h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbmn
FACHBEREICH MATHEMATIK
UND NATURWISSENSCHAFTEN

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Hochschule Darmstadt

Fachbereiche Mathematik und
Naturwissenschaften & Informatik

Konzeption und Evaluierung eines Recommender Systems für eine Frage-Antwort-Plattform

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M. Sc.)
im Studiengang Data Science

vorgelegt von

Daniel Becker

Referent	:	Prof. Dr. Gunter Grieser
Korreferent	:	Prof. Dr. Horst Zisgen

Ausgabedatum	:	1. August 2019
Abgabedatum	:	23. Januar 2020

ERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.

Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, 23. Januar 2020

Daniel Becker

ABSTRACT

Community question answering are very popular on the Internet, where users can ask their questions and get a quick answer to solve their problem. This is also of interest for companies as an internal solution to help employees quickly with problems and to build up an internal knowledge database. By using a recommender system, the existing system should be supported to match questions based on their content and users based on their interest.

In this thesis a concept to implement a recommender system for a community question answering forum for the company PRODYNA SE will be presented. Different approaches to modify and optimise the Recommender System will be used. Three different data sets will be applied in order to consider different use cases and to evaluate the recommender system.

Using Doc2Vec is a good way to present the text of the used questions as vectors. Compared to common methods such as BoW or Tf-idf, this has the additional advantage that a smaller vector is needed for a question, which also results in a faster runtime for further calculations.

The calculation of user profiles, which are used to match questions and users, is limited to the interest of the user in this thesis. Since the interest of a user can change over time, approaches for this behavior are presented and their effects are evaluated. By using a weighted arithmetic mean for the user profiles, which looks more at recent answered questions, and by using a Doc2Vec model for the question profiles, an increase in accuracy of up to 5.79 percentage points is possible compared to the original Tf-idf model and the normal arithmetic mean.

Keywords: *Natural Language Processing, Recommender, Doc2Vec, Similarity, Stack Overflow, Community-Question-Answering*

KURZFASSUNG

Frage-Antwort-Plattformen haben im Internet eine sehr hohe Beliebtheit, da dort Benutzer bei einem Problem ihre Frage stellen können und schnell eine Antwort zur Lösung ihres Problems bekommen. Dies findet mittlerweile auch für Firmen als interne Lösung Interesse, um Mitarbeiter bei Problem schnell helfen zu können und eine interne Wissensdatenbank aufzubauen. Durch die Verwendung eines Recommender Systems, soll das bestehende System dabei unterstützt werden, dass Fragen anhand ihres Inhaltes und Benutzer anhand ihres Interesse zusammengeführt werden.

In dieser Arbeit wird ein Konzept zur Realisierung eines Recommender Systems für eine Frage-Antwort-Plattform der Firma PRODYNA SE erstellt. Dabei werden auf die verschiedenen Phasen zu Erstellung des Recommender Systems eingegangen und Ansätze vorgestellt, um in diese einzugreifen und zu optimieren. Zur Berücksichtigung von verschiedenen Anwendungsszenarien werden drei unterschiedliche Datensätze verwendet, anhand den eine Evaluierung des Recommender Systems vorgenommen wird.

Die Verwendung von Doc2Vec ist eine gute Möglichkeit um die Texte der verwendeten Fragen als Vektor darzustellen. Gegenüber verbreiteten Varianten wie BoW oder Tf-idf entsteht dadurch zusätzlich der Vorteil, dass eine kleinerer Vektor benötigt wird für eine Frage, wodurch auch eine schnelle Laufzeit bei weiteren Berechnungen entsteht.

Bei der Ermittlung von Benutzerprofile, welche zur Übereinstimmung zwischen Fragen und Benutzer dienen, wird sich in dieser Arbeit auf das Interesse des Benutzers beschränkt. Da sich das Interesse eines Benutzer über die Zeit verändern kann, werden für dieses Verhalten Ansätze vorgestellt und deren Auswirkung evaluiert. Durch die Verwendung eines gewichteten arithmetischen Mittelwert für die Benutzerprofile, welches kürzlich beantwortete Fragen stärker betrachtet und der Verwendung eines Doc2Vec Modell für die Frageprofile ist eine Steigerung der Genauigkeit von bis zu 5.79 Prozentpunkte möglich gegenüber dem ursprünglichen Tf-idf Modell und dem normalen arithmetischen Mittelwert.

Schlagwörter: *Natural Language Processing, Empfehlungssystem, Doc2Vec, Ähnlichkeiten, Stack Overflow, Frage-Antwort-Plattform*

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

LISTINGS

ABKÜRZUNGSVERZEICHNIS

API Application Programming Interface

BOW Bag of Words

CQA Community Question Answering

CSS Cascading Style Sheet

HTML Hyptertext Markup Language

IDF inverse document frequency

IR Information Retrieval

KNN künstliches neuronales Netz

LDA Latent Dirichlet Allocation

ML Machine Learning

NER Named Entity Recognition

NLP Natural Language Processing

POS Part-of-Speech

PV-DM Distributed Memory Model of Paragraph Vectors

REST Representational State Transfer

RS Recommender System

SVM Support Vector Machine

TF term frequency

TF-IDF term frequency - inverse document frequency

Teil I

THESIS

1 EINLEITUNG

Zum Aufbau von Wissen haben Frage-Antwort-Plattform[1] (Community Question Answering (CQA)) immer mehr an Bedeutung gewonnen, welche durch die Zusammenarbeit von Benutzern entsteht. Benutzer können auf diesen Plattformen ihre Fragen stellen und durch einen oder mehrere beliebige anderen Benutzern auf dieser Plattform beantwortet werden. Als Beispiel gibt es öffentliche Plattformen wie Quora[2], Yahoo! Answers[3] und gutefrage[4], welche Themen unabhängig sind oder Stack Exchange[5], welches ein Netzwerk von CQA-Plattformen ist, mit z.B. Stack Overflow[6] speziell an Softwareentwickler gerichtet.

Dieses CQA findet auch immer mehr Bedeutung für die firmeninterne Nutzung, um den eigenen Mitarbeitern schnell mit ihren Problem bzw. Fragen zu helfen und gleichzeitig internes Wissen aufzubauen. Um ein erfolgreiches System zu gewährleisten, ist es wichtig, dass viele Benutzer aktiv daran teilnehmen und eine qualitative, sowie kurzfristige Antwort auf neue Fragen geben. Eine aktive Community ist daher ausschlaggebend für den Erfolg dieser Systeme. Mit steigender Anzahl an Aktivitäten, wird es allerdings unübersichtlich, welche Fragen es bereits gibt, welche unbeantwortete Fragen einen Benutzer interessieren können zum beantworten oder ob eine identische Frage evtl. schon gestellt und beantwortet wurde.

1.1 MOTIVATION

Vor dieser Herausforderung steht auch die Firma PRODYNA SE[7], welches für die Verwendung ihrer internen CQA-Plattform eine Optimierung vornehmen will, da das aktuelle System bisher nur eine chronologische Reihenfolge von Fragen darstellt. Dies soll in Form eines Recommender System (RS) stattfinden, welche das System unterstützen soll bzgl. Empfehlungen bei neuen Fragen und den Benutzerpräferenzen. Ein produktbezogenes RS aus dem E-Commerce möchten ein Produkt so oft wie möglich empfehlen (verkaufen), während bei dem CQA die Priorität darin liegt, noch unbeantwortete Fragen zu bevorzugen. Beide haben jedoch das Ziel, das Interesse eines Benutzer zu identifizieren und entsprechend Empfehlungen auszusprechen.

1 Im weiteren Verlauf dieser Arbeit wird die verbreitete Abkürzung *CQA* verwendet, welches im Englischen für *Community Question Answering* steht
2 https://www.quora.com
3 https://answers.yahoo.com
4 https://www.gutefrage.net
5 https://stackexchange.com
6 https://stackoverflow.com
7 https://www.prodyna.com

Während bei öffentlichen CQA-Plattformen die Benutzer sich aktiv Registrieren, um auf dieser Mitzuwirken, müssen bei einer firmeninternen CQA-Plattform die Mitarbeiter dazu motiviert werden, diese zu benutzen. Wenn diese zu selten potentielle Fragen als Empfehlungen bekommen, kann dies den Effekt haben, dass diese die Motivation verlieren und selten auf die Plattform schauen. Bei zu vielen oder falschen Empfehlungen hingegen, kann dies ebenfalls einen negativen Effekt haben, da dies als Spam angesehen werden könnte und das Vertrauen in das RS verloren geht. Der Erfolg hängt somit davon ab, dass eine gute Zuordnung zwischen (unbeantworteten) Fragen und Benutzern vorgenommen wird, welche das Interesse und Motivation haben, um ihre empfohlenen Fragen innerhalb kurzer Zeit zu beantworten.

Das Einbinden von neuen Benutzern ist ein besonderes Problem in ein solches RS, da noch keine Aktivitäten über diese vorhanden sind und somit vorhandenes Potential verloren geht. Durch das Abfragen und dem Andienen zusätzlicher Benutzerpräferenzen bei der Registrierung für das System, kann dies entgegengewirkt werden. Die Vorgabe für das CQA der PRODYNA SE gibt allerdings vor, dass Mitarbeiter dies nutzen können, ohne weitere Angaben vorzunehmen. In einem firmeninternen System gibt es aber bereits durch andere IT-Systeme Informationen zu den Mitarbeitern, welche seine Themeninteresse abdecken und berücksichtigt werden sollten. Die Mitarbeiter können dadurch schneller in das RS eingebunden werden und Fragen als Empfehlungen enthalten, welche sie interessieren, anstatt diese suchen zu müssen auf der Plattform.

Auch die Veränderung des Interesse an einem Thema für einen Benutzer sollte bei den Empfehlungen berücksichtigt werden. Dies wäre z.B. der Fall, wenn sich ein Benutzer vermehrt für eine neue Technologie interessiert und die alte vernachlässigen möchte oder eine neue Position bzw. Aufgaben in der Firma übernimmt und sich auf die neuen Themen konzentrieren möchten. Hierbei sollte also auch beachtet werden, dass auf neue Begebenheiten eines Benutzers eingegangen werden kann und sein aktuelles Interesse ermittelt wird.

Durch die Verwendung eines RS und das Anbinden zusätzlicher Daten außerhalb des eigentlichen Systems, möchte die PRODYNA SE eine Abbildung der Benutzerpräferenzen gewährleisten, um gezielt offene Fragen für diese zu Empfehlen, welche auch in der Lage sein sollten, diese zu beantworten. Auch neue Benutzer innerhalb des Systems können somit auf sie abgestimmte Fragen empfohlen werden, ohne weitere Interaktionen zu verlangen. Dadurch soll das Vertrauen gestiegen werden, damit weitere Mitarbeiter das System nutzen, die aktive Community größer wird und sich folglich auch eine interne Wissensdatenbank aufgebaut wird.

1.2 ZIEL DER ARBEIT

Mit dieser Arbeit soll ein Konzept erstellt werden für ein RS der firmeninterne CQA-Plattform, welches bei der Firma PRODYNA SE eingesetzt werden kann. Das Hauptaugenmerk liegt dabei auf ein mögliches RS für die Empfehlung von neuen Fragen an Benutzer, welche sich in diesem Themengebiet auskennen und diese somit beantworten können. Hierbei sollte in einem ersten Schritt geprüft werden, welche Möglichkeiten es für den Einsatz eines RS auf einer solchen CQA Plattform gibt, welche Bedingungen erfüllt werden müssen und wie dies in die Umgebung der PRODYNA SE eingebunden werden kann.

Da es sich bei den Daten primär um Texte (Fragen und Antworten) handelt, ist es notwendig Methoden zu finden, welche die Inhalte der Texte effektiv in einem Vektor darstellen können, da dies für die weitere Anwendung von Machine Learning (ML) Verfahren notwendig ist. Hierbei soll auch berücksichtigt werden, wie vorhandene Mitarbeiterdaten außerhalb des Systems genutzt werden können, um das RS zu unterstützen bzw. zu optimieren. Es wird sich dabei auf zusätzliche Mitarbeiterdaten, welche ebenfalls in Textform (z.B. interne Blogs, Lebenslauf oder Chats) vorliegen beschränkt. Dies soll ermöglichen, dass die Qualität der Empfehlung von neuen Fragen an Benutzern optimiert werden, um evtl. noch für das System unbekannte Präferenzen des Benutzers zu berücksichtigen und die Startphase von neuen Benutzern vereinfachen. Weiterhin soll berücksichtigt werden, dass sich Benutzerinteresse im Laufe der Zeit verändern können und somit das aktuelle Interesse eines Benutzers stärker berücksichtigt wird. Hierzu werden verschiedene Ansätze beschrieben und evaluiert, wie diese sich auf die vorhandenen Daten auswirken.

Folgende Fragestellungen sollten am Ende dieser Arbeit beantwortet werden können:

- Was ist die bevorzugte Methoden, um die Textinhalte von Fragen als Vektor darstellen zu können für die weitere Verwendung?
- Kann durch die Anbindung zusätzlicher Daten außerhalb des Systems eine Optimierung der Empfehlungen erzielt werden?
- Wie können Veränderungen von Benutzerinteressen berücksichtigt werden, wenn sich diese über einen Zeitraum verändert haben?
- Was muss berücksichtigt werden, um die Erweiterbarkeit zu gewährleisten, wenn in Zukunft noch weitere bisher unbekannte Informationen eingebunden werden sollen?
- Gibt es Unterschiede, welche beachtet werden bei verschiedenen Vorhandenen Ausgangsdaten, welche unterschiedliche Szenarien darstellen?

Da die vorhandenen Daten aus dem CQA-System der Firma PRODYNA SE nicht veröffentlicht werden können, ist es ebenfalls relevant für die spätere Evaluierung, dass ein öffentlich verfügbarer Datensatz gesucht und verwendet wird. Folgende Merkmale sollten dabei an den Datensatz berücksichtigt werden:

- Die Daten sollten öffentlich verfügbar sein, ohne dass es eine Registrierung benötigt.
- Für eine repräsentative Evaluierung sollte die Daten eine gewisse Menge und Vielfalt besitzen. So sollten sich die Fragen nicht um ein einzelnes spezielles Thema (z.B. Python) handeln, sondern umfangreicher sein (z.B. verschiedene Programmiersprachen).
- Es sollte Poweruser geben, welche sehr aktiv in dem System sind über einen längeren Zeitraum und viele Fragen beantwortet haben. Anhand diese soll geprüft werden, wie sich die Veränderung von Benutzerinteressen darstellt und wie auf diese eingegangen werden kann.
- Ebenfalls sollte es die Möglichkeit geben, für die Benutzer des Systems zusätzliche Daten in Textform außerhalb des Systems (z.B. Blogs, Webseiten, Lebensläufe) extrahieren zu können. Diese Daten sollen beispielhaft für die externen Daten stehen, welche zusätzlich angereichert werden.

1.3 GLIEDERUNG

In dem Kapitel 2 wird auf die Ausgangssituation der verwendeten CQA-Plattform der Firma PRODYNA SE eingegangen, was deren Systemarchitektur beschreibt, sowie das Schema der Daten, welche für diese Arbeit verwenden werden. Anschließend werden die Anforderungen beschrieben, um zum einen das RS in das bestehende System zu integrieren und wie und welche Aufgaben das RS bewerkstelligen sollte.

In Kapitel 3 erfolgt eine kurze Einführung in die Verwendung von künstlichen neuronalen Netze, welche für das RS als Klassifizierer verwendet werden, sowie für die Erstellung der Frageprofile. Anschließend wird auf verschiedene Bereiche bei der Verarbeitung von Texten eingegangen, welche relevant sind, um deren Inhalte in einem Vektor zu transformieren, welches für die weitere Verwendung von ML Verfahren notwendig ist, wie z.B. bei der Verwendung eines neuronalen Netz. Abschließend wird in diesem Kapitel noch auf verschiedene Metriken eingegangen, welche zur Berechnung von Ähnlichkeiten zwischen Vektoren und zur späteren Evaluierung der Ergebnisse relevant sind.

Die Beschreibung der Implementierung findet in dem Kapitel 4 statt. Dies beinhaltet die verwendeten Vorverarbeitungsschritte zur Bereinigung der Texte, sowie aufbauend auf diese, Varianten zum Erstellen der Frageprofile als Vektor und die verschieden Parameter, welche währenddessen berücksichtigt werden können. Anschließend werden verschiedene Methoden vorgestellt zur Berechnung der Benutzerprofile, welche aus einer Aggregierungs- und einer Zerfallfunktion besteht und es ermöglichen soll, Veränderungen von Benutzerinteresse zu berücksichtigen. Die Frage- und Benutzerprofile sind der Ausgangspunkt für das RS, welches entscheiden soll, ob zwischen diesen eine Übereinstimmung für eine Empfehlung besteht. Zuletzt wird in diesem Kapitel daher noch auf die verwendete Architektur des neuronalen Netz eingegangen, sowie die Durchführung der Trainings- und Testphase des RS.

Die Evaluierung der vorgestellten Konzepte zum Aufbau eines RS für die CQA-Plattform findet in dem Kapitel 5 statt. Da es sich als Ziel gesetzt wurde zu überprüfen, wie diese auf verschiedenen Anwendungsszenarien Ergebnisse erzielen, wurden drei verschiedene Datensätze verwendet, welche zuerst beschrieben werden. Auf diesen wurden zwei Verfahren mit verschiedenen Parametereinstellung zum Erstellen der Frageprofile angewandt, um zu überprüfen, welches Verfahren und Einstellungen die bessere Ergebnisse erzielt. Aufbauend auf den Frageprofile, welche in diesem Schritt eine hohe Genauigkeit in den Ergebnisse erzielten, wurde weiterführend die Evaluierung der verschiedenen Berechnungen zur Ermittlung der Benutzerprofile, sowie das zusätzliche Einbeziehen der externen Benutzerdaten vorgenommen. Ziel dieses Kapitels ist somit eine Empfehlung vorzunehmen, welche Methoden und Parameter für die Verwendung eines RS gute Ergebnisse erzielen.

