Clustering Ausführung N5

(b) Clustering Ausführung N6

(c) Clustering Ausführung N7

(d) Clustering Ausführung N8

Abbildung 4.5: Ablauf der Clustering Ausführung 5-8



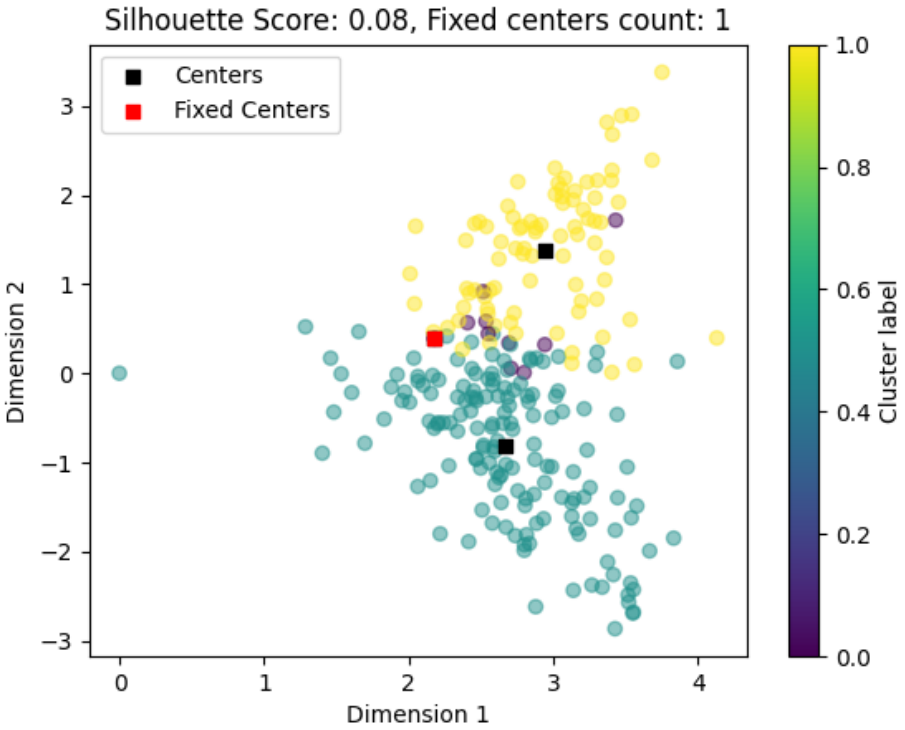
(a) Clustering Ausführung N9

(b) Clustering Ausführung N10

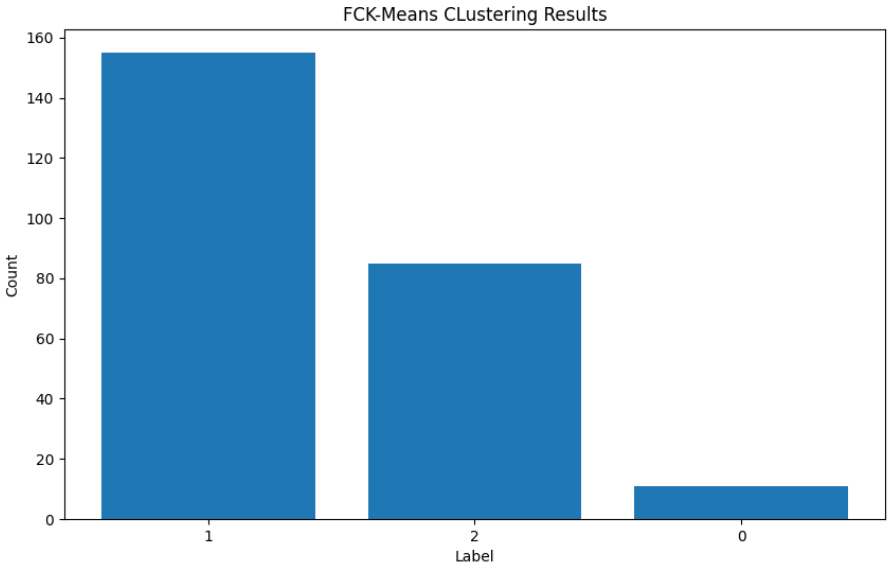
(c) Clustering Ausführung N11

(d) Clustering Ausführung N12

Abbildung 4.6: Ablauf der Clustering Ausführung 9-12



(a) Clustering Ausführung N13



(b) Ergebnisse des FCK-Means-Clustering mit einem fixierten Mittelpunkt

Abbildung 4.7: Das Ende des Clustering-Prozesses

5 Evaluation

Die Evaluation dieses entwickelten Systems zur k-Means-Clusteranalyse von Textdokumenten mit teilweise fixierten Clustermittelpunkten erfolgte umfassend und beinhaltete verschiedene Aspekte. Das Ziel dieser Evaluation ist es, die Leistungsfähigkeit des Systems in verschiedenen Szenarien zu bewerten und seine Fähigkeit zur skalierbaren Verarbeitung großer Datensätze zu überprüfen. Zunächst werden zwei verschiedene Varianten des Systems getestet: eine, die Clustering ohne fixierte Zentren durchführt, und eine andere, die teilweise fixierte Clustermittelpunkte einbezieht. Dieser Ansatz ermöglichte eine vergleichende Analyse der Auswirkungen der Fixierung von Clustermittelpunkten auf die Clusterqualität. Für die Distanzberechnungen zwischen den Datenpunkten und den Clustermittelpunkten werden mehreren Varianten der Distanzmetriken erstellt und davon die besten zwei verwendet: die euklidische Distanz und die Minkowski-Distanz. Dies erlaubte eine Bewertung der Auswirkungen verschiedener Distanzmetriken auf die Clustering-Ergebnisse. Der Prozess beginnt mit einer umfassenden Analyse der Systemleistung auf einem Testdatensatz von 256 Textdokumenten. Hierbei wurden Aspekte wie die Clusterqualität, die Anzahl der benötigten Iterationen für die Konvergenz und die Stabilität der Ergebnisse untersucht. Ein wesentlicher Teil der Evaluation besteht darin, das System auf einem wesentlich größeren Datensatz [1] von 20.000 Textdokumenten aus 20 verschiedenen Datennewsgroups zu testen. Dieses Szenario ziehlt darauf ab, die Skalierbarkeit des Systems unter realistischeren Bedingungen zu überprüfen. Die Ergebnisse der Evaluation werden anhand von Metriken wie dem Silhouetten-Score bewertet, um die Güte der erzeugten Cluster zu quantifizieren (siehe Abbildungen 5.1, 5.2, 5.3, 5.4, 5.5). Die Analyse der Clusterqualität erfolgt unter Berücksichtigung verschiedener Parameter wie der Anzahl der Cluster, der gewählten Metrik und der Fixierung von Clustermittelpunkten. Insgesamt zeigt die Evaluation eine erfolgreiche Anwendung des entwickelten Systems zur Clusteranalyse von Textdokumenten. Die Integration teilweise fixierter Clustermittelpunkte erweis sich als nützlich, um bestimmte Aspekte der Daten zu betonen. Die Auswahl der Distanzmetrik und die Skalierbarkeit des Systems erwiesen sich als zufriedenstellend, insbesondere bei der Verarbeitung großer Datensätze. Die Ergebnisse tragen zur Erkenntnis bei, wie das System in praktischen Anwendungsfällen eingesetzt werden kann und welche Faktoren die Qualität der erzeugten Cluster beeinflussen können.

[1] http://qwone.com/ jason/20Newsgroups/

5.1 Bewertungsmethode: Silhouette Score

Um die Effektivität des FCK-Means Clustering-Ansatzes zu bewerten, wird den Silhouette Score [2] als Bewertungsmetrik ausgewählt. Der Silhouette Score ist eine gängige Metrik, die zur Beurteilung der Qualität von Clustering-Ergebnissen verwendet wird. Es ist ein effektives Mittel zur Quantifizierung, wie gut ein Dokument zu seinem zugewiesenen Cluster passt im Vergleich zu anderen Clustern. Der Silhouette Score eines einzelnen Dokuments wird berechnet, indem man den durchschnittlichen Abstand zwischen diesem Dokument und allen anderen Dokumenten im selben Cluster (auch als Kohäsionsabstand bezeichnet) vom durchschnittlichen Abstand zwischen diesem Dokument und allen Dokumenten im nächsten Cluster (auch als Separationsabstand bezeichnet) subtrahiert und dann das Ergebnis durch den größeren der beiden Abstände teilt. Ein hoher Silhouette Score bedeutet, dass die Kohäsion hoch und die Separation hoch ist, was ein gutes Clustering darstellt [Bha21]. Auf der anderen Seite bedeutet ein niedriger Silhouette Score, dass die Dokumente in einem Cluster nicht gut zu diesem Cluster passen und möglicherweise besser zu einem anderen Cluster passen würden. Ein Silhouette Score von 0 deutet darauf hin, dass das Dokument auf der Grenze zwischen zwei Clustern liegt. Der Gesamtsilhouette Score für ein Clustering-Ergebnis ist der Durchschnitt der Silhouette Scores aller Dokumente. In dieser Arbeit wird der Gesamtsilhouette Score verwendet, um die Qualität der von FCK-Means-Clustering-Ansatz erzeugten Clustering-Ergebnisse zu beurteilen. Es ist wichtig zu beachten, dass der Silhouette Score nicht der einzige Weg ist. Dennoch bietet der Silhouette Score eine nützliche quantitative Bewertung der Clustering-Qualität, die es ermöglicht, die Leistung dieses Ansatzes auf objektive Weise zu beurteilen [Bha21].

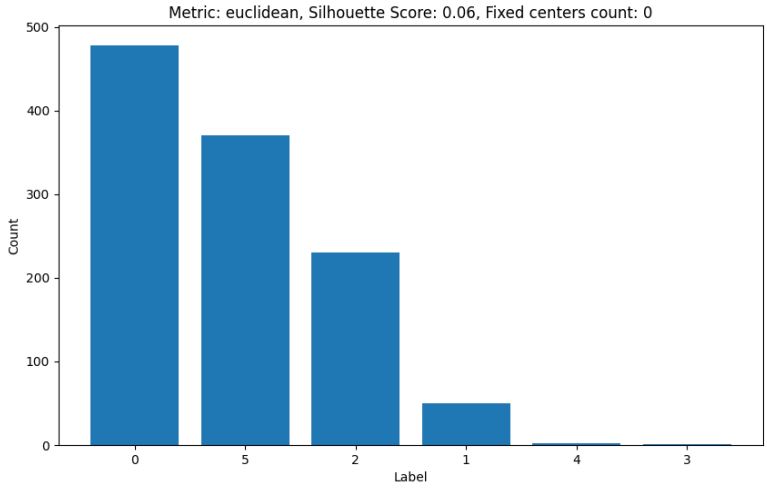
5.2 Testdatensatz: 20 Newsgroups

Für die Evaluation von dem FCK-Means-Clustering-Ansatz wird der 20 Newsgroups-Datensatz [3] verwendet. Der „20 Newsgroups-Datensatz“ ist ein bekannter Datensatz für Textklassifizierungsaufgaben, der Beiträge aus 20 verschiedenen Usenet-Newsgroups enthält. Jeder Beitrag in diesem Datensatz ist ein unabhängiges Dokument, das einer der 20 Newsgroups zugeordnet ist. Die Newsgroups decken ein breites Spektrum an Themen ab, von Religion und Politik bis hin zu Sport und Wissenschaft. Dies macht den Datensatz ideal für die Zwecke des Systems, da er eine repräsentative Auswahl von Themen und Schreibstilen bietet, die man in einer realen Welt Sammlung von Textdokumenten erwarten würde. Um die Anwendbarkeit des FCK-Means-Clustering-Ansatzes auf Szenarien mit einer kleineren Anzahl von Kategorien zu testen, steht zur Verfügung den „20 Newsgroups-Datensatz“ in sechs übergeordnete Kategorien eingeteilt. Diese Kategorien wurden basierend auf thematischen Ähnlichkeiten zwischen den ursprünglichen 20 Newsgroups erstellt. Die sechs übergeordneten Kategorien sind:

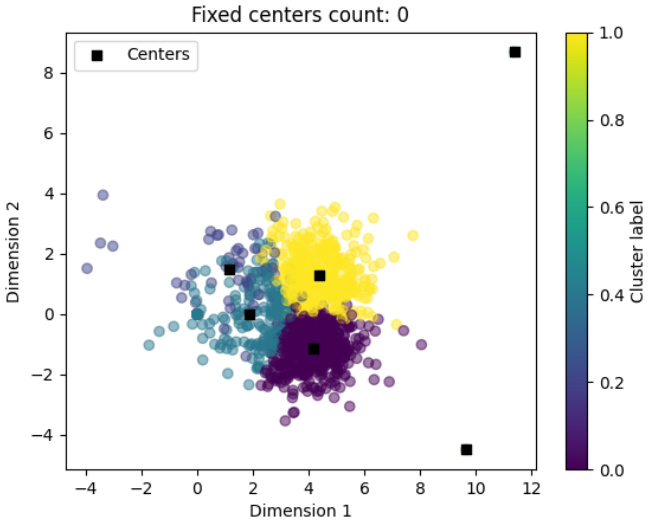
[2] https://scikitlearn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html
[3] http://qwone.com/ jason/20Newsgroups/

1. Technik: Beiträge aus Newsgroups, die sich auf technische Themen konzentrieren.
2. Politik: Beiträge aus Newsgroups, die sich auf politische Diskussionen und Debatten konzentrieren.
3. Religion: Beiträge aus Newsgroups, die sich auf religiöse Themen und Glaubenssysteme konzentrieren.
4. Sport: Beiträge aus Newsgroups, die sich auf verschiedene Sportarten und sportbezogene Themen konzentrieren.
5. Wissenschaft: Beiträge aus Newsgroups, die sich auf wissenschaftliche Themen und Forschung konzentrieren.
6. Sonstiges: Beiträge aus Newsgroups, die nicht in eine der anderen Kategorien fallen.