Das letzte Kapitel 6 beinhaltet eine abschließende Betrachtung und Zusammenfassung der Ergebnisse. Es wird geprüft, ob alle am Anfang gestellten Fragen beantwortet und die Ziele erreicht wurden, sowie ein Ausblick für zukünftige Weiterentwicklungen beschrieben, welche sinnvoll sein könnte zum weiteren Funktionsausbau des RS und dessen Optimierung.

2

FRAGE-ANTWORT-PLATTFORM

Frage-Antwort-Plattformen (Community Question Answering (CQA)) dienen dazu, das Wissen zu erweitern und jemanden mit seinem Problem bzw. Frage schnell zu helfen. Dies geschieht durch das kollaborative Zusammenarbeiten vom Benutzer auf solch einer Plattform. Benutzer können dort ihre Frage stellen und andere Benutzer können diese dann beantworten. Die Benutzer können dabei sowohl die Rolle des Fragestellers, als auch des Beantworter einnehmen. Durch das Fragestellen und beantworten dieser, entsteht eine Wissensdatenbank, welche in Zukunft genutzt werden können, wenn jemand anderes die identische bzw. ähnliche Frage stellen möchte. Diese kann dann dem Fragesteller empfohlen werden, um zu prüfen, ob diese Frage schon gestellt und beantwortet wurde und somit kein Duplikat entsteht und der Fragesteller nicht zuerst auf eine neue Antwort auf seine Frage warten muss.

Die Fragen sollen zu jeden beliebigen Fachgebiet gestellt werden dürfen und somit muss sich auch eine Experte finden lassen, welcher die Frage beantworten kann. Weiterhin ist es wichtig, dass aktive Benutzer vorhanden sind, da die Frage kurzfristig beantwortet werden sollte. In diesem Kapitel soll auf die Gegebenheiten der CQA Plattform im Zusammenhang mit der PRODYNA SE eingegangen werden und wie diese erweitert werden kann, um das vorgestellte Recommender System (RS) in diese zu integrieren.

2.1 BEKANNTE ANSÄTZE

In diesem Abschnitt werden auf vorhandene Ansätze eingegangen und deren verschiedene Anwendungsbereiche im Zusammenhang mit einem RS im Bereich CQA. Ein Teil dieser Veröffentlichungen dienen als Grundlage für die weitere Verwendung in dieser Arbeit. Weiterhin wird am Ende dieses Abschnitt eine Auflistung über Ansätze gemacht, welche in dieser Arbeit verfolgt werden.

Bei der Verwendung von CQA wird unterschieden zwischen der Weiterleitung von neuen Fragen an Benutzer, welche diese am besten beantworten können und dem empfehlen von Fragen, welche einen Benutzer interessieren könnten. Die meisten Ansätze beziehen sich auf das erste Problem, da neue Fragen schnellstmöglich beantwortet werden sollen. Dabei soll eine neue Frage q an eine Gruppe von Benutzer $u_1, u_2, ..., u_n$ weitergeleitet werden, welche am geeignetsten sind diese zu beantworten. Bei dem zweiten Problem soll für einen Benutzer u auf Anfrage eine Liste von Fragen $q_1, q_2, ..., q_n$ empfohlen werden, welche ihn am meisten interessieren könnten.

Der Unterschied zur Weiterleitung von neuen Fragen an Benutzern liegt darin, dass diese daher aktiv am System teilnehmen und Fragen beantworten wollen, während bei dem anderen Ansatz die Benutzer informiert werden. Zusätzlich zum Wissen, eine Frage beantworten zu können, kann daher auch relevant sein, aktive Benutzer stärker zu berücksichtigen, welche Motiviert sind zu helfen [Che+14; Don+15; LLY10; ZLK12; Mac+17].

Der Prozess für das Erstellen von Empfehlungen wird in drei Schritte unterteilt:

1. Das Erstellen eines Fragenprofils.
2. Das Erstellen eines Benutzerprofils.
3. Zusammenführung der Fragen- und Benutzerprofile.

Das Fragenprofil soll den Inhalt einer Frage darstellen. Diese Fragen werden dabei als Vektor dargestellt, durch Verwenden von Bag of Words (BoW), term frequency - inverse document frequency (Tf-idf) [Che+14; Don+15; Dro+11], Topic Modelle wie Latent Dirichlet Allocation (LDA) [Tia+14; LLY10; SGB15; SMP13; SPK14; Kam+18] oder ein Kombination aus diesen [Ria+12]. Ergänzend können auch bei den Fragen hinterlegte Hashtags oder Kategorien berücksichtigt [Don+15; Dro+11; SMP13] werden, welche eine besondere Gewichtung bekommen für das Fragenprofil.

Das Benutzerprofil soll das Interesse und Wissen eines Benutzers darstellen, welches anhand seiner vergangenen beantworten Fragen innerhalb des CQA-Systems ermittelt wird. Hierbei gibt es noch Ansätze, welche zusätzlich die Antwortzeiten, Qualität der Antworten und weitere Features berücksichtigen [ZLK12; Mac+17].

Für die Zusammenführung der Fragen- und Benutzerprofile zur Erstellung der Empfehlungen kann ein Klassifikationsverfahren genutzt werden, welches vorhersagt, wie gut die beiden Profile zusammenpassen [Che+14; Dro+11; Luo+14; ZLK12; Mac+17]. Eine weitere verbreitete Variante ist ein Rankingmodell, welches anhand von Ähnlichkeiten die besten Übereinstimmung ermittelt [Ria+12; Don+15; Tia+14; LLY10; SGB15; SMP13; SPK14; Kam+18] und aus dem Dokumentenretrieval kommt.

Auch die Möglichkeit zur Nutzung von kollaborative Filtern, welche anhand einer Matrix zwischen Benutzer und Kategorien von beantworteten Fragen, Verhaltensmuster entdecken sollen und Empfehlungen aussprechen existieren [Cho+15; YM14]. Ebenfalls gibt es graphbasierte Lösungen, welche in einem Netzwerk die Beziehungen zwischen Benutzer, Fragen und deren Kategorien darstellen und Experten identifizieren sollen für Bereiche zum empfehlen [JA07; ZAA07; BDW08].

Für die Experimente und Evaluierung der Ansätze werden in der Regel Daten von öffentlich verfügbaren CQA-Plattformen verwendet. Die Ansätze konzentrieren sich dabei auf für Branchen unabhängige CQA-Daten, wie *Yahoo Answers* [Dro+11; SMP13; ZLK12] und für Domainspezifische Daten von *Stack Overflow* für Softwareentwickler [Ria+12; Don+15; Tia+14; SGB15; SPK14]. Während die Evaluierung der Experimente in der Regel auf einen extrahierten Datensatz Offline durchgeführt werden, gibt es auch einige Ansätze, welche die Auswirkung Online über einen vorgegebenen Zeitraum betrachtet haben in einem aktiven System [Che+14; SMP13; Mac+17; Kam+18]. Einige Ansätze berücksichtigen für die Erstellung der Benutzerprofile auch Daten außerhalb der CQA-Plattform, um zusätzliche Informationen zu den Benutzern zu erhalten [Luo+14; SGB15; Mac+17].

In dieser Arbeit wird sich auf die Durchführung eines Offline Experiment beschränkt, bei dessen Daten von der öffentliche CQA-Plattform *Stack Overflow* genutzt werden. Dies hat den Vorteil, dass dort Benutzer eine externe Webseite angeben können, welche als Bedingung für das Experiment als zusätzliche externe Daten verwendet werden können. Bei der Erstellung der Frageprofile wird zusätzlich zu dem Tf-idf Verfahren die Verwendung von Doc2Vec betrachtet und gegenübergestellt. Dieses ermöglicht ebenfalls die Verarbeitung von Texten zu Vektoren aber soll besser die Semantik von Wörtern und Texten berücksichtigen.

Bei der Betrachtung der Benutzerprofile wird sich auf das Benutzerinteresse beschränkt, da in dem aktuellen CQA-System der Firma PRODYNA SE weitere Funktionen, wie z.B. Upvote-Funktion für gute Antworten oder Belohnung von aktiven Benutzern derzeit nicht verwendet werden. In die Betrachtung der Benutzerprofile fließt dafür die Verwendung von zusätzlichen externen Daten ein, sowie die Berücksichtigung, dass sich das Interesse eines Benutzer mit der Zeit auch verändern kann und welche Methoden damit verbunden verwendet werden können.

Bei der Zusammenführung der Frage- und Benutzerprofile für die Empfehlung des RS, wird ein künstliches neuronales Netz (KNN) zur Klassifizierung eingesetzt. Gegenüber einem Ranking anhand von Ähnlichkeiten, bietet eine Klassifizierung den Vorteil, dass dies alle positiven Empfehlungen berücksichtigen kann und nicht nur eine vorgegebene Anzahl an Top N Empfehlungen. Ebenfalls wird dadurch gewährleistet, dass das RS in Zukunft einfach um neue Features erweitert werden kann, welche keinen Bezug zur Ähnlichkeit besitzen zwischen dem Frage- und Benutzerprofil.

2.2 AUSGANGSSITUATION

Für die Speicherung der Fragen und Antwort wird die dokumentorientierte NoSQL-Datenbank *MongoDB* verwendet. Über einen weiteren Service wird eine Representational State Transfer (REST) Application Programming Interface (API) bereitgestellt, welcher die Schnittstelle zwischen dem späteren Frontend und der MongoDB darstellt. Dies erlaubt z.B. die Möglichkeit, neue Fragen oder Antworten zu erstellen, überarbeiten oder anzeigen zu lassen. In Abbildung 2.1 wird das verwendete Datenbankschema der MongoDB dargestellt, welches als Ausgangspunkt der zur Verfügung stehenden Daten ist zum Aufbau des RS.

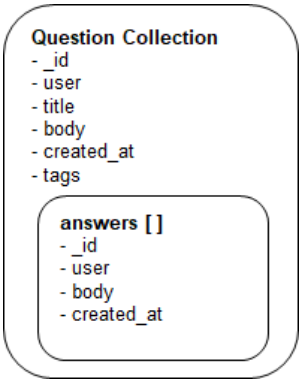


Abbildung 2.1: Schema für die CQA-Plattform in MongoDB

Eine Frage kann dabei mehrere Antworten sowie vom Verfasser hinterlegte Hashtags haben, welches auch in dem Beispielcode 2.1 zu sehen ist. Diese hat zwei Antworten und handelt den Hashtags entsprechend um die Themen *javascript*, *reactjs* und *firebase*. Die Daten unterteilen sich dabei in Metadaten, wie z.B. eine eindeutige ID und der Benutzer, welcher die Frage bzw. Antwort verfasst hat, sowie wann dies verfasst wurde. Weiterhin gibt es inhaltliche Informationen, welche die Frage bzw. Antworten beschreiben (Titel, Textkörper und Hashtags). Zusätzlich zu diesen Informationen, können aus diesen auch weitere relevante Merkmale extrahiert werden. Dies könnte z.B. die Anzahl der Antworten, welche ein Benutzer verfasst hat oder die Zeit, welche benötigt wurde, um die Frage zu beantworten, wodurch sich Rückschlüsse ziehen lassen, wie aktiv ein Benutzer ist.

Abschließend gibt es noch externe Daten, welche sich nicht innerhalb des CQA-Systems befinden aber einen Bezug zu den Benutzern des Systems besitzen können und somit als zusätzliche Information im späteren RS verwendet werden können. Mit Bezug auf die Firma PRODYNA SE kann dies hinterlegte Lebensläufe, verfasste Blogs (Yammer) oder Projektkommunikation (Teams) von Mitarbeiter sein, um auch nach Interessen und Wissen der Benutzer außerhalb des Systems zu finden bzw. den Start von neuen Mitarbeitern innerhalb des Systems zu vereinfachen.

Listing 2.1: Beispielfrage mit Antworten im JSON Format


```
{
        "_id": "5d9aeaa19ef3e6303a225c92",
        "user": "6945866",
        "created_at": "2018-01-01T00:07:04.000Z",
        "title": "Relation between storage and database - Firebase",
        "body": "<p>I'm facing yet another issue.</p> ...",
        "answers": [
                {
                        "_id": "5dc40a3d89b0e23550a23d42",
                        "user": "9105626",
                        "created_at": "2018-01-01T00:14:57.000Z",
                        "body": "<p>I would suggest using .getDownloadURL(). ..."
                },
                {
                        "_id": "5dc40e7b89b0e23550a23e20",
                        "user": "2019221",
                        "created_at": "2018-01-01T00:19:30.000Z",
                        "body": "<p>You can generate the push ID before you upload ..."
                }
        ],
        "tags": [ "javascript", "reactjs", "firebase" ]
}
```


2.3 ANFORDERUNGEN AN DAS RECOMMENDER SYSTEM

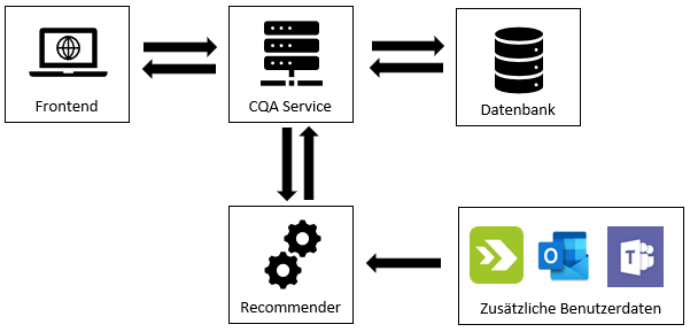


Abbildung 2.2: Architektur mit Integration des Recommender Systems

Das spätere RS sollte in die vorhanden Architektur integriert werden können. Hierzu sollte ein eigenständiger Service bereitgestellt werden, welcher eine RESTful API zu Verfügung stellt, um die notwendigen Empfehlungen zu berechnen, sowie die hinterlegten Modelle neu zu trainieren oder aktualisieren zu können. Dieser sollte wie in Abbildung 2.2 als zusätzlicher Service verwendet werden und kommuniziert mit den vorhandenen CQA-Service, sowie einen lesenden Zugriff auf die evtl. zusätzlichen Benutzerdaten außerhalb des CQA-Systems.

Das RS persistiert hierbei zum einen das trainierte Modell für die Vorhersagen, sowie die Fragen- und Benutzerprofile, um diese direkt verfügbar zu haben und eine bessere Performance zu gewährleisten, anstatt bei jeder Anfrage diese neu berechnen zu müssen.

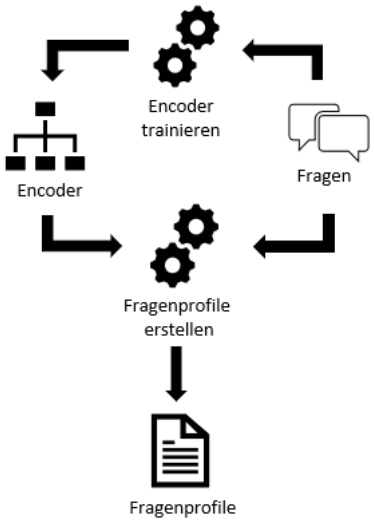


Abbildung 2.3: Ablauf zum Erstellen der Fragenprofile

Das RS soll genutzt werden, um die folgenden Fragen zu beantworten:

- Welche Fragen sind sich ähnlich aufgrund ihrem Inhalt und Themengebiet?
- Welche Benutzer können eine neue Fragen am wahrscheinlichsten beantworten?
- Welche unbeantwortete Fragen sollte einen Benutzer empfohlen werden zum beantworten?

Text-Encoder:
Der Text-Encoder nimmt einen Text entgegen, wendet die hinterlegten Vorverarbeitungsschritte darauf an, um diesen anschließend in einen trainierten Modell zu verwenden und daraus den Vektor des Textes zu erlangen. Dieser Ablauf ist in Abbildung 2.3 dargestellt. Die Vorverarbeitungsschritte werden in dem Unterkapitel 4.1 vorgestellt, sowie Varianten zum erstellen der Vektoren in Unterkapitel 4.2.

Fragenprofile:
Die Fragenprofile sind die Vektoren, welche durch den Text-Encoder berechnet werden, indem die inhaltlichen Informationen (Titel, Textkörper und Hashtags) der Frage verwendet werden. Diese Vektordarstellung ist relevant für die spätere Anwendung eines Machine Learning (ML) und Verarbeitungsverfahren. Persistiert werden diese mit der eindeutige ID der Frage, den dazugehörigen Vektor, sowie zusätzlich eine Liste der Benutzer, welche die Frage beantwortet haben und der Zeitstempel der Antwort. Diese Liste soll gewährleisten, dass ein schnellerer Zugriff auf relevanten Informationen verfügbar ist, ohne externe Hilfe. So kann z.B. direkt ermittelt werden, welche Fragen noch unbeantwortet sind oder die Benutzerprofile daraus abgeleitet werden.

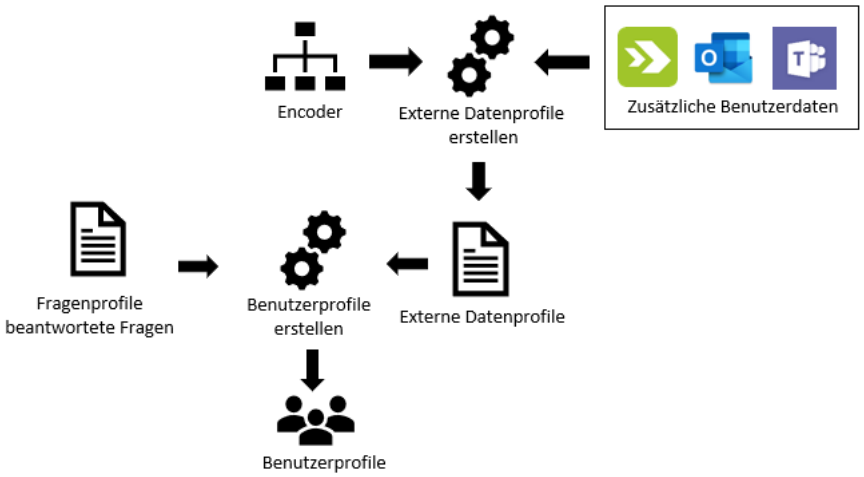


Abbildung 2.4: Ablauf zum Erstellen der Benutzerprofile

Benutzerprofile:
Die Benutzerprofile werden anhand der Fragenprofile berechnet und beinhalten eine eindeutige ID des Benutzers, sowie den berechneten Vektor des Benutzers. Die Qualität dieses Benutzerprofils ist ausschlaggebend für die späteren Empfehlungen des Benutzers. Daher soll zusätzlich zu den intern produzierten Benutzerdaten auch weitere Daten außerhalb des Systems verwendet werden, um zu prüfen wie sich dies positiv auf das RS auswirkt. Ebenfalls kann das Benutzerprofil verwendet werden, um zu überprüfen, welche Benutzer ähnliche Interesse haben und somit ein Clustering Verfahren anzuwenden, um Expertengruppen zu identifizieren. In Abbildung 2.4 ist der Aufbau des Benutzerprofils dargestellt. Dies soll sich aus den beantwortete Fragen des Benutzer, sowie den zusätzlichen Benutzerdaten berechnet werden, welche den gleichen Text-Encoder verwenden, wie bei den Fragenprofile, um die gleiche Grundlage zu haben. Die verwendeten Methoden zur Ermittlung des Benutzerprofils werden in dem Unterkapitel 4.3 betrachtet.

Klassifizierungsmodell:
Dies dient zur Vorhersage, wie gut zwei Profile zueinander passen und leitet daraus die Empfehlungen ab. Die Architektur des neuronale Netz, welches zur Vorhersagen von Empfehlungen verwendet wird in dem Unterkapitel 4.4 genauer erläutert.

Ermittlung ähnlicher Fragen:
Bei der Ermittlung nach den ähnlichen Fragen geht es darum, welche vorhanden Fragen eine hohe inhaltliche Ähnlichkeit mit einer Suchanfrage haben. Dadurch können zum einem identische Fragen ermittelt werden, welche bereits beantwortet wurden und somit schon den Benutzer weiterhelfen. Weiterhin kann dies genutzt werden, um ähnliche Fragen zu empfehlen, welche den Benutzer evtl. auch interessieren könnten.

Weiterleitung von Fragen:
Da auf einer CQA-Plattform viele Fragen entstehen, kann es für den Benutzer unübersichtlich werden, die Fragen zu finden, welche er beantworten kann. Die Weiterleitung von Fragen ist die Hauptaufgabe des RS. Anhand der Fragen- und Benutzerprofile soll geprüft werden, welche Übereinstimmungen am besten passen. Es soll die Benutzer, welche Fragen beantworten wollen dabei unterstützen, sich auf die Fragen zu konzentrieren, welche sie interessieren. Dabei wird unterschieden zwischen zwei Ansichten.

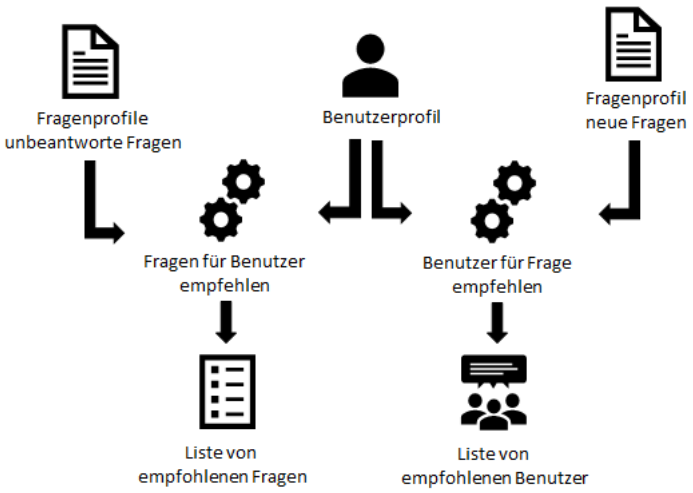


Abbildung 2.5: Empfehlungen zwischen Fragen und Benutzer

- **Benutzer für Frage:**
  Wenn eine neue Frage erfasst wurde, soll diese an Benutzer weitergeleitet werden, welche diese zum einen inhaltlich Beantworten können und erwartet wird, dass dies kurzfristig geschieht. Dabei sollte aber auch berücksichtigt werden, dass wenn regelmäßig Fragen empfohlen werden, welche nicht in das Themengebiet des Benutzers fallen und er diese daher nicht beantworten kann, dass dadurch die Motivation des Benutzers verschwindet den Service zu nutzen. Es sollte daher nicht jeder Benutzer über neue Fragen informiert werden, sondern nur, wenn dieser aufgrund seines Benutzerprofils Interesse aufweist.

- **Fragen für Benutzer:**
  Die andere Sicht ist, wenn ein Benutzer aktiv an der Plattform teilnimmt. Wenn sich ein Benutzer auf der Plattform einloggt, soll diesem eine Liste von (unbeantwortete) Fragen empfohlen werden, welche er am wahrscheinlichsten beantworten kann anhand seines ermittelten Benutzerprofils.

3 GRUNDLAGEN

In diesem Kapitel werden die Grundlagen beschrieben, welche zum Verständnis der Implementierung und der Evaluierung des Recommender System (RS) relevant sind in den folgenden Kapiteln. In dem Abschnitt 3.1 wird auf die Verwendung von künstliches neuronales Netz (KNN) eingegangen, welche später der Kern des RS sind, da diese die Vorhersage machen, ob eine Empfehlung stattfindet. Im Abschnitt 3.2 wird auf die Vorverarbeitungsschritte eingegangen, welche bei der Verwendung von Texten relevant sind, um diese anschließend mit Methoden aus 3.3 weiter verarbeiten zu können und diese in einen Vektor zu transformieren. In dem Abschnitt 3.4 wird darauf eingegangen, wie die Ähnlichkeit zwischen zwei Vektoren berechnet werden kann und in 3.5 auf die Bewertungskriterien, welche für die spätere Evaluierung relevant sind.

3.1 KÜNSTLICHE NEURONALE NETZE

Die Grundlagen von KNN baut auf das Werk von Aggarwal [Agg18] auf. Die Verwendung von KNN beschränkt sich in dieser Arbeit auf die Lösung von Klassifizierungsproblemen. Dies findet dabei sowohl Anwendung in der Erstellung der Fragenprofile in Abschnitt 4.2.2, sowie den eigentlichen RS selbst in Abschnitt 4.4. In beiden Fällen wird das gleiche Ziel angestrebt, indem eine Eingabe in Form eines Vektors entgegengenommen wird, diese durch ein neuronales Netz gesendet wird und als Ausgabe ein Vektor mit der Größe entsprechend der Anzahl der Klassen des Problems und deren Wahrscheinlichkeit für ein Eintreten. Die einfachste Varianten eines neuronalen Netzes wäre in diesem Fall die Verknüpfung zwischen den Eingabevektor und dem Ausgabevektor, wobei jeder Wert eines Vektors als Neuron bezeichnet wird. In der Abbildung 3.1 wird diese einfache Struktur mit einem Neuron als Ausgabe darstellt, welches Perceptron genannt wird. Die Ausgabe erhält eine Verbindung zu jedem Wert des Eingabevektors bzw. Neuron.

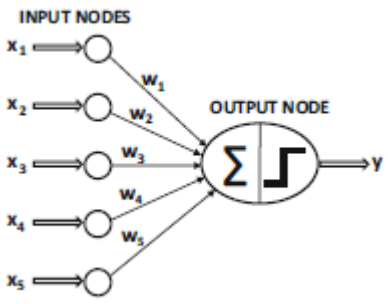


Abbildung 3.1: Aufbau eines Perceptrons [Agg18]

Diese Verbindungen werden mit Gewichten versehen, um die Neuronen aus der Eingabe unterschiedlich zu berücksichtigen bei der Verwendung. Die Werte aus der Eingaben werden dabei jeweils mit ihrem Gewicht multipliziert und anschließend die Summe über diese gebildet, welche die Eingabe des Neuron entspricht. Auf diesen summierten Eingabewert wird anschließend eine Aktivierungsfunktion verwendet und der daraus berechnete Wert ist der Ausgabewert des Neuron. Hierbei gibt es verschiedene Aktivierungsfunktionen, welche verwendet werden. Der Verlauf einigen bekannten Aktivierungsfunktionen wird in Abbildung 3.2 gezeigt. Im Falle des RS, welcher in dieser Arbeit vorgestellt wird, ist es Sinnvoll die Sigmoid Aktivierungsfunktionen für das Ausgabeneuron zu verwenden, da dies den Eingabewert auf einen Wert zwischen 0 und 1 abbildet, welches als Wahrscheinlichkeit gesehen werden kann für eine positive Übereinstimmung bzw. Empfehlung zwischen einer Frage und einem Benutzer.

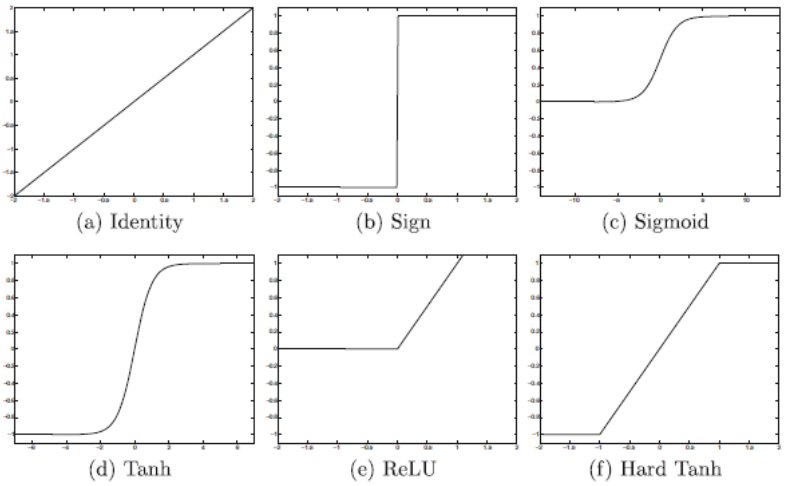


Abbildung 3.2: Verschiedene Aktivierungsfunktionen [Agg18]

Bei der Summierung der Eingabe für ein Neuron, kann zusätzlich noch ein Bias als konstanter Wert einfließen. Der summierte Eingabewert net_j für das Neuron j unter der Betrachtung von n Neuronen als Eingabe und einem Bias b berechnet sich:

$$net_j = \sum_{i=1}^{n} x_i \cdot w_{ij} + b$$

Der Bias verschiebt somit die summierten gewichtete Werte der Eingabeneuronen, wodurch zusätzlich kontrolliert werden kann, wie stark oder schwach die Aktivierung der Ausgabe stattfindet. Bei einem positiven Bias muss der summierte Eingabewert weniger stark sein, damit dieser einen hohen Wert in der Aktivierungsfunktion erreicht, während ein negativer Bias dazu verwendet wird, dass der summierte Eingabewert höher sein muss, damit das Neuron aktiviert wird.

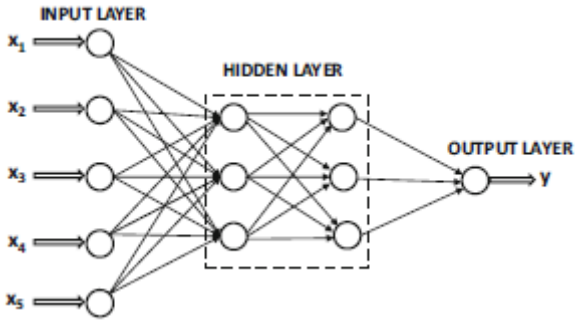


Abbildung 3.3: Neuronales Netz mit zwei versteckten Schichten [Agg18]

Die Erweiterung des einfachen Perceptron ist die Verwendung von zusätzlichen versteckten Schichten zwischen dem Eingabe- und Ausgabevektor des neuronalen Netzes, welches auch als tiefes neuronales Netz genannt wird. In der Abbildung 3.3 ist ein Beispiel eines solches neuronalen Netzes mit zwei zusätzlichen Schichten zwischen der Eingabe und Ausgabe dargestellt. Das Vorgehen ist hierbei das gleiche. Jedes Neuron in der ersten verstecken Schicht bekommt eine Verbindung mit Gewichten zu den Neuronen aus der Eingabeschicht. Es wird jeweils die Summe der gewichteten Werte berechnet und eine Aktivierungsfunktion angewendet, woraus die Ausgabe des Neuron entsteht. Diese bekommen dann jeweils wieder eine Verbindung mit Gewichten zur der zweiten versteckten Schicht, welche dort als Eingabe entgegengenommen wird, ebenfalls eine Aktivierungsfunktion verwendet wird und sich dieses abschließend in der Ausgabeschicht wiederholt.

Die Gewichte zwischen den einzelnen Schichten im Zusammenhang mit der jeweils genutzten Aktivierungsfunktion sind somit ausschlaggebend, welche Ausgabe am Ende des neuronalen Netz entsteht und dementsprechend, wie gut dieses Netz für den Anwendungsfall funktioniert. Ein neuronales Netz muss daher trainiert werden, um zu wissen, welche Gewichte optimal sind. Hierzu muss ein Trainingsdatensatz enthalten sein, welcher die Eingabevektoren enthält, sowie die erwarteten Ausgabenvektoren. Vor Beginn der Trainingsphase werden die Gewichte in der Regel mit zufälligen Werten initial definiert. Anschließend werden die Trainingsdaten in das neuronale Netz gesteckt und berechnet, welche Ausgabe mit den aktuellen Gewichten entsteht. Diese vorhergesagten Werte können dann mit den tatsächlichen Werten abgeglichen werden, um den Fehler des neuronalen Netz zu berechnen. Dieses Verfahren wird auch *Forwardpropagation* genannt, da die Werte einmal vorwärts durch das neuronale Netz gesendet werden. Ziel ist es nun, die Gewichte des neuronalen Netz so anzupassen, dass dieser ermittelte Fehler der Forwardpropagation reduziert wird. Hierzu wird eine Backpropagation (zu deutsch etwa: Fehlerrückführung) durchgeführt, welche nun den Fehler ausgehend von der Ausgabeschicht zurück zur Eingabeschicht propagiert. Dabei wird überprüft, wie stark ein Gewicht einen Einfluss auf den Fehler besitzt und wird entsprechend angepasst, um diesen zu reduzieren, wenn erneut eine Forwardpropagation durchgeführt wird. Dieses Verfahren der

Forward- und Backpropgation kann dabei beliebig oft wiederholt werden, bis sich die vorhergesagten Ausgaben des neuronalen Netz an die tatsächlichen Werte angenähert haben und keine Optimierung mehr möglich ist.

3.2 VORVERARBEITUNG VON TEXTEN

Bei dem Vorverarbeitungsschritt geht es darum, dass Texte entgegengenommen werden und diese von ausgewählten Methoden verarbeitet werden, wodurch die Inhalte des Textes vereinheitlicht oder das hinterlegte Vokabular reduziert werden kann durch das entfernen von nicht-relevanten Informationen. Weiterhin ist es für viele Machine Learning (ML) Methoden notwendig, dass die Texte als eine Liste von Wörter bzw. Tokens vorliegen, anstatt eine einzelne Textkette zur weiteren Verarbeitung. Die hier beschriebene Methoden zur Vorverarbeitung von Texten bauen im wesentlichen auf die Werke von Manning und Schütze [MRS08, Kapitel 2] und Sarkar [Sar16] auf.

Tokenizierung:
Einer der ersten Schritte für die Verarbeitung von Texten ist, diese aufzuteilen in eine Sequenz von Tokens. Dies ist notwendig für die weitere ML Methoden, da diese keine ganzen Texte verarbeiten können, sondern eine Sequenz von Tokens benötigen. Die Tokens entsprechen in der Regel einzelne Wörter aber können auch N-Gramme sein. N-Gramme bedeutet, dass N aufeinanderfolgende Wörter als ein Token gesehen werden. Besonders Sonderzeichen spielen bei der tokenizierung eine große Rolle. So kann z.B. ein Punkt eingesetzt werden als Ende des Satzes, zur Formatierung von Zahlen oder gehört zu einem eingetragenen Namen. Standardmäßig werden für das Aufteilen der Texte das Leerzeichen verwendet und zusätzliche Sonderzeichen, wie z.B. ein Tabstopp oder Zeilenumbruch.

Normalisierung:
In diesem Bereich sollen auf die Normalisierung von Wörtern eingegangen. Diese Methoden sollen gewährleisten, dass Wörter, welche eine abgewandelte Zeichenkette verwenden aber die gleiche Bedeutung haben, diese dennoch gleichgestellt werden. Hierzu werden die Wörter gerne in Kleinschreibung umgewandelt, sowie durch Lemmatisierung oder Stemming eine Rückführung auf die sprachliche Grundform vorgenommen. Dadurch sollen die Inhalte von Texte besser vergleichbar und das verwendete Vokabular auf dem Textkorpus reduziert werden.

Kleinschreibung:
Die Kleinschreibung soll zum einen gewährleisten, dass wenn Wörter am Satzanfang groß geschrieben werden aber innerhalb eines Satzes klein geschrieben werden, diese die gleiche Bedeutung zugeschrieben wird. Ebenfalls ist dies bei Suchanfragen sinnvoll, da dort meistens nicht explizit auf die Groß- und Kleinschreibung geachtet wird.

Lemmatisierung:
Die Lemmatisierung wird angewandt, um verschiedene Wortformen eines Wortes wieder auf seine Grundform zu zurückzuführen. Dies geschieht in der Regel durch die Verwendung eines hinterlegten Lexikon und einer Analyse der Morphologie des Wortes. Bei der Lemmatisierung würden z.B. die englischen Wörter *am*, *are* und *is* zurückgeführt auf die Grundform *be*. Wie bei der Kleinschreibung, werden auch hier Wörter zum einen vereinheitlicht aber gleichzeitig auch die Größe des gesamten Vokabular verkleinert.