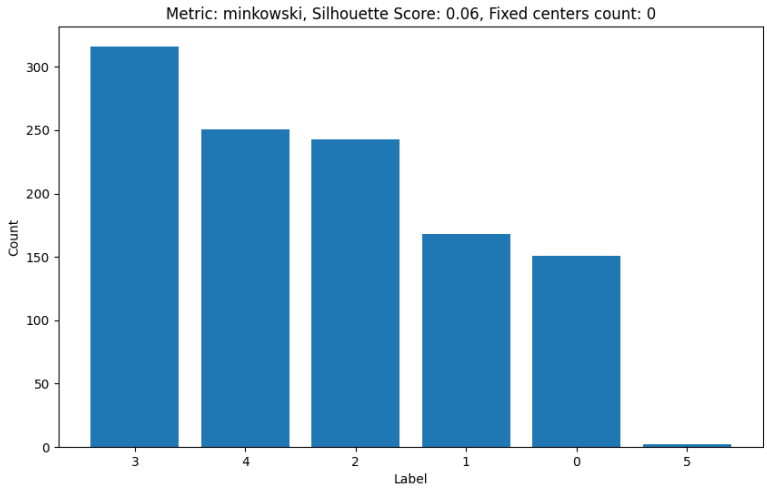
Dieser Ansatz ermöglicht es, die Leistung des FCK-Means-Clustering-Ansatzes bei der Gruppierung von Dokumenten in eine relativ kleine Anzahl von Clustern zu bewerten, die thematisch distinkt und gut definiert sind. Dabei wird der Silhouette Score verwendet, um die Qualität der resultierenden Clustering-Ergebnisse zu quantifizieren.



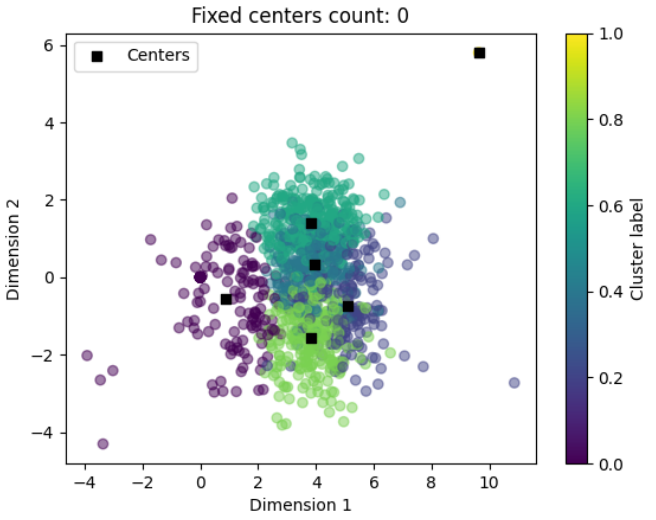
(a)



(b)

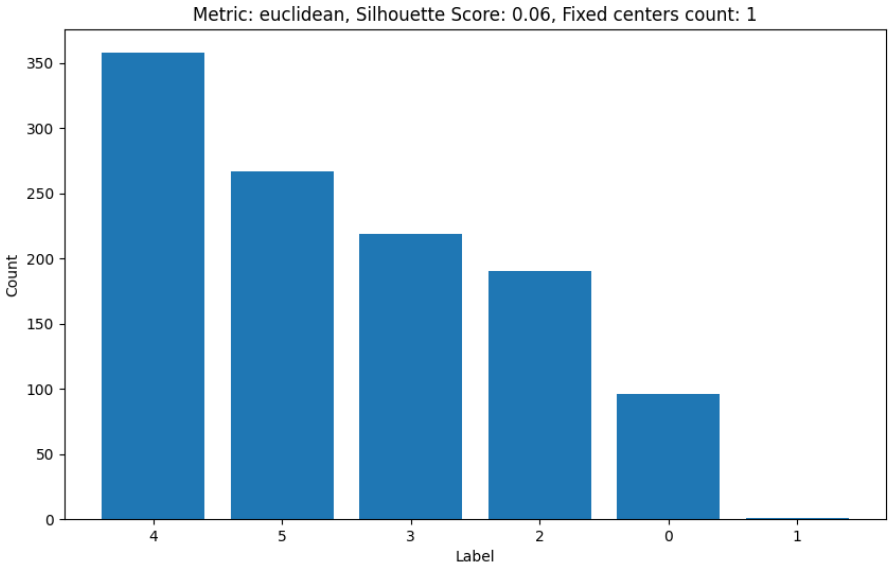


(c)

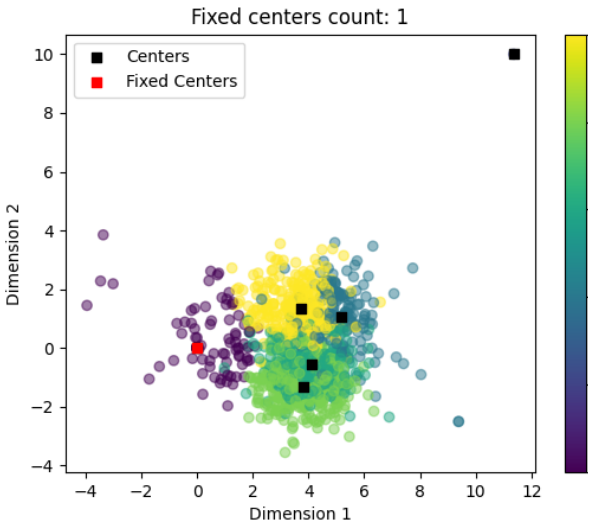


(d)