Stemming:
Stemming ist eine Alternative zur Lemmatisierung und dient ebenfalls dem Zweck, Wörter zu vereinheitlichen. Gegenüber der Lemmatisierung wird dabei aber ein regelbasierte Veränderung vorgenommen. Als einer der bekanntesten wird hierbei der *Porter's Algorithmus* [Por80] angesehen. Regelbasiert wird in mehrere Phasen die Auswirkung der Anpassung von Suffixe auf das Wort geprüft. So werden z.B. durch die Verwendung von Stemming die Wörter *car*, *cars*, *car's* auf das Wort *car* vereint.
Da Sprache sehr komplex sein kann und Stemming regelbasiert funktioniert, kann dies sehr schnell zur fehlerhaften Normalisierung des Wortes führen. In Information Retrieval (IR) Systemen wird daher oft die Lemmatisierung bevorzugt verwendet, welche auch eine bessere Interpretierbarkeit besitzt. Ein Nachteil von Lemmatisierung kann auftreten, wenn neue unbekannte Wörter im Text verwendet werden, welche noch nicht in dem Lexikon enthalten sind und somit nicht normalisiert werden.

Stoppwörter entfernen:
Wenn ein Wort sehr häufig in einer Sprache verwendet wird und somit regelmäßig in den verwendeten Texten vorkommt, werden diese als Stoppwörter bezeichnet. Diese helfen zwar einen Satz zu gestalten aber haben für den eigentlichen Inhalt des Textes keine Bedeutung. Für das Indexieren und speichern eines Textes werden diese daher gerne vorher entfernt. Das Identifizieren von Stoppwörter geschieht über eine vorgegebene Liste mit Stoppwörtern. Für die englische Sprache wurden z.B. *a, an, and, are, as, at* als Stoppwörter identifiziert. Je nach Anwendungsgebiet kann es aber Sinnvoll sein, auch seine eigene Liste mit Stoppwörtern zu erstellen, welche Domainspezifischer sind. Gewisse Methoden, welche im späteren IR Bereich vorgestellt werden, erkennen diese aber eigenständig und gewichten diese entsprechend weniger für die Bedeutung von Texten.

Sonderzeichen und -Wörter entfernen:
Auf Community Question Answering (CQA) Plattformen werden oftmals Hyptertext Markup Language (HTML) und Cascading Style Sheet (CSS) zur Formatierungen verwendet, um gewisse Passagen im Text hervorzuheben. Diese werden mitgespeichert aber haben nichts mit dem eigentlichen Inhalt des Textes zu tun. Es ist daher Sinnvoll diese zu entfernen, um auf spätere angewandte Verfahren nicht durch Formatierungen zu beeinflussen. So soll-

te z.B. die Ermittlung ähnlicher Fragen, dies durch den Inhalt vorgenommen werden und nicht anhand ähnlicher Formatierungen. Ebenfalls können hierzu die Bereinigung von Sonderzeichen (z.B. Tabstop oder Zeilenumbrüche) gehören, sowie ganze Wörter (z.B. URLs oder Hashtags), um diese separat zu behandeln. Das Erkennen findet dabei durch die Verwendung von regulären Ausdrücken oder den Abgleich mit vorher definierten Listen statt.

Part-of-Speech Tagging:
Beim Part-of-Speech (POS) Tagging (zu Deutsch: Wortart identifizieren) geht es darum, zu ermitteln, welche Wortart (Nomen, Verb, Adverb, etc.) ein Wort im Bezug auf den Text besitzt. Gegenüber der Lemmatisierung, welche die verschiedenen Varianten eines Wortes reduzieren soll, wird POS verwendet, um die Funktion von verschiedenen Wörter innerhalb eines Satzes zu identifizieren. Dies ist nicht nur eine nützliche Information des Wortes selbst, sondern gibt auch Auskunft über benachbarte Wörter. Dies ist hilfreich bei der Verwendung von Named Entity Recognition (NER) aber auch als Vorverarbeitungsschritt, um z.B. den Text auf gewisse POS zu reduzieren. Das erkennen des POS eines Wortes findet über trainierte POS Tagger statt, da ein Wort je nach Satzaufbau unterschiedlich interpretiert werden kann.

Bei den beiden Beispielsätzen

- Ich werde in den Urlaub *fliegen*.
- *Fliegen* werden von dem Geruch von Essen angelockt.

ist das Wort *fliegen* im ersten Satz ein *Verb* und im zweiten Satz ein *Nomen*. Die Wortart ist hierbei abhängig von den benachbarten Wörtern und dem Kontext des Satzes [SW17, S. 649; JM09, S. 151+].

3.3 FEATURE ENGINEERING VON TEXTEN

Nachdem die Vorverarbeitungsschritten durchgeführt wurden, werden in diesem Abschnitt Methoden erläutert, die aus dem vorverarbeiteten und tokenizierten Text, Features extrahieren. Diese haben alle als Ziel, einen Vektor aus dem Text zu erstellen, welche dessen Inhalt repräsentieren und ermöglicht, den Vektor für das Trainieren und Anwenden des später implementierten RS zu verwenden.

3.3.1 *Bag-of-Words*

Der Bag of Words (BoW) Ansatz [MRS08, S. 117; JM09, S. 65] ist einer der einfachsten Methoden, um einen Text als Vektor darstellen zu können. Dabei werden die Häufigkeiten zu jeden Wort in einem Text gespeichert. Die Idee dahinter ist, dass zwei Dokumente mit einem ähnlicher BoW Repräsentation auch einen ähnlichen Inhalt besitzen.

Zu beachten sind bei dem BoW Ansatz allerdings folgende Punkte:

- Bei Verwendung von BoW wird keine Struktur des Textes gespeichert. Ein zweiter Text mit gleicher Worthäufigkeit aber andere Reihenfolge der Wörter wird als identisch angesehen. z.B. *John liebt Marry* ist identisch mit *Marry liebt John*. Es wird bei diesem Ansatz also davon ausgegangen, dass es keinen Zusammenhang zwischen zwei Wörter in einem Text gibt.
- Die Größe des Vektors entspricht dem kompletten Vokabular über alle verwendete Dokumente, damit die Vektoren untereinander die gleiche Größe besitzen und somit vergleichbar werden. Es werden also auch Wörter, welche nicht in einem Dokument verwendet werden mit einer Häufigkeit von 0 gespeichert.

Da beim BoW keine Struktur gespeichert wird, kann sich das entfernen von Stoppwörter zur Nutzen gemacht werden, um die Größe des Vokabulars und damit verbundene Vektorgröße zu reduzieren.

	also	bag	book	he	his	in	like	of	word
He likes bag of words. Also he likes books.	1	1	1	2	0	0	2	1	1
He likes the book in his bag.	0	1	1	1	1	1	1	0	0

Tabelle 3.1: Beispielberechnung anhand von BoW

In Tabelle 3.1 werden zwei Dokumente als BoW Vektor dargestellt. Die Wörter werden entsprechend ihrer Häufigkeit bewertet und es wird die maximale Ausprägung über den kompletten Corpus benötigt, wodurch auch nicht vorhandene Wörter mit Nullen aufgefüllt werden müssen. Weiterhin ist auch zu sehen, dass keine Struktur mehr der Texte vorhanden ist, da die Wörter in beliebiger Reihenfolge in den Vektor gespeichert werden. Da ein Großteil des verwendeten Vokabulars übereinstimmt, könnten diese als ähnliche Texte angesehen werden.

3.3.2 *TF-IDF*

Bei dem BoW wird die Häufigkeit von Wörtern in einem Text beachtet. Bei großen Texten entsteht somit auch eine höhere Häufigkeit der Wörter, was ein Vergleich der Texte erschwert. Daher ist es sinnvoll, diese zu normalisieren. Mit der Anwendung der term frequency - inverse document frequency (Tf-idf) Gewichtung soll ermittelt werden, welche Wörter für ein Dokument ausschlaggebend sind gegenüber den anderen Dokumenten. Ebenfalls werden mit diesem Ansatz automatisch auch Stoppwörter ermittelt, welche z.B. in einem Großteil der Dokumente vorkommen und somit keine Aussagekraft für den kompletten Textkorpus besitzen.

In einem ersten Schritt wird auch bei dieser Methode die Worthäufigkeit (term frequency (TF)) entsprechend des BoW Ansatz berechnet. Darauf aufbauen können dann die gewichtete und normalisierten Tf-idf Werte berechnet werden [NM12, S. 49–50; JM09, S. 112–115; MRS08, Kapitel 6].

Zuerst sollte die Worthäufigkeit für jedes Dokument normiert werden, um große und kleine Texte gleichzustellen und besser untereinander vergleichbar zu machen. Dies geschieht, indem mit folgender Formel berechnet wird, wie hoch die Häufigkeit eines Terms gegenüber den restlichen Termen des gleichen Dokument ist.

$$w_{dk} = \frac{tf_{dk}}{\sqrt{\sum_{i=1}^{t}(tf_{di})^2}} \tag{3.1}$$
$$tf_{dk} = \text{Häufigkeit von Term } k \text{ in Dokument } d$$

Dies wird nun noch mit der inverse document frequency (idf) $\frac{N}{n_k}$ ergänzt, um die Bedeutung des Wortes über den kompletten Corpus zu bekommen, indem überprüft wird, wie hoch der Anteil der Dokumente ist, welche den Term k verwenden gegenüber der Anzahl aller Dokumente N im Korpus. Dadurch entsteht ein Wert, welcher eine bessere Interpretierbarkeit gegenüber den ursprünglichen Wert besitzt, da dieser durch die Normalisierung einen Wertebereich zwischen 0 und 1 besitzt und die Wichtigkeit innerhalb des Dokuments und des kompletten Corpus ersichtlich ist. Dies vereinfacht das Vergleichen zwischen verschiedener Dokumente auf ihre Ähnlichkeit.

$$w_{dk} = \frac{tf_{dk} * \log\frac{N}{n_k}}{\sqrt{\sum_{i=1}^{t}(tf_{di}\log\frac{N}{n_i})^2}} \tag{3.2}$$
$$N = \text{Anzahl der Dokumente}$$
$$n_k = \text{Anzahl der Dokumente, welche den Term } k \text{ enthalten}$$

Durch die zusätzliche Verwendung des Logarithmus kann eine Skalierung der idf Gewichtung vorgenommen werden, wodurch seltene Wörter nicht zu stark gewichtet werden und Wörter, welche in jedem Dokument vorhanden sind (z.B. Stoppwörter) auf Null reduziert und entfernt werden, da diese keinen Mehrwert für den Textkorpus besitzen.

In Tabelle 3.2 wird ein Beispiel für die Berechnung von Tf-idf dargestellt. Für die drei Dokumente wurde mit dem BoW die Worthäufigkeit *TF* ermittelt. Dokument 1 hat hier sehr hohe Werte, was auf einen große Text hindeutet, während Dokument 3 mit seinen kleinen Werte nur ein kleiner Textabschnitt sein wird. Nach der Normalisierung sind die Dokumente untereinander besser vergleichbar, da die Textgröße keine Rolle mehr spielt. Mit der anschließenden Berücksichtigung der idf ist ersichtlich, durch welche Wörter die jeweiligen Dokumente spezifiziert sind. Durch den Logarithmiertem idf wird das Wort *Library* auf 0 gesetzt, da dies in jeden Dokument

Dokument 1	tf	tf normiert	mit idf	mit $log(idf)$
Library	40	0.74	0.59	0.00
Book	30	0.56	0.67	0.83
Paper	20	0.37	0.45	0.55

Dokument 2	tf	tf normiert	mit idf	mit $log(idf)$
Library	10	0.81	0.47	0.00
Article	6	0.49	0.84	0.97
Paper	4	0.32	0.28	0.24

Dokument 3	tf	tf normiert	mit idf	mit $log(idf)$
Abstract	1	0.19	0.37	0.48
Book	5	0.96	0.92	0.88
Library	1	0.19	0.12	0.00

Tabelle 3.2: Beispielberechnung anhand von Tf-idf

vorkommt. Dies kann zwar für ein einzelnes Dokument interessant sein aber für Vergleiche innerhalb des gesamten Corpus spielt dies keine Rolle und kann entfernt werden.

3.3.3 *Word Embedding mit Word2Vec*

Durch die Verwendung von Word Embedding (dt. Worteinbettung), werden Wörter eines Text Corpus auf einen vorgegebene n-dimensionalen Vektor dargestellt. Dabei wird nicht nur das Wort alleine betrachtet, sondern auch den Kontext eines Wortes innerhalb der Texte, indem auch die benachbarten Wörter berücksichtigt werden. Dies ermöglicht es, dass Wörter untereinander verglichen werden können, welche eine ähnliche Semantik besitzen, was durch einen ähnlichen Vektor repräsentiert wird [Ben+03; GPJ18].

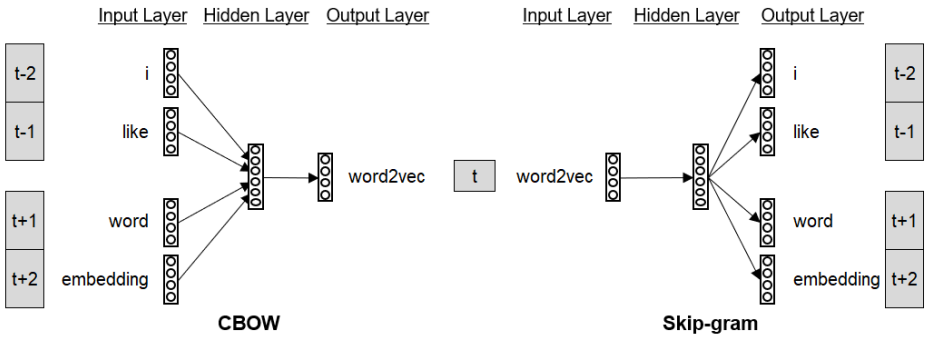


Abbildung 3.4: Architektur der beiden Word2Vec Modelle CBOW und Skip Gram

Word2Vec (Word-to-Vector) ist ein verbreiteter Ansatz zum erstellen von Word Embedding, welcher von Mikolov et. al [Mik+13b; Mik+13a] eingeführt wurde. Es handelt sich hierbei um ein unüberwachtes Lernen, welches ein neuronales Netz als Klassifizierungsmodell verwendet. Es wird dabei zwischen zwei verschiedene Ansätze zum Trainieren des Modells unterschieden, welche in Abbildung 3.4 abgebildet sind. Beide Varianten verwenden jeweils ein Wort und dessen benachbarten Wörter aus dem Trainingsdaten. Die Anzahl der benachbarten Wörter, welche berücksichtigt werden sollen, kann als Parameter vorgegeben werden.

Bei dem Skip Gram Modell wird ein Modell trainiert, welches ein einzelnes Wort als Eingabe verwendet und daraus als Ausgabe die benachbarten Wörter Vorhersagen soll. Das CBOW Modell hingegen vertauscht die Eingabe und Ausgabe und trainiert somit ein neuronales Netz, welches anhand benachbarten Wörtern das fehlende Wort vorhersagen soll. Die Eingabe- und Ausgabe-Vektoren entsprechen jeweils der Größe des verwendeten Vokabulars. Diese sind in jedem Trainingslauf One-Hot-Encoded. Dies bedeutet, dass bei den Vektoren nur jeweils das verwendete Wort eine 1 als Wert enthält, während die restlichen auf 0 gesetzt werden. Der Hidden-Layer entspricht der vorgegebenen Größe des gewünschten Word Embeddings, welches der späteren Vektorgröße entspricht, auf den die Wörter reduziert werden soll. Der Vektor eines Wortes entspricht dabei der Gewichtungsmatrix zwischen dem Neuron des Wortes und dem Hidden Layer. [GPJ18; JM09, S. 118+]

3.3.4 *Text Embedding mit Doc2Vec*

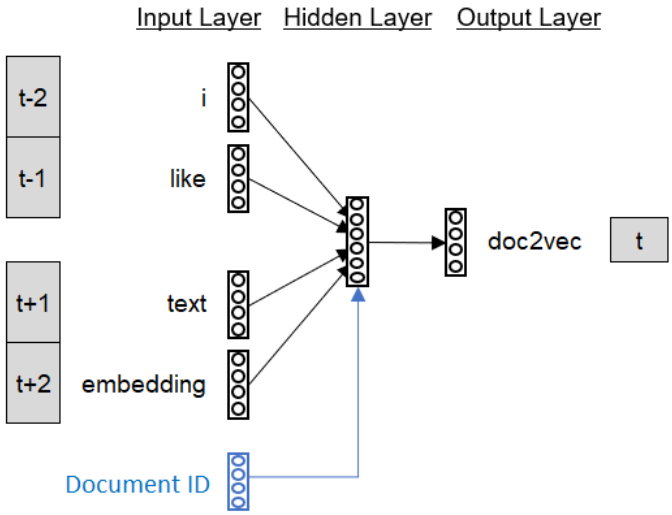


Abbildung 3.5: Erweiterung des Word2Vec CBOW Ansatz zu Doc2Vec PV-DM

Bei der Verwendung von Word2Vec steht im Vordergrund Wörter als Vektoren darzustellen, um diese untereinander vergleichen zu können. Für den

späteren Anwendungsfall in dieser Arbeit ist es allerdings notwendig, komplette Textabschnitte als Vektoren darzustellen zu können. Hierzu gibt es einige Methoden, welche die ermittelten Wortvektoren eines Textes mit Hilfe von Word2Vec zu einen verallgemeinerten Textvektor umwandeln, indem der Mittelwert der einzelnen Vektoren berechnet wird oder durch zusätzliche Berücksichtigung der Tf-idf Werte des Textes. [CJMS17; BMA16].

Um komplette Texte direkt als Vektor zu trainieren und darstellen zu können, wurde von Mikolov et. al [Mik+13a; LM14] eine Erweiterung des Word2Vec Ansatzes vorgestellt, welches unter dem Namen Paragraph2Vec (Paragraph-to-Vector) bzw. Doc2Vec (Documents-to-Vector) bekannt ist. Es wird ein zusätzlicher Eingabevektor (Dokumentvektor genannt) verwendet, welcher die Dokumente eindeutig identifizieren kann. Dieser wird im Training berücksichtigt, indem sich dieser nicht ändert bei der Verwendung von Wörtern, die in dem Dokument verwendet werden. Dies wird als eine Erinnerung verwendet, welche die verwendete Wörter in den Kontext des Dokumentes stellen, wodurch auch der Name "Distributed Memory"(dt. Verteilter Speicher) entsteht.