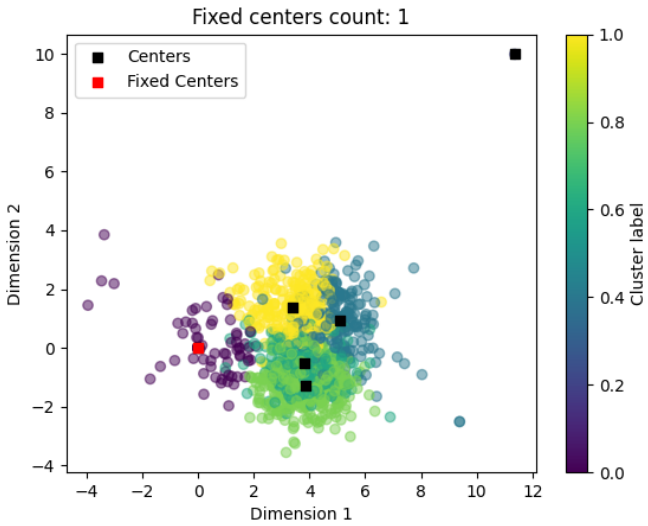
Abbildung 5.1: Evaluation von Clustering basierend auf euklidischer und Minkowski-Distanz ohne festgelegte Mittelpunkte.



(a)

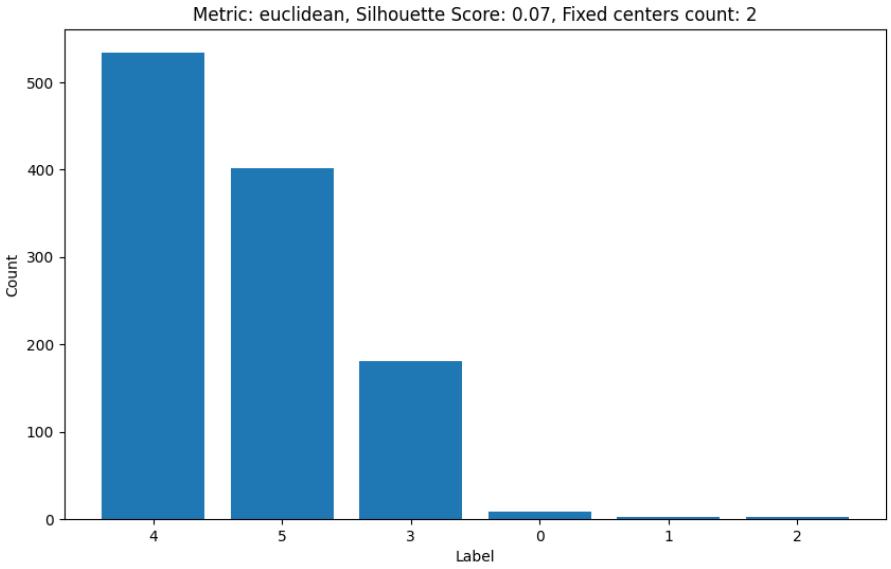


(b) Anfang der Ausführung

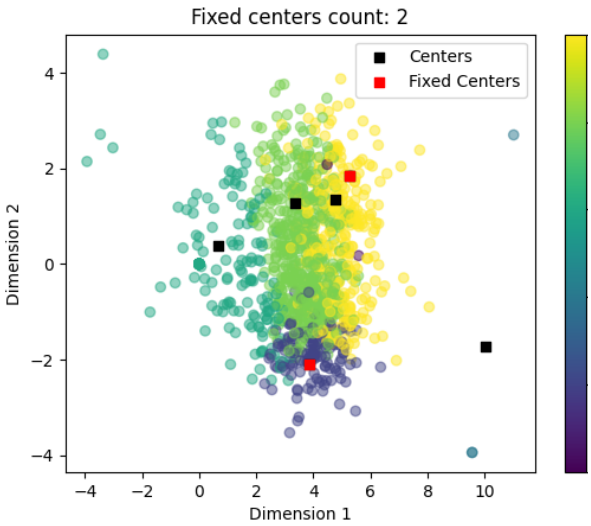


(c) Ende der Ausführung

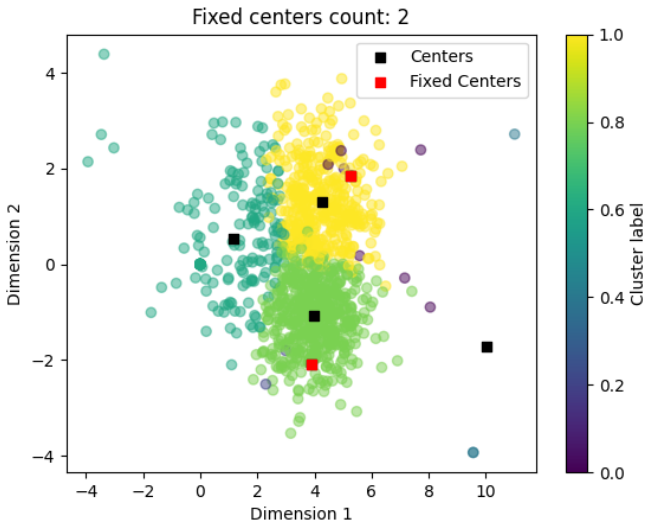
Abbildung 5.2: Evaluation von Clustering basierend auf euklidischer Distanz mit einem festgelegten Mittelpunkt.



(a)