In Abbildung 3.5 ist diese Distributed Memory Model of Paragraph Vectors (PV-DM) Erweiterung dargestellt. Es wird für ein eindeutige Dokumenten ID für die Wörter *i*, *like*, *text* und *embedding* das neuronale Netz trainiert, dass es als fehlendes Wort *doc2vec* vorhersagt. Nach dem Training kann die ausgehende Gewichtungsmatrix des Dokumentvektors verwendet werden, um den Vektor eines kompletten Dokuments zu erhalten. Dokumente mit einem ähnlichen Inhalt würden hierbei auch einen ähnlichen Vektor erhalten. Ebenfalls kann dieses Modell auch verwendet werden, um die Word Embeddings aus Word2Vec zu erhalten, welche sich wie bereits bei dem Word2Vec Verfahren aus der Gewichtung des Wortes im Hidden-Layer ergeben.

Eine Herausforderung des Doc2Vec Modells ist allerdings, dass diese nur die Dokumente als Vektor direkt abrufen kann aus der Gewichtungsmatrix, welche während der Trainingsphase auch in dem Dokumentenvektor verwendet wurden. Um dennoch unbekannte Dokumente als Vektor darstellen zu können, kann eine *Inference stage* (dt. Schlussfolgerung) verwendet werden. Hierbei wird das bereits trainierte neuronale Netz verwendet und der Dokumentenvektor um das neue unbekannte Dokument ergänzt. Die bestehenden Gewichte werden dabei festgesetzt, was bedeutet, dass diese nicht mehr verändert werden dürfen, um die gleichen Word Embeddings zu verwenden. Es wird lediglich die Gewichtung des neuen Dokument als trainierbar deklariert, wodurch dies durch eine zusätzliche Trainingsphase und dem Gradientenverfahren entsprechend der Fehlerreduzierung angepasst werden und nach Abschluss dieser Trainingsphase der Vektor des Dokument abgeleitet werden kann.

3.4 ÄHNLICHKEITSMASSE

Nachdem die Dokumente in Vektoren umgewandelt wurden, können diese genutzt werden, um die Ähnlichkeit von Dokumenten zu bestimmen. Im Bereich des Natural Language Processing (NLP) ist die Kosinus-Ähnlichkeit am meisten verbreitet. Diese verwendet das Skalarprodukt aber normalisiert dies noch durch die Berücksichtigung der Vektorlängen, wodurch ein Wert für die Ähnlichkeit zwischen -1.0 und 1.0 entsteht, wodurch eine einfachere Interpretation des Ergebnis entsteht gegenüber dem Skalarprodukt. Bei Verwendung der Tf-idf sind die Werte des Vektors bereits normalisiert, wodurch das Skalarprodukt und die Kosinus-Ähnlichkeit die gleichen Ergebnisse berechnen. Bei Verwendung von BoW oder Doc2Vec ist dies aber nicht der Fall und die berechneten Werte unterscheiden sich. Das Skalarprodukt $dot(\vec{a}, \vec{b})$ und die Kosinus-Ähnlichkeit $\cos(\vec{a}, \vec{b})$ zwischen zwei Vektoren $\vec{a}$ und $\vec{b}$ berechnet sich wie folgt [JM09, S. 111–112]:

$$
\begin{aligned}
dot(\vec{a}, \vec{b}) &= \vec{a} \cdot \vec{b} = \sum_{i=1}^{N} a_i b_i \\
\cos(\vec{a}, \vec{b}) &= \frac{\vec{a} \cdot \vec{b}}{\mid \vec{a} \mid \cdot \mid \vec{b} \mid} = \frac{\sum_{i=1}^{N} a_i b_i}{\sqrt{\sum_{i=1}^{N} a_i^2} \sqrt{\sum_{i=1}^{N} b_i^2}} \\
N &= \text{Anzahl der Dimensionen}
\end{aligned}
\tag{3.3}
$$

Die Kosinus-Ähnlichkeit hat einen Wertebereich zwischen -1 und 1 mit folgender Bedeutung:

- -1.0: Die beiden Vektoren liegen genau in entgegengerichteter Richtung.
- 0.0: Die beiden Vektoren sind unabhängig (Orthogonalität). Es besteht somit keine Ähnlichkeit.
- 1.0: Die beiden Vektoren zeigen in die gleiche Richtung. Sie sind somit identisch.

	Abstract	Article	Book	Library	Paper
Dokument 1	0.00	0.00	0.83	0.00	0.55
Dokument 2	0.00	0.97	0.00	0.00	0.24
Dokument 3	0.48	0.00	0.88	0.00	0.00

Tabelle 3.3: Beispiel Tf-idf Werte zur Berechnung der Kosinus-Ähnlichkeit

$$
\begin{aligned}
dot(Dok1, Dok2) = \cos(Dok1, Dok2) &= 0.13 \\
dot(Dok1, Dok2) = \cos(Dok1, Dok3) &= 0.73 \\
dot(Dok1, Dok2) = \cos(Dok2, Dok3) &= 0.00
\end{aligned}
$$

Bei der Berechnung der Kosinus-Ähnlichkeit anhand der Tf-idf Werte in 3.3, ergibt sich zwischen dem *Dokument 2* und *Dokument 3* keine Ähnlichkeit, da keine Übereinstimmung von verwendeten Wörtern vorliegt, während das *Dokument 1* und *Dokument 3* eine Ähnlichkeit von 73% besitzen. Bei Betrachtung der Ausgangsabbildung 3.2 ist ersichtlich, dass diese beide Dokumente sich auf *Book* konzentrieren. Da es sich um normalisierte Tf-idf Werte handelt, würde das Skalarprodukt die gleiche Werte berechnen in diesem Beispiel.

3.5 BEWERTUNGSKRITERIEN

	Relevant	**Nicht relevant**
Gefunden	*Richtig positiv*	*Falsch positiv*
Nicht gefunden	*Falsch negativ*	*Richtig negativ*

Tabelle 3.4: Konfusiosnmatrix

Zur Bewertung der Ergebnisse wird ein binäres Klassifikationsproblem verwendet. Es wird dabei eine Vorhersage gemacht, ob ein Benutzer ein Interesse gegenüber einer Frage hat oder nicht. Dies wird während der Trainingsphase mit den tatsächlichen Werten abgeglichen, wodurch eine Konfusionsmatrix gemäß Tabelle 3.4 entsteht.

Die Werte der Tabelle haben dabei folgende Bedeutung:

- **Richtig positiv:** Es wurde ermittelt, dass ein Interesse besteht, welches auch zutrifft.
- **Falsch positiv:** Es wurde ermittelt, dass ein Interesse besteht, welches aber nicht zutrifft.
- **Falsch negativ:** Es wurde ermittelt, dass kein Interesse besteht, obwohl doch ein Interesse besteht.
- **Richtig negativ:** Es wurde ermittelt, dass kein Interesse besteht, welches tatsächlich auch nicht besteht.

Aus diesen vier Fällen können drei Metriken abgeleitet werden, welche zur Bewertung der späteren verwendeten Methoden im Zusammenhang mit dem RS als Klassifizierer genutzt werden können.

$$\text{Trefferquote} = \frac{\text{Richtig positiv}}{\text{Richtig positiv + Falsch negativ}}$$

$$\text{Genauigkeit} = \frac{\text{Richtig positiv}}{\text{Richtig positiv + Falsch positiv}}$$

$$\text{F-Maß} = \frac{\text{Trefferquote * Genauigkeit}}{\text{Trefferquote + Genauigkeit}}$$

Die Trefferquote gibt den Anteil an, wie viele der relevanten Werte tatsächlich gefunden wurden. Die Genauigkeit berechnet den Anteil der korrekt ermittelten Werten anhand der Summe der durch das RS gefunden Werte. Das F-Maß hingegen ist das harmonische Mittel der Trefferquote und Genauigkeit und kann als kombinierte Maßeinheit genutzt werden [Kub15].

4 IMPLEMENTIERUNG

In diesem Kapitel soll erläutert werden, welche Methoden verwendet wurden, um ein Recommender System (RS) zu Erstellen und diesem im anschließenden Kapitel zu evaluieren. Die Implementierung wurde mit Python[1] Version 3.6.6 durchgeführt. Die genaueren verwendeten Python Module, sowie Versionen sind in der Anlage A aufgelistet.

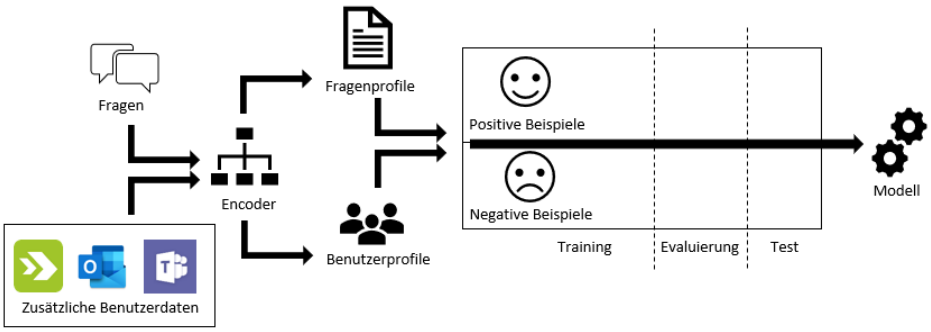


Abbildung 4.1: Übersicht für die Implementierung des Systems

In Abbildung 4.1 ist eine Kurzübersicht der Implementierung dargestellt, welche ausgehend von den Textcorpus der Fragen und zusätzlicher Benutzerdaten, diese durch einen Text-Encoder sendet, welcher aus der Vorverarbeitung des Textes und anschließende Transformation in einen Vektor besteht. Dadurch können die Frage- und Benutzerprofile abgeleitete werden, sowie durch das Zusammenfügen dieser, die positiven Daten (Benutzer interessiert diese Frage) und negative Daten (Benutzer interessiert diese Frage nicht) erstellt werden. Diese werden dann zeitlich aufgeteilt, um ein Modell zu trainieren und zu evaluieren. Das fertige Modell kann dann in den produktiven Einsatz verwendet werden aber sollte regelmäßig neu trainiert und evaluiert werden, bei steigender Anzahl von Fragen und Antworten.

4.1 VORVERARBEITUNG DER FRAGETEXTE

Die Vorverarbeitungsschritte sollen die Textinhalte bereinigen, um nicht relevante Informationen und diese anschließend in ein Format bringen für die weitere Verwendung von Machine Learning (ML) Verfahren. Verwendet wurde die Python Bibliothek Spacy[2] mit dem dort hinterlegten englischen Sprachmodell, da sowohl bei der Firma PRODYNA SE, sowie für die spätere Evaluierung verwendeten Testdaten von Stackoverflow auf einer engli-

1 https://www.python.org
2 https://spacy.io

schen Kommunikation aufbauen. Für andere Anwendungsfälle unterstützt dies aber auch weitere Sprache (z.B. Deutsch, Spanisch oder Französisch), sowie ein unabhängiges Mehrsprachiges Sprachmodell. Spacy kann im Zusammenhang mit dieser Arbeit genutzt werden, um z.B. das Extrahieren von Sätzen aus einem Text, das Erkennen von Part-of-Speech (POS) Tags, der Lemmatizierung und entfernen von Stoppwörtern, sowie das abschließende umwandeln in Tokens vorzunehmen.

Listing 4.1: Preprocessing Methode

```
def preprocessing(text, url=True, html_tags=True, hashtags=True,
    sentences=False, special_characters=True, stopwords=True, nummeric=
    True, pos=None, lemmatize=True, lowercase=True, tokenize=True)
```

Diese Schritte wurde in eine eigene Methode 4.1 zusammengefasst implementiert, um zu steuern, welche Schritte ausgeführt werden für einen übergebenen Text. Dies ermöglicht es, zu überprüfen, ob und wie sich die Auswahl von verschiedenen Vorverarbeitungsschritten auf das spätere Ergebnis auswirken. Die übergebene Parameter haben dabei folgende Bedeutung:

- **url**: Das Erkennen und Entfernen von URL Adressen
- **html tags**: Das Erkennen und Entfernen von HTML-Tags, welche zur Formatierung dienen.
- **hastags**: Hashtags werden entfernt, da diese separat behandelt werden sollen.
- **sentences**: Ob der Text in seine einzelnen Sätze aufgeteilt und behandelt werden soll oder als ganzer Text. Dies macht z.B. bei Verwendung von Word Embedding wie Word2Vec Sinn, welches nur die Wortbeziehungen innerhalb von Sätzen betrachten soll.
- **special characters**: Das entfernen von Tabstopps, Zeilenumbrüche und redundanten Leerzeichen.
- **stopwords**: Das Entfernen von erkannten Stoppwörtern.
- **nummeric**: Das Entfernen von Zahlen im Text.
- **pos**: Das Einschränken auf eine übergebene Liste von POS-Tags[3]
- **lemmatize**: Ob eine Lemmatizierung der Wörter zu ihrem Stammwort stattfinden soll.
- **lowercase**: Der Text bzw. Wörter werden in Kleinbuchstaben umgewandelt.
- **tokenize**: Ob der verarbeitete Text am Ende wieder als eine Zeichenkette oder eine Liste mit Wort-Tokens zurückgegeben werden soll.

3 Zum ermitteln der POS-Tags wurde Spacy verwendet. Dies unterstützt im Englischen die folgenden POS https://spacy.io/api/annotation#pos-universal

4.2 ERSTELLUNG DER FRAGEPROFILE

Für die Ermittlung der Frageprofile werden der Titel der Frage und der eigentliche Fragetext zusammengefasst als ein Text, darauf die Vorverarbeitungsschritte angewandt und anschließend noch um die hinterlegte Hashtags ergänzt. Der daraus resultierende vorverarbeitete Textcorpus wird anschließend verwendet, um gemäß den Methoden im Abschnitt 3.3 ein Modell zu trainieren, welches diese in einen Vektor umwandeln kann. Dieser Vektor ist das ermittelte Frageprofil und dient als Ausgangspunkt für die weitere Verwendung.

Mit steigender Anzahl von Fragen in dem System, bekommen diese Modelle auch mehr Informationen über den verwendeten Textcorpus und können das Frageprofil somit auch detailierter entsprechend ihrer Charakteristik darstellen. Wenn z.B. bei der Verwendung der term frequency - inverse document frequency (Tf-idf) Methode neue Fragen wiederkehrend ein gleiches Schlüsselwort (z.B. Produktname) im Text verwendet, so wird dieses Wort aufgrund der inverse document frequency (idf) Eigenschaft an Bedeutung verlieren, da dies vermehrt im Textcorpus verwendet wird und somit weniger Aussagekraft für eine einzelne Frage besitzt. Dies wiederum bedeutet, dass die Frageprofile keine statische Vektoren sind und sich über die Zeit verändern können mit dem eintreffen neuer Fragen. Um zu verhindern, dass die zeitliche Performance zur Ermittlung einer Empfehlung bei neu erstellten Fragen abnimmt, da das komplette System sich neu Trainiert, werden diese auf dem aktuellen vorhandenen Modell aktuallisiert. Die Modelle werden somit nich mit jeder neuen Frage neu trainiert, sondern sollten sich nach einem vorgegebenen Zeitraum (z.B. wöchentlich) oder nach dem Übersteigen von noch nicht im Modelltraining berücksichtigte neue Fragen (z.B. x% neue Fragen) aktuallisieren. Das Frageprofil besitzt dadurch eine Gültigkeit bis zum nächsten kompletten neuen Training des RS, wodurch der hinterlegte Vektor nur einmal ermittelt und gespeichert werden muss. Dies soll die Performance während des Einsatzes des RS erhöhen, da diese nicht immer neu berechnet werden müssen, sondern direkt abgegegriffen werden können.

4.2.1 *Tf-idf*

Als Ausgangsbasis wurde sich für ein Tf-idf Modell entschieden, welches die Fragen als Vektor darstellen. Um die Größe des Vektor zu reduzieren, wurde die Vorverarbeitungsmethode angewandt, um URL-Adressen, HTML-Tags, Stoppwörter zu entfernen und anschließend das Vokabular per Lemmatizierung und Kleinschreibung zu vereinheitlichen. In der späteren Evaluierung wurde weiterhin noch überprüft, wie sich die Einschränkung auf bestimmte POS-Tags (Nomen, Verb, Adjektiv) auf die Größe des Vokabular (Vektorgröße) und die Ergebnisse auswirkt. Ebenfalls wird noch geprüft, wie sich die Anwendung von Luhns Einschränkungsmethode [LS68], welche in

Abbildung 4.2 dargestellt ist die Ergebnisse beeinträchtigen. Diese stellt da, welche Wörter am meisten Aussagekraft besitzt. Wörter welche in den Dokumenten zu häufig vorkommen, haben für ein einzelnes Dokument wenig bedeutung, genauso wie Wörter, welche zu selten vorkommen. In der Abbildung wird durch die Glockenkurve dargestellt, in welchen Häufigkeitsbereich die Wörter am meisten Aussagekraft besitzen. Daher können Wörter, welche sich außerhalb der Bergrenzungen (Upper- und Lower cut-off) befinden vernachlässigt werden. Gesteuert werden kann dies über die folgenden Parameter:

- **Minimum Dokumenthäufigkeit:** Wie oft ein Wort mindestens in den Dokumenten vorkommen soll, bevor es berücksichtigt wird.
- **Maximum Dokumenthäufigkeit:** Wie oft ein Wort maximal in den Dokumenten vorkommen darf, bevor es nicht mehr berücksichtigt wird.
- **Maximale Vokabulargröße:** Einschränkung auf Wörter, entsprechend der sortierten Worthäufigkeit über den kompletten Corpus verteilt.

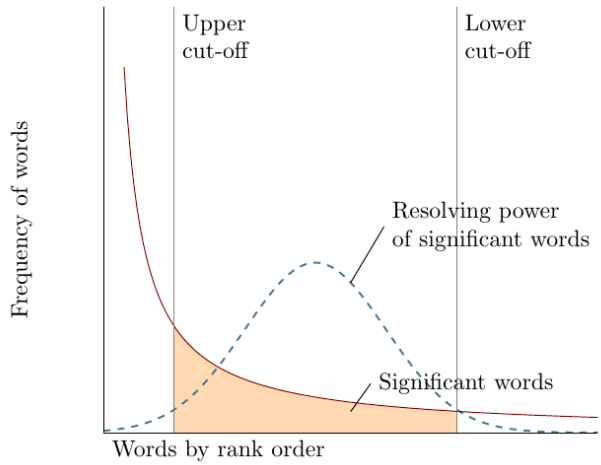


Abbildung 4.2: Zipfsches Gesetz mit Luhns empfohlener Einschränkungsmethode [LS68]

	Abstract	Article	Book	Library	Paper
Dokument 1	0.00	0.00	0.83	0.00	0.55
Dokument 2	0.00	0.97	0.00	0.00	0.24
Dokument 3	0.48	0.00	0.88	0.00	0.00

(Dok.1, Book) = 0.83
(Dok.1, Paper) = 0.55
(Dok.2, Article) = 0.97
(Dok.2, Paper) = 0.24
(Dok.3, Abstract) = 0.48
(Dok.3, Book) = 0.88

Abbildung 4.3: Komprimiertes Format einer dünnbesetzten Matrix

Bei der Verwendung der Tf-idf Methode entsteht eine Matrix der Größe DxV mit D der Anzahl der Dokumente und V der Größe des verwendeten Vokabulars. Da ein Dokument nur einen Bruchteil des hinterlegten Vokabulars verwendet, entsteht eine dünnbesetzte Matrix, welche viele Nullen aufweist, welche dennoch gespeichert werden müssen und somit einen

erhöhten Speicherbedarf des Vektors benötigt. Um dies entgegenzuwirken, werden die Werte in einem komprimierten Format gespeichert, welches nur die relevanten Koordinaten mit Wert speichert. Dies ist ein Abbildung 4.3 dargestellt, welches eine dünnbesetzte Matrix mit 12 Werten in ein komprimiertes Format mit nur noch 6 benötigten gespeicherten Werten umwandelt. Für die Anwendung späterer Machine Learning Verfahren können diese wieder umgewandelt werden für einen benötigten Vektor, indem dieser wieder in einen dünnbesetzen Vektor transformiert wird und nicht vorhandene Koordinaten mit Nullen aufgefüllt werden.

4.2.2 *Doc2Vec*

Zusätzlich wurde sich als Alternative zu Tf-idf für die Doc2Vec Methode entschieden, um die Fragen in einen entsprechenden Vektor zu transformieren. Dies soll ermöglichen, dass nicht nur die Worthäufigkeit innerhalb der Dokumente verwendet wird, sondern auch der Aufbau eines Dokumentes und die Beziehungen zwischen den Wörtern berücksichtigt wird, um deren Semantik besser zu verstehen. Ein weiterer Vorteil des Doc2Vec ist, dass die Fragen auf einen kleineren Vektor dargestellt werden gegenüber Tf-idf, wodurch eine schnellere Laufzeit bei der späteren Verwendung der Vektoren zu Berechnungen gegeben ist. Der Nachteil welcher daraus entsteht ist, dass die Vektoren nicht mehr direkt interpretiert werden können, da es nur noch eine Koordinate im N-Dimensionalen Raum ist und nur noch geprüft werden kann, wie die Distanz zu anderen Vektoren ist, um entsprechend ähnliche Vektoren bzw. Fragen zu ermitteln.

Ebenfalls sollte beachtet werden, dass die Ergebnisse von Doc2Vec abhängig sind von der Anzahl der Dokumente zum Trainieren des Modells. Je größer der verwendete Textcorpus ist zum trainieren, desto besser kann das Doc2Vec Modell auch die Semantik der Texte lernen. Dies hat auch eine Auswirkung auf die Auswahl der Vektorgröße, da bei kleineren Datenmengen und Vokabular ein kleinerer Vektor ausreichend sein kann, während mit steigender Menge und je nach Anwendungsfalls ein größerer Vektor empfehlenswert ist [Mel+16].

Um der Herausforderung mit den Datenmenge entgegenzuwirken, kann das Doc2Vec Modell nicht nur mit den Fragentexten trainiert, sondern ebenfalls die Antworttexte mit einbezogen werden für ein umfangreicheres Verständnis des Textkorpus. Zusätzlich muss für das Trainieren des Doc2Vec Modells eine Fenstergröße angegeben werden, um zu definieren wie viele benachbarten Wörter mit einbezogen werden, um ein Wort vorherzusagen. Während ein kleiner Wert ein Modell mehr daraufhin trainiert Synonyme für das Wort zu finden, werden bei einer höher gewählten Fenstergröße mehr Attribute zu dem Thema berücksichtigt, in dessen Kontext das Wort genutzt wird [LG14].

Auch bei dem Doc2Vec Verfahren ist es sinnvoll Vorverarbeitungsschritte anzuwenden, um ein einheitliches Vokabular zu verwenden (z.B. Lemmatizierung und Kleinschreibung). Allerdings muss hierbei beachtet werden, dass durch das Entfernen von Stoppwörtern und Filterung von POS-Tags, dies auch eine Auswirkung auf die gewählte Fenstergröße für benachbarte Wörter hat, da Wörter entfernt werden und diese somit näher zusammenrücken.

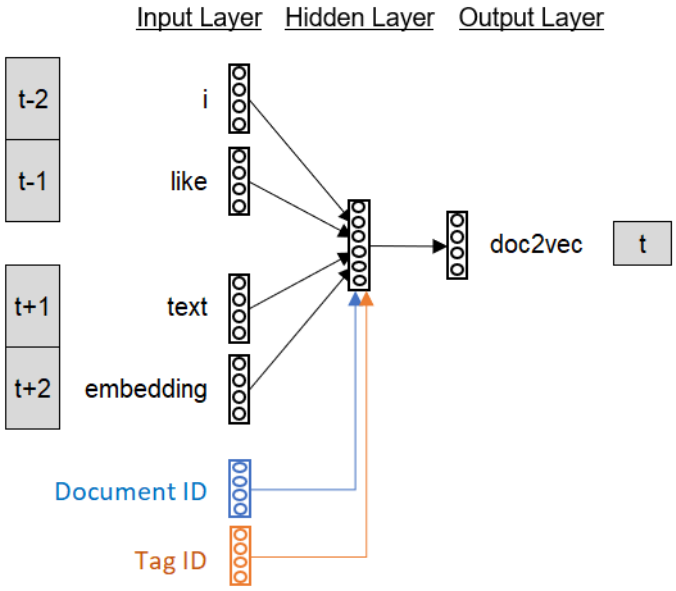


Abbildung 4.4: Doc2Vec PV-DM Modell mit zusätzlichen Trainingsvektor

Während des Trainings des Word2Vec Modells wird jeweils eine eindeutige ID der Frage berücksichtigt. Diese Vorgehen ist jedoch nicht beschränkt auf die Verwendung dieser ID. So kann dieser Vektor, welcher als Erinnerung dient, zu welchen Dokument die aktuell verwendete Wörter gehören beliebig erweitert werden [Shp19]. Im Falle der CQA Plattform, kann entsprechend der Abbildung 4.4 z.B. die Information der dazugehörigen Hashtags als zusätzliches Vektor mitgegeben werden als weiteres Kategorisierungsmerkmal. Für die Evaluierung wurden die folgende Varianten verwendet, um zu prüfen, wie sich diese auf die Ergebnisse auswirken:

- **Eindeutige ID:** Dies ist die ursprüngliche Methode von Doc2Vec. Dabei wird jedem Dokument eine eindeutige Kennung mitgegeben. Dies kann eine Fortlaufende Nummer sein oder die im CQA System hinterlegten Fragen- und Antworten-ID.
- **Hashtags der Frage:** Da in der Regel bei Fragen auch Hashtags hinterlegt werden, welche die Fragen kategorisieren, können diese auch für Doc2Vec verwendet werden. Dadurch wird ein Bezug zwischen einem Hashtag und den Texten, die diesen Hashtag verwenden hergestellt. Da für das Trainieren sowohl die Fragen, als auch Antworten verwendet werden können, werden bei den Antworten auch die Hashtags

verwendet, welche für die beantwortete Frage hinterlegt sind. Diese Hashtags werden allerdings nur zusätzlich zu dem Verfahren mit der eindeutigen ID verwendet wie in Abbildung 4.4 dargestellt, da der Bezug zum eigentlichen Dokument bestehen bleiben soll.

- **Benutzer der Antworten:** Als dritte Variante wird noch getestet, wie sich die hinterlegten Benutzer der Antworten zu einer Frage auf das Ergebnis auswirken. Dabei werden bei den Frage alle Benutzer der Antworten mitgegeben, während bei den Antwort selbst, nur der Benutzer der Antwort verwendet wird. Dadurch soll geprüft werden, ob es möglich ist die Frageprofile anhand von ähnlichen Benutzerinteressen erstellen zu können.

4.3 BERECHNUNG DER BENUTZERPROFILE

Das Benutzerprofil ist das Gegenstück des Frageprofils und soll das Interesse des Benutzers widerspiegeln. Dies baut auf die Frageprofile auf, welche der Benutzer beantwortet hat zzgl. seiner ggfs. externen hinterlegten Daten außerhalb der Community Question Answering (CQA) Plattform. Die externen Daten werden dabei mit den gleichen Vorverarbeitungsschritten und dem trainierten Modell transformiert, welche auch für die Erstellung der Frageprofile verwendet wurden. Dadurch soll gewährleistet werden, dass das gleiche Vokabular verwendet wird und die gleiche Vektorgröße besitzt wie diese und somit wie eine beantwortete Frage behandelt wird und untereinander vergleichbar sind.

Um das aktuelle Interesse eines Benutzers zu berechnen, werden die ihm zuordenbare Vektoren verwendet und in einen einzelnen Vektor transformiert der gleichen Vektorlänge. Dies kann z.B. durch die Bestimmung des Mittelwertes der Vektoren erreicht werden. Da hierbei allerdings ältere und jüngere beantwortete Fragen gleichermaßen in die Berechnung des Mittelwertes fließen, bilden diese unter Umständen nicht das aktuelle Benutzerinteresse ab, falls sich dies über die Vergangenheit verändert hat. Um dies entgegenzuwirken kann eine Formel für den exponentiellen Zerfall verwendet, welche ältere Daten abschwächen soll, damit diese weniger Aussagekraft für die finalen Berechnung des Benutzerprofils besitzen [Li+14; DL05].

$$\begin{aligned} f(t) &= e^{-\lambda t} \\ \lambda &= \text{Zerfall } [0, \infty) \\ t &= \text{Zeitlicher Abstand} [0, \infty) \end{aligned} \tag{4.1}$$

Je höher das λ gewählt wird, desto weniger sollen veraltete Daten berücksichtigt werden. Dies wird in Abbildung 4.5 dargestellt. Bei Verwendung eines $\lambda = 0.015$ wird z.B. die 10. letzte Fragen noch mit 87% berücksichtigt, während dies bei einem $\lambda = 0.05$ auf 64% sinkt. In der späteren Evaluierung werden verschiedene λ-Werte genauer getestet und deren Auswirkung.

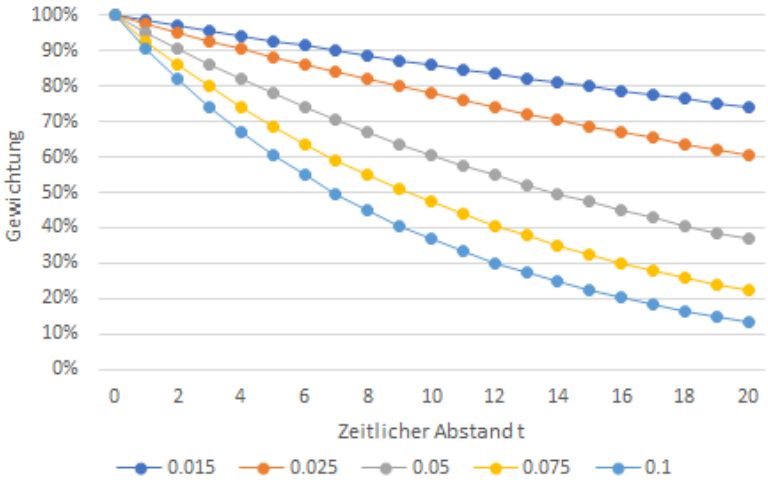


Abbildung 4.5: Verlauf des exponentiellen Zerfalls bei Verwendung verschiedener λ Werte

Nachdem die Gewichtungen ermittelt wurden, müssen diese abschließend dennoch durch eine Methode in das endgültige Benutzerprofil überführt werden. Im folgenden werden Methoden vorgestellt, welche die Gewichtungen nutzen, um das Benutzerprofil zu ermitteln unter der Berücksichtigung von zeitlichen Veränderung des Benutzerinteresse.

4.3.1 *Maximum Methode*

Die erste Variante soll das maximale Benutzerinteresse darstellen. Dies baut darauf aus, dass bei Verwendung von Tf-idf ein höherer Wert für ein Wort bedeutet, dass dies wichtig wahr innerhalb eines Textes gegenüber den restlichen Dokumenten. Hierzu werden über die relevanten Vektoren eines Benutzers pro Wort der maximale Wert verwendet und es entsteht das Benutzerprofil, welches die gleiche Vektorlänge besitzt wie ein Frageprofil. Um die Veränderungen des Interesse über die Zeit zu berücksichtigen, werden die Vektoren zuvor noch mit der Zerfallsgewichtung multipliziert, wie in 4.2 gegeben. Dies ermöglicht es, mehrere verschiedene Interessen bzw. Themengebiete des Benutzers zu berücksichtigen aber durch die Verwendung des

Strafterms und damit verbunden die Abnahme der Werte, dass dieser nicht Experte für alle Fragen wird.

$$
\begin{aligned}
f(\vec{v}) &= e^{-\lambda \vec{v}_t} \cdot \vec{v} \\
\vec{v} &= \text{Vektor des Frageprofils} \\
\vec{v}_t &= \text{Zeitlicher Abstand des Frageprofils für Zerfall}
\end{aligned} \tag{4.2}
$$

Original Werte

t	Tok1	Tok2	Tok3
4	0.600	0.100	0.200
3	0.500	0.200	0.450
2	0.550	0.550	0.450
1	0.550	0.550	0.400
0	0.150	0.650	0.300

(1)

x

f(t)
0.905
0.928
0.951
0.975
1.000

(2)

=

Gewichtete Werte

Tok1	Tok2	Tok3
0.543	0.090	0.181
0.464	0.186	0.417
0.523	0.523	0.428
0.536	0.536	0.390
0.150	0.650	0.300

(3)

max =>

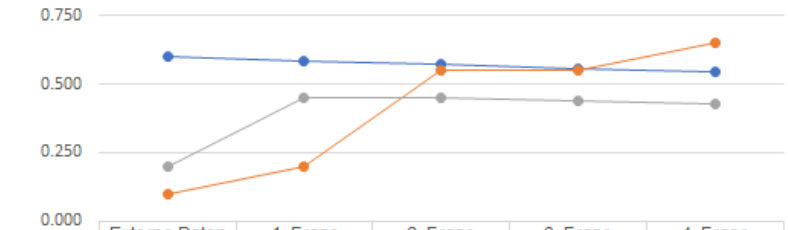
Benutzerprofil

Tok1	Tok2	Tok3
0.543	0.650	0.428

(4)

Abbildung 4.6: Beispiel für die Berechnung eines Benutzerprofils mit einem exponentiellen Zerfall von $\lambda = 0.025$ und der Maximum Methode

Dies wird in Abbildung 4.6 dargestellt. In diesem Beispiel gibt es fünf relevante Vektoren (vier beantwortete Fragen und die externen Daten) für den Benutzer. Die ursprünglichen Tf-idf-Werte *(1)* werden mit der Gewichtung *(2)* multipliziert, wodurch die gewichteten Tf-idf-Werte *(3)* entstehen. Aus diesen werden pro Wort der maximale Wert ermittelt, welches das Benutzerprofil *(4)* widerspiegelt. Zu sehen ist in diesem Beispiel für das erste Wort *(Tok1)*, der Tf-idf-Wert aus den externen Daten verwendet wird. Aufgrund des Zerfallsparameters wird dieser allerdings anstatt mit dem ursprünglichen Wert von 0.600 nur noch mit einem Wert 0.542 berücksichtigt. Sollte der Benutzer weitere Fragen beantworten, welches dieses Wort nicht bzw. selten verwenden, so wird dies immer weiter abgeschwächt.



	Externe Daten	1. Frage	2. Frage	3. Frage	4. Frage
Tok1	0.600	0.585	0.570	0.556	0.542
Tok2	0.100	0.200	0.550	0.550	0.650
Tok3	0.200	0.450	0.450	0.439	0.428

Abbildung 4.7: Beispiel für den Verlauf eines Benutzerprofils mit einem exponentiellen Zerfall von $\lambda = 0.025$ und der Maximum Methode

In Abbildung 4.7 wird gezeigt, wie sich das Benutzerprofil entwickelt anhand der Beispieldaten aus Abbildung 4.6. Es werden die Werte der Wörter jeweils nach dem aktuellen Schritt (Nur externe Daten, nach der 1. beantworteten Fragen, etc.) dargestellt. Ziel ist es, dass die externe Daten als Starthilfe

verwendet werden und sich über die Zeit ein immer genaueres Benutzerprofil erstellt.

Zu berücksichtigen ist allerdings, dass dieses Verfahren nur auf Vektoren sinnvoll ist, bei diesen ein höherer Wert auch die Relevanz des Wortes darstellt. Bei Verwendung eines Embedding wie z.B. Doc2Vec ist die Größe eines Wert innerhalb des Fragevektors nicht identisch mit seiner Wichtigkeit gegenüber einem Tf-idf-Vektor, wo ein hoher Wert auch ein wichtiges Wort für das Dokument darstellt.