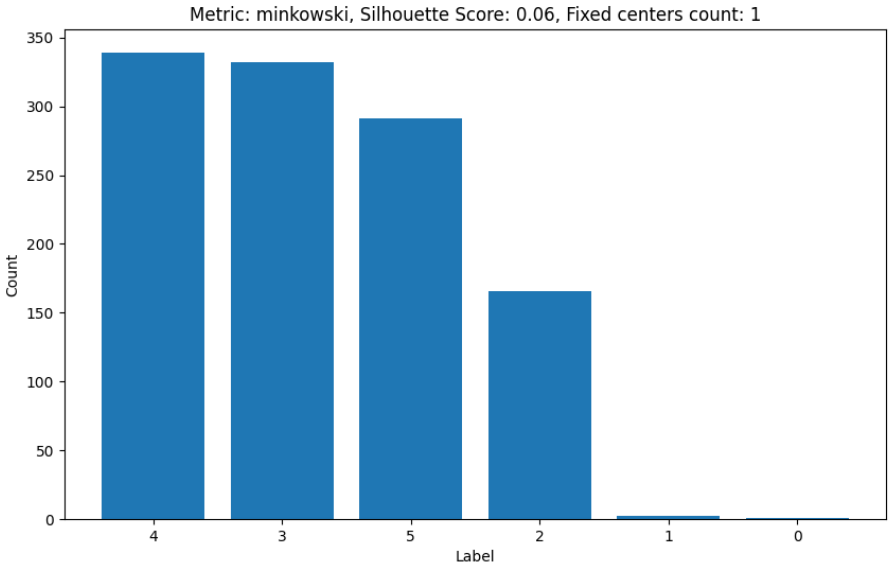


(b) Anfang der Ausführung

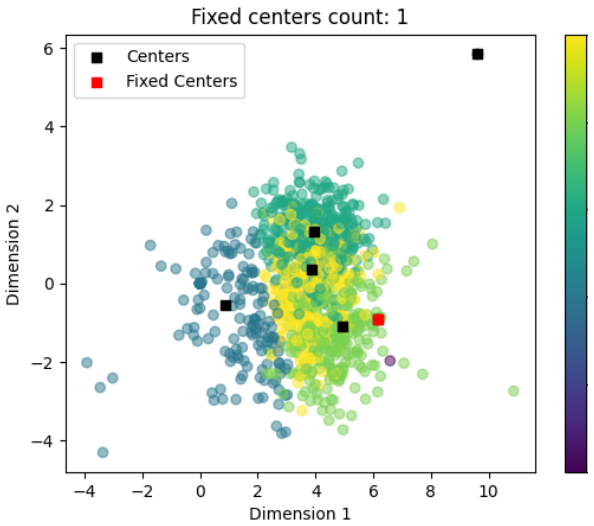


(c) Ende der Ausführung

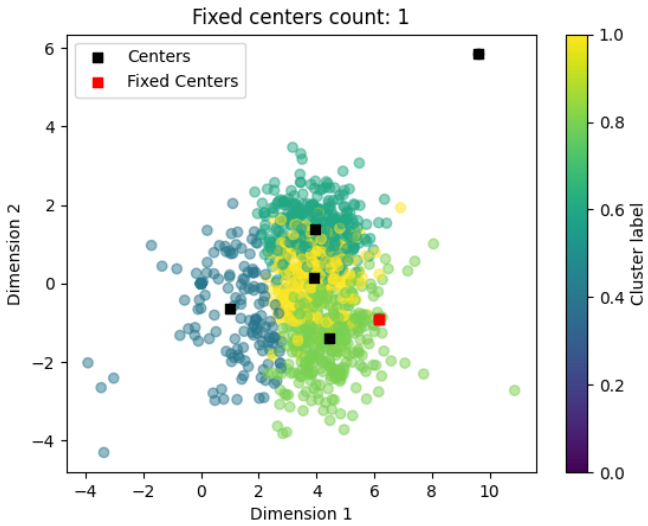
Abbildung 5.3: Evaluation von Clustering basierend auf euklidischer Distanz mit zwei festgelegten Mittelpunkten.



(a)

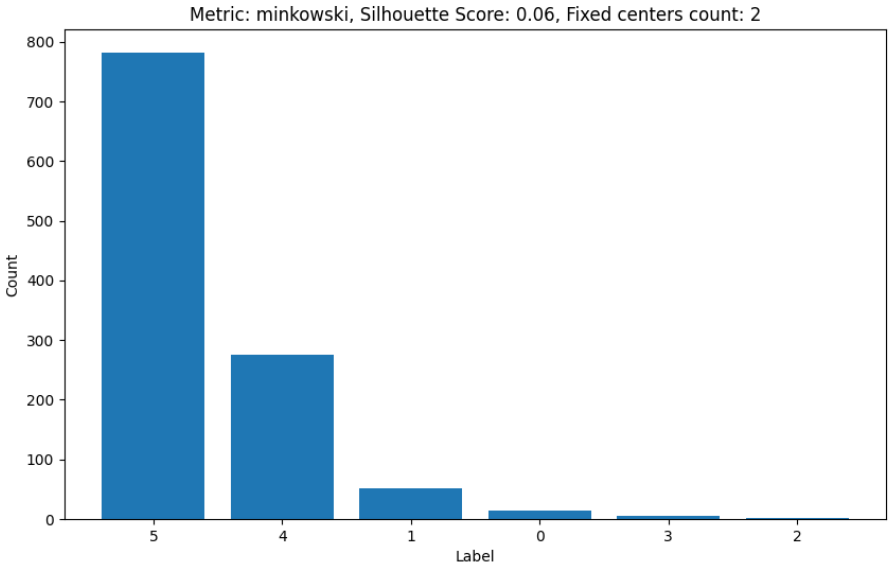


(b) Anfang der Ausführung

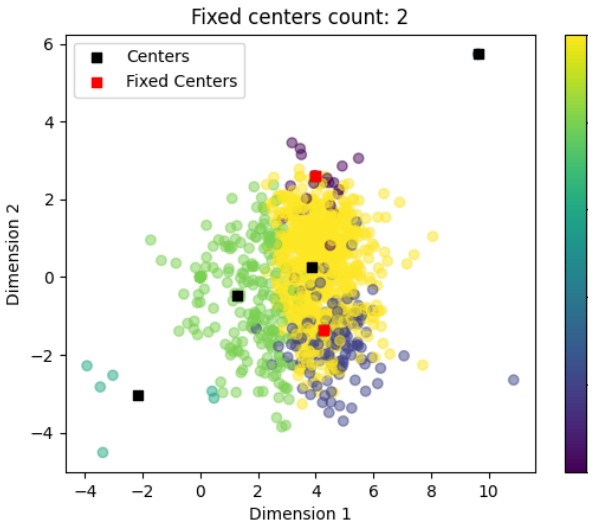


(c) Ende der Ausführung

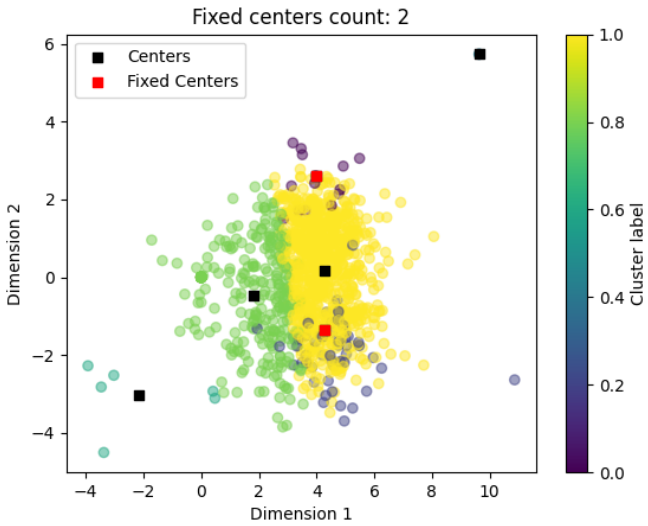
Abbildung 5.4: Evaluation von Clustering basierend auf Minkowski-Distanz mit einem festgelegten Mittelpunkt.



(a)



(b) Anfang der Ausführung



(c) Ende der Ausführung

Abbildung 5.5: Evaluation von Clustering basierend auf Minkowski-Distanz mit zwei festgelegten Mittelpunkten.

Tabelle 5.1: Silhouette-Scores für unterschiedliche P-Werte, Metriken und diverse Szenarien

Silhouette-Score		**Anzahl der Fixierten Mittelpunkte**		
P-Wert	**Metrik**	**0**	**1**	**2**
	Euklidische Distanz	0.0681	0.0580	0.0599
1	Minkowski-Distanz	0.0598	0.0620	0.0672
2	Minkowski-Distanz	0.0558	0.0861	0.0703
3	Minkowski-Distanz	0.0709	0.0672	0.0481
4	Minkowski-Distanz	0.0620	0.0870	0.0707

Die vorgestellte Tabelle 5.1 präsentiert Silhouette-Score-Werte für verschiedene Kombinationen von Metriken und Clustering-Szenarien, wobei P-Werte von 1 bis 4 für die Minkowski-Distanz verwendet wurden. Der Silhouette-Score bewertet die Clusterqualität anhand der inneren Datenpunktabstände und der Distanz zwischen Clustern. Höhere Werte zeigen eine bessere Clusterung an. Die Tabelle zeigt diese Werte für die euklidische und Minkowski-Distanz sowie für unterschiedliche Anzahlen von festen Mittelpunkten in den Clustern. Sie ermöglicht den Vergleich der Effizienz des Clusterings in verschiedenen Einstellungen. Der Silhouette-Score bewertet definitiv die Qualität von Clustering-Verfahren, da er eine quantitative Metrik zur Bewertung der Effizienz der Clusterung darstellt. In der präsentierten Tabelle, wo sowohl die euklidische als auch die Minkowski-Distanz verwendet wurden, zeigen die nah beieinander liegenden Werte, dass die Wahl der Metrik einen Einfluss auf die Clusterqualität haben kann. Die Tatsache, dass die Silhouette-Score-Werte ähnlich sind, könnte darauf hinweisen, dass in diesen spezifischen Daten und Clustering-Szenarien die beiden Metriken vergleichbare Ergebnisse liefern. Das bedeutet, dass sowohl die euklidische als auch die Minkowski-Distanz ähnlich gut bei der Trennung der Datenpunkte in Cluster funktionieren. Dennoch ist es wichtig zu beachten, dass trotz ähnlicher Ergebnisse die tatsächliche Verteilung der Daten und die Struktur der Cluster eine Rolle spielen können. Es ist auch erwähnenswert, dass die Wahl des P-Werts in der Minkowski-Distanz signifikante Unterschiede in den Silhouette-Score-Werten ergab. Insbesondere bei P=2, was der euklidischen Distanz entspricht, konnten höhere Silhouette-Scores erzielt werden. Dies zeigt, dass die Anpassung des P-Werts eine entscheidende Rolle bei der Auswahl der geeigneten Metrik für ein bestimmtes Clustering-Problem spielt.

Durch eine gründliche Untersuchung der vorherigen Resultate (siehe Abbildungen 4.1, 4.7b) bei der Anwendung des Algorithmus mit fixierten Mittelpunkt im Vergleich zur Anwendung ohne fixierten Mittelpunkt wird kein signifikanter Unterschied zwischen diesen Ergebnissen offenkundig. Im Falle der Anwendung des Algorithmus ohne explizite Festlegung eines Mittelpunktes repräsentieren die entstandenen Cluster Gruppierungen von Textdokumenten, wobei jede dieser Gruppen ähnliche Texte enthält. Bei der Anwendung des Algorithmus zur Festlegung eines fixierten Mittelpunktes bilden die resultierenden Cluster ebenfalls Ansammlungen von Textdokumenten, wobei die Zuordnung dieser Cluster zu einem fixierten Mittelpunkt von deren Ähnlichkeit zu diesem Mittelpunkt abhängt. Cluster, die eine größere Anzahl von Texten beinhalten, weisen eine höhere Ähnlichkeit zum fixierten Mittelpunkt auf, während Cluster mit weniger Texten eine geringere Ähnlichkeit damit besitzen. Aus den vorherigen Ausführungen ergibt sich, dass bei der Festlegung eines fixierten Mittelpunkts einige Cluster an Größe zunehmen, da sie mehr Texte beinhalten als solche, die dem Mittelpunkt ähnlich sind. Andererseits verringert sich bei einigen Clustern die Größe, da sie weniger Ähnlichkeit mit dem fixierten Mittelpunkt aufweisen. Die Schlussfolgerung daraus lautet, dass die Anwendung des Algorithmus, welcher ein fixierter Mittelpunkt berücksichtigt, keine bedeutende Auswirkung auf die Qualität der Textklassifizierung hat, obwohl der Silhouettenkoeffizient nicht abnimmt. Die optimale Darstellung der Cluster-Klassifizierung wird erreicht, wenn das gewählte fixierten Mittelpunkt eine hohe Ähnlichkeit zu allen Elementen des Clusters aufweist, wodurch alle Cluster im Ergebnis eng beieinander liegen.

5.3 Anwendung und Kontext

Der entwickelte Ansatz zur Textklassifikation und -clustering hat breite Anwendungsmöglichkeiten in verschiedenen Kontexten und Szenarien. Im Folgenden werden einige potenzielle Anwendungsbereiche hervorgehoben.

5.3.1 Empfehlungssysteme

In Empfehlungssystemen könnte der Ansatz verwendet werden, um Benutzern Inhalte vorzuschlagen, die ähnlich zu den von ihnen bereits konsumierten oder favorisierten Inhalten sind. Die generierten Cluster könnten als Grundlage für solche Empfehlungen dienen, indem sie ähnliche Dokumente zusammenfassen und damit eine schnelle und effiziente Identifikation von relevanten Inhalten ermöglichen.

5.3.2 Informationsmanagement und Wissensorganisation

Der Ansatz könnte auch in Kontexten wie Unternehmen oder Forschungsinstitutionen eingesetzt werden, wo es oft notwendig ist, große Mengen an Dokumenten zu verwalten und zu organisieren. Durch das Clustering ähnlicher Dokumente könnten Informationen

leichter auffindbar und zugänglich gemacht werden, was die Effizienz und Produktivität steigern könnte.

5.3.3 Content Aggregation und Kuratierung

In der Medien- und Nachrichtenindustrie könnte der entwickelte Ansatz dazu beitragen, ähnliche Artikel oder Geschichten zu aggregieren und personalisierte Nachrichtenfeeds zu erstellen. Zusätzlich könnte durch die Einbeziehung von Sentiment-Analyse der Ton oder die Stimmung der Artikel analysiert und berücksichtigt werden, um noch relevantere und ansprechendere Inhalte bereitzustellen.