4.3.2 *Gewichtetes arithmetisches Mittel*

Als zweite Variante wird das gewichtete arithmetisches Mittel [Wik19] vorgestellt, welches auch auf dem Doc2Vec-Vektoren angewandt werden kann. Da mit steigender Anzahl von beantworteten Fragen eines Benutzer die Gewichtung der Zerfallsformel immer stärker gegen Null tendiert, wie bereits in Abbildung 4.5 gezeigt wird, würde bei Anwendung des gewöhnlichen arithmetischen Mittel auf die gewichteten Werte diese auch immer stärker gegen Null tendieren.

$$
\begin{aligned}
f(\vec{v}) &= \frac{e^{-\lambda \vec{v}_t}}{\sum_{\vec{u}}^{V} e^{-\lambda \vec{u}_t}} \cdot \vec{v} \\
V &= \text{Relevante Vektoren des Benutzers} \\
\vec{v}_t &= \text{Zeitlicher Abstand } t \text{ des Vektors}
\end{aligned}
\tag{4.3}
$$

Stattdessen wird das gewichtete arithmetische Mittel der Gleichung 4.3 verwendet. Hierbei wird ermittelt, wie hoch der Anteil des aus dem Zerfalls berechneten Gewicht gegenüber der Summe der dort berechneten Gewichte ist. Dieses neue Gewicht wird dann mit dem jeweiligen korrespondierenden Vektor multipliziert und abschließend summiert als endgültiger Benutzervektor.

$$
g(V) = f(\vec{V}_1) + f(\vec{V}_2) + ... + f(\vec{V}_n) \tag{4.4}
$$

Original Werte

t	Tok1	Tok2	Tok3
4	0.600	0.100	0.200
3	0.500	0.200	0.450
2	0.550	0.550	0.450
1	0.550	0.550	0.400
0	0.150	0.650	0.300
Summe			

(1)

x

f(v)
0.162
0.179
0.198
0.219
0.242
1.000

(2)

=

Gewichtete Werte

Tok1	Tok2	Tok3
0.097	0.016	0.032
0.090	0.036	0.081
0.109	0.109	0.089
0.120	0.120	0.088
0.036	0.157	0.073
0.452	*0.439*	*0.362*

(3)

sum =>

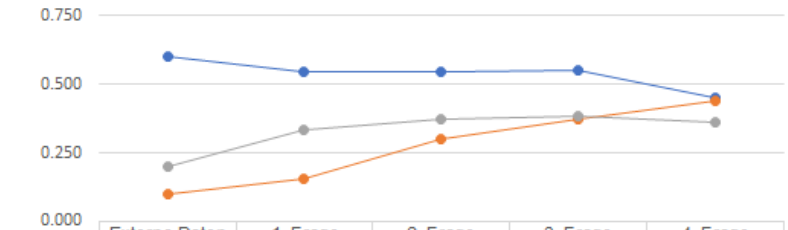
Benutzerprofil

Tok1	Tok2	Tok3
0.452	0.439	0.362

(4)

Abbildung 4.8: Beispiel für die Berechnung eines Benutzerprofils mit einem exponentiellen Zerfall von $\lambda = 0.10$ und dem gewichteten Mittelwert

In Abbildung 4.8 wird eine Beispielrechnung mit einem $\lambda = 0.10$ berechnet. Dort ist zu sehen, dass bei fünf vorhandenen Vektoren (vier beantwortete Fragen und die externen Daten) die zuletzt beantwortete Frage mit 24.2% ins Gewicht fallen würde, während die externen Daten nur noch mit 16.2% berücksichtigt werden. Der dazugehörige Verlauf des Benutzerprofils nach den einzelnen Fragen ist in Abbildung 4.9 dargestellt.



	Externe Daten	1. Frage	2. Frage	3. Frage	4. Frage
Tok1	0.600	0.547	0.548	0.549	0.451
Tok2	0.100	0.153	0.299	0.372	0.440
Tok3	0.200	0.332	0.375	0.382	0.362

Abbildung 4.9: Beispiel für den Verlauf eines Benutzerprofils mit einem exponentiellen Zerfall von $\lambda = 0.10$ und dem gewichteten Mittelwert

4.3.3 *Mittelwert der letzten n Fragen*

Während die Verwendung des gewichteten arithmetischen Mittelwert alle vergangenen beantwortete Fragen eines Benutzer beachtet, auch wenn diese immer mehr an Bedeutung bzw. Gewichtung verlieren, so wird nun noch eine Variante vorgestellt, welche einen harte Trennung von vergangenen Fragen vornimmt. Dabei wird ein Wert N vorgegeben und nur die N zuletzt beantworteten Fragen des Benutzers berücksichtigt. Alle vorherigen Fragen fließen somit nicht mehr in die Berechnung des Benutzerprofils, wodurch immer das aktuelle Interesse dargestellt werden soll. Durch die Wahl des N kann gesteuert werden, wie weit zurück geschaut werden soll.

	1. Frage	2. Frage	3. Frage	4. Frage	5. Frage
Nach 1. Frage	x				
Nach 2. Frage	x	x			
Nach 3. Frage	x	x	x		
Nach 4. Frage		x	x	x	
Nach 5. Frage			x	x	x

Tabelle 4.1: Einschränkung der Fragen für das Benutzerprofil unter Verwendung der jeweils letzten drei Fragen

Original Werte

t	Tok1	Tok2	Tok3
4	0.600	0.100	0.200
3	0.500	0.200	0.450
2	0.550	0.550	0.450
1	0.550	0.550	0.400
0	0.150	0.650	0.300

(1)

last 3 =>

Letzten 3 Werte

Tok1	Tok2	Tok3
0.550	0.550	0.450
0.550	0.550	0.400
0.150	0.650	0.300

(2)

mean =>

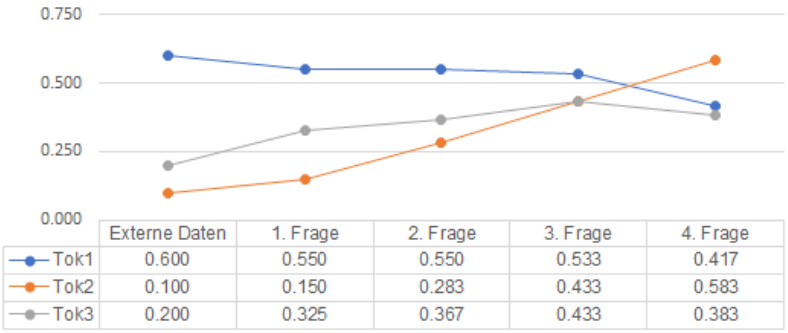
Benutzerprofil

Tok1	Tok2	Tok3
0.417	0.583	0.383

(3)

Abbildung 4.10: Beispiel für die Berechnung eines Benutzerprofils mit Berücksichtigung der letzten drei Fragen

In der Tabelle 4.1 wird der Verlauf dargestellt, welche Frage in Betrachtung gezogen werden für das Benutzerprofil nach der jeweiligen beantworteten Frage mit einem Fenster von $N = 3$. Bei den ersten beiden Fragen, werden nur diese betrachtet, da die Anzahl der Fragen kleiner als N ist und ab der dritten Fragen werden dann nur noch die letzten drei Fragen verwendet.



	Externe Daten	1. Frage	2. Frage	3. Frage	4. Frage
Tok1	0.600	0.550	0.550	0.533	0.417
Tok2	0.100	0.150	0.283	0.433	0.583
Tok3	0.200	0.325	0.367	0.433	0.383

Abbildung 4.11: Beispiel für Verlauf eines Benutzerprofils mit Berücksichtigung der letzten drei Fragen

In der Abbildung 4.10 wird die Berechnung anhand eines konkreten Beispiels vorgenommen, welches auch bei den beiden vorherigen Varianten genutzt wurde. Es werden nur die zeitlich drei letzten Frageprofile verwendet und über diese der Mittelwert gebildet zur Bestimmung des aktuellen Benutzerprofils. In der Abbildung 4.11 ist der dazugehörige Verlauf dargestellt.

Wenn ein Benutzer sich nicht mehr für ein altes Themengebiet interessiert, wo er in Vergangenheit viele Antworten erfasst hat und sich auf ein neues konzentrieren möchte, kann diese Methode zum Vorteil werden, da die alten Fragen ab einen gewissen Zeitpunkt komplett wegfallen. Wenn sich allerdings ein Benutzer für mehrere Themengebiete gleichzeitig interessiert, kann dies zu Komplikationen führen, wenn das Fenster zu klein gewählt wird, da dann evtl. gewisse Interessen bei der Berechnung des Benutzerprofils nicht mehr berücksichtigt werden.

4.4 NEURONALES NETZ ARCHITEKTUR

Für die Klassifizierung wurde sich für die Verwendung eines künstliches neuronales Netz (KNN) entschieden. In einer Vorabprüfung wurde festgestellt, dass ein einfaches Distanzmodell, welches anhand der Ähnlichkeit zwischen dem Fragen- und Benutzerprofil eine Trennung zwischen einer positiven und negativen Empfehlung macht ausreichend ist. Aufgrund dieser einfachen Trennung ist das KNN weniger Anfällig bzgl. eines Overfitting gegenüber der Verwendung eines tiefen neuronales Netz, welches zusätzliche versteckte Schichten zur Verdichtung besitzen. Bei diesen kann die Gefahr bestehen, dass wenn ein Themengebiet häufiger vertreten ist gegenüber anderen, was sich in der Anzahl von ähnlichen Fragenprofile bemerkbar macht, während der Trainingsphase ein Overfitting auf dieses stattfindet und unabhängig vom zugehörigen Benutzerprofil eine Empfehlung vorhergesagt wird. In einem umgekehrten Fall könnte dies auch passieren, wenn ein Benutzer bereits sehr viele Fragen beantwortet hat und das KNN lernt, diesen immer zu empfehlen unabhängig davon, ob die Frage ihn interessiert. Da im Zuge diese Arbeit allerdings bei der Erstellung der negativen Daten zu jedem Benutzerprofil ein Frageprofil ermittelt wird, welches der Benutzer nicht beantwortet hat, wären in diesem Fall die positive und negative Daten ausgeglichen.

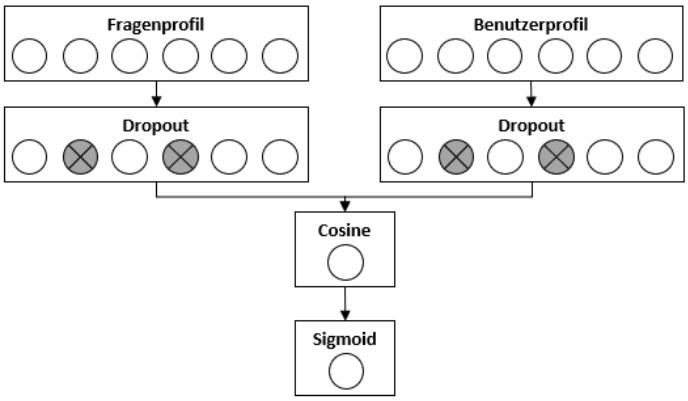


Abbildung 4.12: Verwendetes Distanz Modell als neuronales Netz

Die Verwendung von anderen Klassifizierungsverfahren, wie z.B. eine Support Vector Machine (SVM) [Jam+13b] oder eine Logistische Regression [Jam+13a], welche als Eingabe die Ähnlichkeit oder Distanz zwischen dem Frage- und Benutzerprofil erhalten, erzielen ähnliche gute Ergebnisse und können als Alternative verwendet werden. Ein Vorteil von neuronalen Netzen ist allerdings, dass dies weitere Möglichkeiten bietet, um zum einen die Qualität noch etwas zu optimieren, sowie die spätere Verwendung zu vereinfachen. So bietet dies die Möglichkeit, mehrere Vektoren als Eingabe zu erhalten und diese im neuronalen Netz unterschiedlich zu behandeln, bevor diese zusammengeführt werden und die Klassifizierung zu trainieren und vorherzusagen. Dies ist vor allem im Hinblick auf eine spätere Erweiterbarkeit

hilfreich, wenn weitere Features, wie z.B. Motivation oder Antwortqualität des Benutzer berücksichtigt werden sollen.

In dieser Arbeit werden das Frage- und das Benutzerprofil als separate Eingabevektoren verwendet, welche jeweils eine eigene Dropout-Schicht erhalten, bevor die Kosinus-Ähnlichkeit berechnet wird. Diese einfache Netzwerk Architektur wird in Abbildung 4.12 dargestellt. Abschließend wird noch einzelne Ausgabe verwendet, welche vorhersagen soll, ob das Frage- und Benutzerprofil zusammenpassen und eine positive Empfehlung besteht oder keine Übereinstimmung besteht. Dieses neuronales Netz hat zwei Parameter (einen Bias und eine Gewichtung bei der Ausgabeschicht), welche während der Trainingsphase optimiert werden können, um die Fehler zu reduzieren. Es wird somit ein neuronales Netz verwendet, welches lernt, ab welcher Ähnlichkeit zwischen dem Frage- und dem Benutzerprofil eine Übereinstimmung stattfindet.

Im folgenden wird eine kurze Erläuterung der verwendeten Komponenten in dem neuronalen Netz aufgelistet:

- **Fragenprofil und Benutzerprofil:** Dies sind die beiden Vektoren der Frage und des Benutzer, welche als Eingabewerte für das KNN dienen.
- **Dropout:** Während der Trainingsphase werden ein vorgegebener Anteil an Werten der vorherigen Schicht deaktiviert, wodurch diese keine Ausgabe mehr erzeugen. Dadurch soll ein Overfitting während der Trainingsphase vermieden werden, da berücksichtigt wird, dass leicht abgewandelte Vektoren ein gleiches Ergebnis erzielen sollen [Sri+14].
- **Cosine:** Dies nimmt zwei Vektoren der gleichen Größe entgegen und berechnet die Kosinus-Ähnlichkeit, wodurch eine Reduzierung auf nur einen Ausgabewert entsteht.
- **Sigmoid:** Da es sich bei der Problemstellung um ein binäres Klassifierungsproblem handelt, wird als Ausgabeschicht ein einzelnes Neuron verwendet mit Sigmoid als Aktivierungsfunktion. Dadurch wird eine Wahrscheinlichkeit trainiert, dass eine Übereinstimmung zwischen den Frage- und Benutzerprofil besteht und eine Empfehlung ausgesprochen werden kann. Die Verwendung der Sigmoid-Funktion $f(x) = \frac{1}{1+e^{-x}}$ gewährleistet, dass als Ausgabewert ein Wert zwischen 0 und 1 entsteht, welcher als Wahrscheinlichkeit für eine positive Empfehlung verwendet wird.

4.5 TRAININGS-, EVALUIERUNG- UND TESTDATEN

Die Erstellung und Aufteilung von Trainings-, Evaluierung- und Testdaten ist eine besondere Herausforderung für die Verwendung eines ML Klassifizierer im Zusammenhang mit den vorhandenen Daten. Es kann nicht wie üblich eine zufällige Stichprobe aus den Daten gezogen werden für das Trainieren und Testen der Daten, sondern es sollte der zeitliche Verlauf der Fragen und Antworten berücksichtigt werden, um nicht evtl. ein Data Leakage zu erhalten. Ein Data Leakage wäre in diesem Fall z.B., dass für das Training Daten verwenden werden, welche es zeitlich gesehen noch nicht wissen darf. Die Daten werden daher nach ihrem Beantwortungsdatum sortiert und die Aufteilung anhand eines Zeitpunkt vorgenommen. Alle Datensätze nach diesem Zeitpunkt (die jüngsten) werden als Testdaten und alle vorherigen Datensätze als Trainingsdaten vorgesehen. Dies wird wie in Abbildung 4.13 innerhalb der Trainingsdaten wiederholt, um zusätzlich noch einen unbekannten Evaluierungsdatensatz während des Trainings zur Verfügung zu haben. Dieses Verfahren spiegelt die Realität am besten wieder, da der Klassifizierer später für neue unbeantwortete Fragen verwendet werden soll.

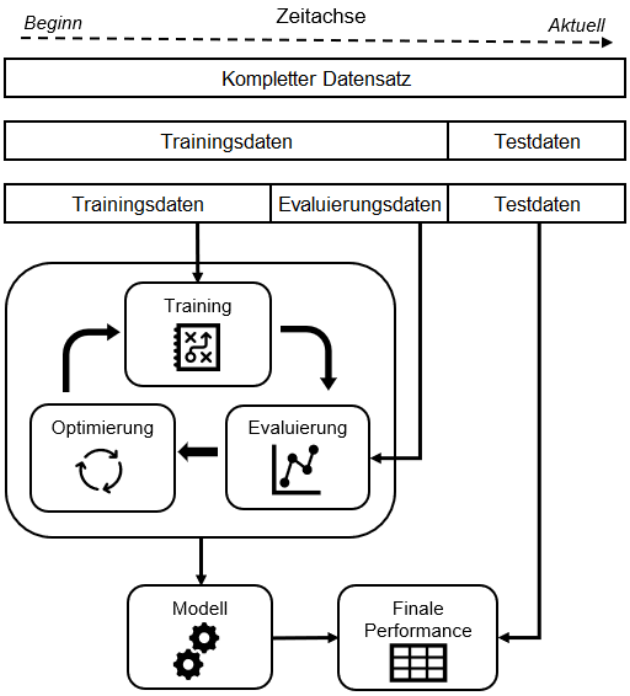


Abbildung 4.13: Aufteilung und Verwendung der Trainings-, Evaluierungs- und Testdaten

Diese drei Datensätze haben folgende Verwendung:

- **Trainingsdaten:** Die Trainingsdaten werden verwendet, um das neuronale Netz zu lernen. Diese werden als Eingabe in das neuronale Netz verwendet und die vorhergesagten Werte mit den erwartenden Ausga-

bewerte verglichen. Der daraus folgende Fehler wird verwendet, um das neuronale Netz zu optimieren.

- **Evaluierungsdaten:** Die Evaluierungsdaten werden nicht für das Training verwendet, sondern dienen zur Messung, welche Ergebnisse (Fehler bzw. Genauigkeit) das neuronale Netz auf unbekannte Datensätze erzielt nach einem Trainingsdurchlauf. Wenn keine Verbesserung der Ergebnisse auf den Evaluierungsdaten mehr stattfindet, kann das weitere Training unterbrochen werden und das finale Modell anhand dessen beste Ergebnis ausgewählt werden.