5.4 Praktische Herausforderungen und Lösungsansätze

Während der Entwicklung und Implementierung des vorgestellten Ansatzes traten verschiedene praktische Herausforderungen auf. Im Folgenden werden diese Herausforderungen und die angewandten Lösungsansätze diskutiert.

5.4.1 Datenvolumen und Rechenkapazität

Eine der größten Herausforderungen bei der Arbeit mit Textdaten ist das Management des Datenvolumens und der Rechenkapazität. Insbesondere stellte die ressourcenintensive Aufgabe der Generierung von BERT-Einbettungen eine Herausforderung dar. Um dieses Problem zu bewältigen, wurde die Entscheidung getroffen, eine verteilte Berechnungsstrategie anzuwenden. Dabei wurde die Aufgabe auf mehrere Rechenknoten aufgeteilt, um die Berechnungszeit zu reduzieren.

5.4.2 Qualität der Eingabedaten

Ein weiterer kritischer Faktor war die Qualität der Eingabedaten. Schlecht formatierte oder unstrukturierte Daten können die Qualität der resultierenden Cluster erheblich beeinträchtigen. Um dies zu bewältigen, wurde einen umfassenden Vorverarbeitungsprozess durchgeführt, um die Daten zu bereinigen und zu normalisieren. Dies beinhaltete Schritte wie das Entfernen von Stoppwörtern, die Durchführung einer Lemmatisierung und die Korrektur von Rechtschreibfehlern. Dies stellte sicher, dass die Eingabedaten in einem guten Zustand waren, bevor sie in den Clustering-Prozess eingingen.

6 Zusammenfassung und Ausblick

Die Vorliegende Bachelorarbeit liegt in der Anwendung des FCK-Means (Fixed Centroids K-Means) Clustering-Algorithmus auf Textdokumente. Die Hauptzielsetzung dieser Forschungsarbeit besteht darin, ein verfeinertes Clustering-Verfahren zu entwickeln und umzusetzen, das eine sinnvolle Gruppierung von Textdokumenten gewährleistet. Hierbei kommt eine modifizierte Ausführung des etablierten K-Means-Clustering-Algorithmus zum Einsatz. In dieser Variante sind die Mittelpunkte der Cluster teilweise fixiert, was dazu beiträgt, die Güte der erzielten Clustering-Ergebnisse zu erhöhen. Der erste Abschnitt der Arbeit legt die Grundlagen im Bereich Natural Language Processing (NLP) und Information Retrieval dar, um das Verständnis für die Verarbeitung von Textdaten und die Herausforderungen bei der Gruppierung von Dokumenten zu schaffen. Es werden auch die herkömmlichen K-Means- und K-Medoid-Clustering-Methoden erläutert, um den Kontext für die Einführung des FCK-Means-Algorithmus zu schaffen. Die Forschungsmethodik umfasst mehrere Schritte, beginnend mit der Datenvorverarbeitung, in der Textdaten aufbereitet werden, um eine geeignete Repräsentation für das Clustering zu schaffen. Anschließend erfolgt die Merkmalextraktion, bei der relevante Eigenschaften der Textdokumente extrahiert werden.

Der Kern der Arbeit liegt im FCK-Means-Clustering, bei dem die teilweise fixierten Clustermittelpunkte dazu beitragen, die Konvergenzgeschwindigkeit und Qualität der Clusterbildung zu verbessern. Die Ergebnisse der Forschung werden in zwei Phasen visualisiert: In der ersten Phase werden die Ergebnisse des FCK-Means-Clustering ohne fixierte Mittelpunkte präsentiert, gefolgt von der zweiten Phase mit den fixierten Mittelpunkten. Die Visualisierungen verdeutlichen die Effektivität des Algorithmus bei der Gruppierung von Textdokumenten und die Auswirkungen der fixierten Mittelpunkte auf die erzielten Ergebnisse. Obwohl diese Arbeit einen wichtigen Schritt in der Weiterentwicklung von Clustering-Verfahren für Textdokumente darstellt, dennoch gibt es noch Raum für zukünftige Erweiterungen und Forschungen auf diesem Gebiet. Eine vielversprechende Ausrichtung könnte darin bestehen, die Verbesserung der fixierten Zentren stärker zu fokussieren und den Algorithmus weiter zu optimieren.

In kommenden Untersuchungen könnten verschiedene Ansätze erforscht werden, um die Auswahl und Positionierung der fixierten Mittelpunkte genauer zu betrachten und wie diese den Einfluss auf die Qualität der Clusterbildung beeinflussen. Die Einbindung fortgeschrittener Techniken des Natural Language Processing (NLP) zur Extraktion von Textmerkmalen stellt ebenfalls einen interessanten Ansatz dar. Durch die Berücksichtigung

wichtiger semantischer Informationen könnten die Ergebnisse des Clusterings weiter verfeinert werden. Ein umfassender Vergleich des FCK-Means-Algorithmus mit anderen modernen Clustering-Methoden könnte zusätzliche Erkenntnisse darüber liefern, wie effektiv er ist und ob er in verschiedenen Szenarien möglicherweise besser abschneidet. Zusammenfassend trägt diese Arbeit dazu bei, Clustering-Verfahren für Textdaten weiterzuentwickeln. Sie eröffnet interessante Möglichkeiten für zukünftige Forschung und Anwendungen im breiten Feld der Textanalyse und -verarbeitung.

Literaturverzeichnis

[A+23] Merhad Ay, Lale Özbakır, Sinem Kulluk, Burak Gülmez, Güney Öztürk, and Sertay Özer. FC-KMeans: Fixed-centered K-Means algorithm. *Expert Systems With Applications*, 211:118656, 1 2023.

[Bha21] Ashutosh Bhardwaj. Silhouette Coefficient - towards data science. 12 2021.

[BHJ22] Hamid Bekamiri, Daniel S. Hain, and Roman Jurowetzki. A survey on sentence embedding models Performance for patent analysis. *arXiv (Cornell University)*, 4 2022.

[BKL09] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. 6, 2009.

[CY10] Danyang Cao and Bingru Yang. An improved k-medoids clustering algorithm. In *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, volume 3, pages 132–135, 2010.

[Dee23] Deepanshi. Text preprocessing in NLP with Python codes. *Analytics Vidhya*, 4 2023.

[DNHOK20] Moni Kishore Dhar, S. M. Nahid Hasan, Tahsin Rahaman Otushi, and Musharrat Khan. Entropy-based feature selection for data clustering using k-means and k-medoids algorithms. In *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, pages 36–40, 2020.

[Egg22] Roman Egger. *Text representations and word embeddings*. 1 2022.

[EJYHG20] Wang Huai Mo Fan Li Qijun Wang E, An Jian, Liang Yan, and Wang Hong-Gang. Optimization of K-medoids Algorithm for Initial Clustering Center. *Journal of physics*, 1487:012011, 3 2020.

[Fah21] Ahmed Fahim. K and starting means for k-means algorithm. *Journal of Computational Science*, 55:101445, 10 2021.

[GGGT21] Vedant Amit Gandhi, Vaibhav Bharat Gandhi, Dhrumil Vinil Gala, and Prachi Tawde. A study of machine translation approaches for gujarati to english translation. In *2021 Smart Technologies, Communication and Robotics (STCR)*, pages 1–5, 2021.

[Gro22] Felix Gross. *Grundlagen des Text Mining*. 1 2022.

[Gru23] Hannes Grunert. Vorlesung data warehouses, data mining, 2023.

[Hu22] Jiawei Hu. An Overview for text representations in NLP - Towards Data science. 3 2022.

[IEA+23] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Heming Jia. K-Means Clustering Algorithms: A comprehensive review, variants analysis, and advances in the era of Big data. *Information Sciences*, 622:178–210, 4 2023.

[JM09] Dan Jurafsky and James H. Martin. Speech and Language Processing, 1 2009.

[JS17] Kavitha Jayaram and K Sangeeta. A review: Information extraction techniques from research papers. In *2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pages 56–59, 2017.

[KCA16] Shruti Kapil, Meenu Chawla, and Mohd Dilshad Ansari. On k-means data clustering algorithm with genetic algorithm. In *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pages 202–206, 2016.

[KMB+17] Vladimer Kobayashi, Stefan T. Mol, Hannah A. Berkers, Gábor Kismihók, and Deanne N. Den Hartog. Text mining in organizational research. *Organizational Research Methods*, 21(3):733–765, 8 2017.

[KS21] Rashi Kumar and Vineet Sahula. Intelligent approaches for natural language processing for indic languages. In *2021 IEEE International Symposium on Smart Electronic Systems (iSES)*, pages 331–334, 2021.

[Lan68] F. W. Lancaster. Information retrieval systems; characteristics, testing, and evaluation. 1968.

[Lan22] Fei Lan. Research on Text Similarity Measurement Hybrid Algorithm with Term Semantic Information and TF-IDF Method. *Advances in multimedia*, 2022:1–11, 4 2022.

[Lay23] Abdesslem Layeb. CKMeans and FCKMeans : Two Deterministic Initialization Procedures for KMEANS Algorithm using a modified crowding distance, 4 2023.

[LGF+22] Zuhe Li, Qingbing Guo, Chengyao Feng, Liya Deng, Qiuwen Zhang, Jianwei Zhang, Fengqin Wang, and Qian Sun. Multimodal sentiment analysis based on interactive transformer and soft mapping. *Wireless Communications and Mobile Computing*, 2022:1–12, 2 2022.

[Lin23] Chen Ling. Improving open information extraction with large language models: A study on Demonstration Uncertainty, 9 2023.

[LL12] Gang Li and Fei Liu. Application of a clustering method on sentiment analysis. *Journal of Information Science*, 38(2):127–139, 2 2012.

[LL19] Nico Litzel and Stefan Luber. Was ist der K-Means-Algorithmus? 3 2019.

[LM14] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *International Conference on Machine Learning*, 4:1188–1196, 6 2014.

[LS23] Carsten Lanquillon and Sigurd Schacht. *Informationsextraktion aus Texten*. 1 2023.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[MB00] Ujjwal Maulik and Sanghamitra Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern Recognition*, 33(9):1455–1465, 9 2000.

[McC19] Chris McCormick. BERT Word Embeddings Tutorial · Chris McCormick, 5 2019.

[McG20] Milecia McGregor. 8 Clustering algorithms in machine learning that all data scientists should know. *freeCodeCamp.org*, 9 2020.

[Mel23] Melanie. NLP – Natural Language Processing: eine Einführung, 4 2023.

[Min20] 5 Am häufigsten verwendete Distanzmetriken beim maschinellen Lernen. *ICHI.PRO*, 12 2020.

[MK20] Helfried Moosbrugger and Augustin Kelava. *Qualitätsanforderungen an Tests und Fragebogen („Gütekriterien“)*. 1 2020.

[Mog09] Isaac C. Mogotsi. Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze: Introduction to Information Retrieval. *Information Retrieval*, 13(2):192–195, 9 2009.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Neural Information Processing Systems*, 26:3111–3119, 12 2013.

[NZT+11] Feiping Nie, Zinan Zeng, Ivor W. Tsang, Dong Xu, and Changshui Zhang. Spectral embedded clustering: A framework for in-sample and out-of-sample spectral clustering. *IEEE Transactions on Neural Networks*, 22(11):1796–1808, 2011.

[PS22] Rahul Pradhan and Dilip Kumar Sharma. RETRACTED ARTICLE: An ensemble deep learning classifier for sentiment analysis on code-mix Hindi–English data. *Soft Computing*, 27(15):11053, 4 2022.

[R23] Adriano R. Künstliche Intelligenz: Zusammenfassung eines Textes mithilfe des TF IDF, 2 2023.

[RPS23] Yutika Rajanak, Ratna Patil, and Yadvendra Pratap Singh. Language detection using natural language processing. In *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, volume 1, pages 673–678, 2023.

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. Document length normalization. Information Processing and Management. *32(5):619–633*, 1 1996.

[SEKX98] Joerg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery*, 2(2):169–194, 1 1998.

[Sin21] Falak Singh. Sentiment analysis made easy using VADER. *Analytics India Magazine*, 2 2021.

[SJA22] Akash S, Neeraj Jayaram, and Jesudoss A. Desktop based smart voice assistant using python language integrated with arduino. In *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*, pages 374–379, 2022.

[SRM+12] Mohameth-François Sy, Sylvie Ranwez, Jacky Montmain, Armelle Regnault, Michel Crampes, and Vincent Ranwez. User centered and ontology based information retrieval system for life sciences. *BMC Bioinformatics*, 13(S1), 1 2012.

[Sun08] Ji-Gui Sun. Clustering algorithms research: Clustering algorithms research. *Journal of Software*, 19:48–61, 2008.

[Zhe19] Yuhan Zheng. An exploration on text classification with classical machine learning algorithm. In *2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 81–85, 2019.

Selbstständigkeitserklärung zur Bachelorarbeit

Ich, Abdullatif Akrin, erkläre, dass ich die vorliegende Arbeit zum Thema „*k-Means-Clustering von Textdokumenten mit teilweise fixierten Clustermittelpunkten*“ selbständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, 02. Oktober 2023

Ort, Datum

Unterschrift