- **Testdaten:** Da die Evaluierungsdaten zwar nicht für die eigentliche Trainingsphasen verwendet werden aber dennoch an der Auswahl des finalen Modells beteiligt sind, sollten abschließend noch der dritte Datensatz mit Testdaten verwendet werden, um zu überprüfen, welche Ergebnisse das neuronale Netz mit diesen erzielt. Die Genauigkeit sollte dabei in dem Bereich sein, wie bei den Evaluierungsdaten.

Zu beachten ist allerdings, dass die Evaluierungs- bzw. Testdaten dennoch die Daten aus den vorherigen Stufen verwendet für die Erstellung des Benutzerprofils, da diese auf alle vergangen Daten aufbauen. Dies ist notwendig, da sich diese für die realistische Darstellung anhand der kompletten Historie der Benutzerinteraktionen berechnen sollte. Daher ist es möglich, dass Fragen, welche mehrfach beantwortet wurden auch in jedem Datensatz vorkommen. Lediglich das dazugehörige Benutzerprofil ist immer einzigartig zu einem Zeitpunkt.

Eine weitere Herausforderung für das Training des neuronalen Netz ist es, dass die vorhandenen Daten nur positive Beispiele bereitstellt. Dies bedeutet, dass nur bekannt ist, welche Fragen ein Benutzer beantwortet hat aber nicht, welche Fragen er nicht beantworten würde. Dies ist allerdings für ein Klassifizierungsproblem notwendig, damit der Klassifizierer lernen kann, wann die Übereinstimmung zwischen einer Frage und Benutzer positiv oder negativ sein sollte und eine Empfehlung ausgesprochen werden kann. Daher wird in den folgenden beiden Abschnitten sowohl auf die Erzeugung von positiven und negativen Daten eingegangen.

4.5.1 *Positive Daten*

Die positive Daten stehen für die Übereinstimmung zwischen einer Frage und einem potentiellen Beantworter. Das neuronale Netz muss allerdings nicht nur mit den Frage- und Benutzerprofilen trainiert werden, sondern kann weitere Methoden verwenden, um die Anzahl der Trainingsdaten zu vergrößern. Dadurch kann ein zu schnelles Overfitting der Trainingsdaten reduzieren werden, das neuronale Netz verallgemeinern und für evtl. weitere Anwendungsbereiche genutzt werden.

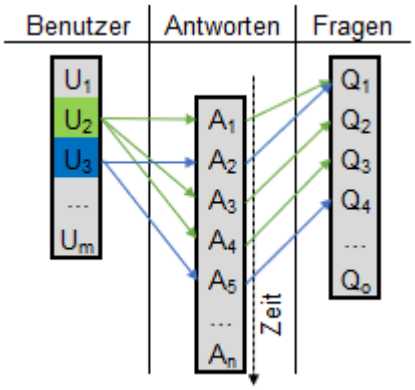


Abbildung 4.14: Zuordnung zwischen Benutzer und beantworten Fragen

	—— Zeit —–>					
Frage-ID:	Q_1	Q_1	Q_2	Q_3	Q_4	...
Antwort-ID:	A_1	A_2	A_3	A_4	A_5	...
Beantworter:	U_2	U_3	U_2	U_2	U_3	...

Tabelle 4.2: Darstellung von 4.14 in Tabellenform transformiert mit zeitlicher Sortierung anhand der Antworten

Das Erstellen der Eingabedaten für den Klassifizierer unterteilt sich dabei in die Schritte Filterung, Gruppierung & Aggregation, sowie evtl. Vertauschen der beiden Eingabevektoren. Es wird dabei über jedes Frage und Beantworter Paar Iteriert und ausgehend von dieser die passenden Vektoren ermittelt. Der erste Eingabevektor ist somit der Fragevektor und der zweite Eingabevektor der ermittelte, welche im weiteren als Referenz-Vektor oder ja nach Anwendungsfall Benutzer-Vektor genannt wird.

Filterung:
Ausgehend von der Frage werden passende Daten gefiltert die zeitlich zu dieser Frage in Relation stehen. Diese Relation entsteht über die Benutzer, welche die Frage beantwortet haben und in Abbildung 4.14 dargestellt wird bzw. in Tabelle 4.2 als Tabellenform. Dabei gibt es die folgenden vier Filtermöglichkeiten:

- *Gleich:* Es werden nur die Daten der aktuelle Frage berücksichtigt. Dies wäre die Frage selbst, sowie deren Antworten bei Bedarf.
- *Vorher:* Es werden alle Fragen berücksichtigt, welche die Benutzer beantwortet haben, welcher auch die aktuelle Frage beantwortet hat. Anhand der Tabelle 4.2 hat als Beispiel der Benutzer U_2 zum Zeitpunkt als er die Frage Q_3 beantwortet hat vorher bereits die Fragen Q_1 und Q_2 beantwortet. Diese können dann im nächsten Schritt verwendet werden, um z.B. das Benutzerprofil zum aktuellen Zeitpunkt der Frage Q_3 zu ermitteln.

Filtermethode	Frage-ID	Referenz-ID	Frage-Vektor	Referenz-Vektor
Vorher	Q_2	U_2	Q_2	Q_1
Vorher	Q_3	U_2	Q_3	Q_1
Vorher	Q_3	U_2	Q_3	Q_2
Vorher	Q_4	U_3	Q_4	Q_1
Gleich	Q_1	Q_1	Q_1	Q_1
Gleich	Q_2	Q_2	Q_2	Q_2
Gleich	...	...	...	...
Später	Q_1	U_2	Q_1	Q_2
Später	Q_1	U_2	Q_1	Q_3
Später	Q_1	U_3	Q_1	Q_4
Später	Q_2	U_2	Q_2	Q_3

Tabelle 4.3: Beispiel für verschiedene Filtermethoden anhand Abbildung 4.14

- *Später:* Dies funktioniert wie *vorher*, nur das Fragen berücksichtigt werden, welcher der Benutzer noch beantworten wird in der Zukunft. Da Benutzer eine Frage erst beantworten können, wenn diese überhaupt gefragt wurde, kann hiermit simuliert werden, wie es wäre, wenn die aktuelle Frage zu einem späteren Zeitpunkt gestellt wurde. Es wird hierbei die Annahme getroffen, dass der Benutzer diese Frage auch beantworten würde, wenn diese zu einem anderen Zeitpunkt gestellt wird.
- *Kombinationen:* Weiterhin sind Kombinationen dieser Methoden möglich. So kann es Sinnvoll sein die Vereinigungsmenge aus *Vorher* und *Später* zu verwenden, um eine Menge mit evtl. ähnlichen Fragen zu erhalten, welche die Benutzer der aktuelle Frage interessiert haben.

Gruppierung:

Frage-ID	Referenz-ID	Frage-Vektor	Referenz-Vektoren
Q_2	U_2	Q_2	Q_1
Q_3	U_2	Q_3	Q_1 + Q_2
Q_4	U_3	Q_4	Q_1
...	...	...	...

Tabelle 4.4: Beispiel für Gruppierung von positiven Daten, ausgehend von Tabelle 4.2

Die Gruppierung kann auf die gefilterten Daten angewandt werden. Diese findet pro Filtermethode (*Vorher*, *Später* oder Kombinationen) statt. Innerhalb der Filtermethoden kann eine Gruppierung anhand der Frage-ID, Referenz-ID oder anhand der Frage-ID und Referenz-ID stattfinden. In Tabelle 4.4

wird das oben gezeigte Beispiel gruppiert anhand der vorher-Methode. Dies gruppiert die Daten so, dass pro Frage und Beantworter die zum Zeitpunkt der Antwort vorher beantworten Fragen des Benutzers auflistet, woraus sich später das vergangene Benutzerprofil ableiten lässt.

Sortierung:
Standardmäßig werden die Daten zeitlich nach der Benutzerinteraktion sortiert. Dies ist z.B. notwendig, um für die Benutzerprofile ältere beantworte Fragen weniger zu berücksichtigen. Bei Berücksichtigung einer solchen anschließenden zeitlichen Bestrafung, kann es aber auch sinnvoll sein, dass die Daten in umgekehrter Reihenfolge oder zufällig sortiert werden. Dies baut auf die bereits beschriebene Annahme auf, dass Benutzer ihre Fragen beantwortet hätten, wenn diese auch in einer anderen Reihenfolge gestellt wurde und kann zu einer Verallgemeinerung des Modells bewirken.

Anpassungen:
Auf den Referenz-Vektor können noch Anpassungen vorgenommen werden. Dies kann zum einen die Bestrafung von älteren Fragen sein, wie es für die Berechnung des Benutzerprofils verwendet wird. Dies ermöglicht es in Kombination mit der Filterung *Vorher* das Benutzerprofil zu erstellen, welches der Benutzer zum jeweiligen Zeitpunkt der Beantwortung seiner Fragen hatte. Ebenfalls wird die Möglichkeit geboten, die Werte des Vektors zufällig anhand einer Verteilungsannahme zu adjustieren. So kann es z.B. sinnvoll sein, dass die Werte mit einer zufällig Gleichverteilung auf dem Intervall $[0.9, 1.0]$ und der gleichen Größe des Referenzvektors multipliziert wird. Dies soll eine Generalisierung in den Trainingsdaten bewirken, wodurch Benutzer mit sehr ähnliche (angepasste) Fragen ebenfalls die Frage beantwortet hätten.

Aggregation:
Auf die gruppierten Daten können dann Aggregationen vorgenommen werden, wobei sich Lageparameter aus der deskriptiven Statistik empfehlen. Dies können die Verwendung von verschiedenen Mittelwerte (Arithmetisches, Geometrisches Mittel, etc.) oder Quantile (Maximum, Median, etc.) beinhalten, sowie der vorgestellte gewichteter Mittelwert. Aufgrund der Gruppierung wird dies auf die Referenz-Vektoren angewandt, da diese sich unterscheiden, während der Frage-Vektor als Ausgangspunkt identisch bleibt.

Vertauschen:
Abschließend wird noch die Möglichkeit geboten, dass ein Frage-Vektor mit einem Referenz-Vektor vertauscht wird. Hierbei sollte allerdings beachtet werden, dass keine redundante Daten entstehen, da dies sonst diese Paarungen bevorzugen wird beim Training. Dies wäre z.B. der Fall, wenn z.B. ein Vertauschen nach der Filtermethode *Gleich* verwendet wird, da dort der Referenz-Vektor bereits identisch ist mit den Frage-Vektor. Ziel dieses Vorgehen soll sein, dass nicht nur die vorhanden Frage als Frage-Vektor verwendet werden, sondern auch ähnliche, welche in Zukunft auftreten könnten. Je

nach Anwendungsfall, kann dies auch genutzt werden, um das Modell so zu verallgemeinern, dass dies nicht nur für das Übereinstimmen zwischen Fragen- und Benutzerprofile funktioniert. Dadurch wird es möglich, dass auch zwei Frageprofile bzw. zwei Benutzerprofile miteinander verglichen werden können, ob diese eine thematische Übereinstimmung besitzen.

4.5.2 *Negative Daten*

Die negativen Daten sind relevant für den Klassifizierer, um diesen darauf hin zu trainieren, ob eine Frage für einen Benutzer interessant sein könnte (positiv) oder uninteressant (negativ) sein wird. Da diese in den zugrundeliegenden Daten nicht gegeben sind, müssen diese künstlich erstellt werden. Dabei sollte darauf geachtet werden, dass die Menge der positiven und negativen Daten ausgeglichen sind, um diese die gleiche Gewichtung für die Vorhersagen zu geben. Dies soll verhindern, dass ansonsten die Hauptklasse, welche stärker vertreten ist in den Datensätze primär vorhergesagt wird, da bei dem Klassifizierungsproblem die Genauigkeit als relevante Bewertungsmetrik verwendet wird [LD13]. Daher ist es auch Sinnvoll, dies ebenfalls auf den Testdaten durchzuführen, um die Konfusionsmatrix komplett darstellen zu können und alle Metriken daraus berechnen zu können.

Das Erstellen der negativen Daten baut auf den positiven Daten auf. Pro vorhandener positiver Datensatz sollte ein passendes Negativbeispiel ermittelt werden aus den restlichen Daten. Da die Daten aus der Verknüpfung eines Frage-Vektor und einem Referenz-Vektor bestehen, gibt es folgende zwei Möglichkeiten zum ermitteln eines zugehörigen negativen Beispiels:

1. Es wird ausgehend von der Frage ein negatives Beispiel aus den Referenz-Vektoren gesucht oder
2. Es wird ausgehend von dem Referenz-Vektor eine Frage als negatives Beispiel ermittelt.

Ebenfalls ist die Verwendung von beiden Varianten möglich, wodurch die doppelte Menge an negativen Beispielen entstehen würde. Je nach Anwendungsfall, können die unterschiedlichen Varianten Vorteile bringen. Aus zeitlichen Gründen wird in dieser Arbeit aber keine Evaluierung der verschiedenen Varianten durchgeführt und sich für die zweite Variante entschieden, da die Priorität auf die Sicht des Benutzers gelegt wird. Dadurch entsteht in den fertigen Daten für jedes Benutzerprofil eine dazugehörige Frage, die er tatsächlich beantwortet hat und eine korrespondierende Frage, welcher der Benutzer nicht beantwortet hat als negatives Beispiel.

Das weitere Vorgehen ist bei beiden Varianten identisch und lediglich der Ausgangspunkt ist ein anderer. Die positiven Daten sind zeitlich sortiert anhand des Zeitpunkt der Antwort. Es soll in der Nachbarschaft eines Datensatz ein passendes negatives Beispiel gefunden werden. Dies baut auf die Idee auf, dass ein Benutzer seine Frage beantwortet hat aber z.B. die

vorherige Frage nicht beantwortet hat, weil diese ihn wahrscheinlich nicht interessiert hat. Dabei müssen die benachbarten Datensätze jeweils vorab bereinigt werden, indem diese keine Fragen sein dürfen, welche der Benutzer beantwortet hat, um eine Kollision zwischen gleichen positiven und negativen Daten zu vermeiden. Dies wäre der Fall, wenn die gleiche Frage zuvor von einem anderen Benutzer ebenfalls beantwortet wurde.

Bei der Betrachtung, werden sowohl vorherige als auch nachfolgende benachbarte Datensätze berücksichtigt, um zu gewährleisten, dass auf Datensätze, welche ansonsten keine vorherigen potentiellen Kandidaten haben auch eine negative Zuordnung erhalten. Weiterhin wird ein Fenster gewählt, wie viele benachbarte Datensätze in Frage kommen. Durch die Auswahl eines größeren Fenster, besteht zum einen die Möglichkeit, bei Bedarf mehrere negative Datensätze zu erzeugen, sowie spezifischer ein negatives Beispiel für den Benutzer auszuwählen.

Die einfachste Methode ist die zufällige Auswahl aus den potentiellen benachbarten Datensätze als negatives Beispiel. Eine weitere Möglichkeit ist die Berechnung der Distanz zwischen der aktuellen positiven Frage und den potentiellen negativen Fragen, wobei die Frage mit der höchstens Distanz als negatives Beispiel ausgewählt wird. Dies baut darauf auf, dass eine Frage evtl. nur nicht beantwortet wurde von dem Benutzer, weil diese bereits schon von einen anderen Benutzer beantwortet wurde. Durch eine höhere Distanz soll die Wahrscheinlichkeit erhöht werden, dass dies tatsächlich eine Frage ist, die den Benutzer nicht interessiert hat.

$$\begin{aligned}
\text{Euklidische Distanz} &: \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \\
\text{Manhattan-Distanz} &: \sum_{i=1}^{n} \mid x_i - y_i \mid \\
\text{Kosinus-Distanz} &: 1 - \frac{\vec{a} \cdot \vec{b}}{\mid \vec{a} \mid \cdot \mid \vec{b} \mid} \\
\text{Negatives Skalarprodukt} &: -\vec{a} \cdot \vec{b}
\end{aligned} \tag{4.5}$$

In 4.5 werden verschiedene Distanzmetriken dargestellt zwischen zwei Vektoren, welche verwendet werden können. Die Kosinus-Ähnlichkeit, sowie das Skalarprodukt wurden dabei etwas modifiziert, um entsprechend den anderen Distanz-Metriken zu gewährleisten, dass ein höhere Wert auch eine höhere Distanz bedeutet. Diese erhalten anhand ihrer Distanz einen Rang und die Fragen mit den höchsten Rang werden bevorzugt gewählt. Die Auswahl einer Distanzmetrik kann zu unterschiedlichen Ergebnisse kommen aber es wird die Verwendung des Skalarprodukt empfohlen, da diese am wenigsten Rechenoperationen benötigt zur Berechnung der Distanz und somit die Laufzeit der Trainingsphase reduziert, was sich vor allem auf der Verwendung von großen Datenmengen bemerkbar macht. Dies wird daher

auch im weiteren in dieser Arbeit verwendet unter der Berücksichtigung der direkten benachbarten Datensätze, was einer Fenstergröße von 1 entspricht.

Positive Frage-ID	Referenz-ID	Frage-Vektor Tok1	Tok2	Tok3	Tok4	Negative Frage-ID
Q_1	U_1	0.2	0.6	0.6	0.0	Q_2
Q_1	U_2	0.2	0.6	0.6	0.0	Q_3
Q_2	U_2	0.1	0.5	0.8	0.2	Q_3
Q_3	U_3	0.6	0.2	0.1	0.4	Q_4
Q_3	U_4	0.6	0.2	0.1	0.4	Q_4
Q_4	U_5	0.0	0.3	0.3	0.3	Q_3

Tabelle 4.5: Beispiel für die Distanzberechnung und Ermittlung eines negativen Beispiels

Ausgehend von einem Beispiel entsprechend Tabelle 4.5 werden die direkten Nachbarn als potentielle negative Datensätze betrachtet, sowie das Skalarprodukt zur Distanzberechnung verwendet. Auf der rechten Seite sind die entsprechenden negativen Datensätze abgebildet, welche ermittelt wurden. Bei dem ersten Datensatz Q_1U_1 wurde Q_2 als negatives Beispiel verwendet, da dies keine vorherigen hat und der zweite Datensatz die gleiche Frage ist, weshalb diese übersprungen wurden. Bei dem zweiten Datensatz Q_1U_2 ist die vorherige Frage identisch und die nachfolgende Frage von dem gleichen Benutzer, weshalb diese beiden übersprungen werden müssen und nur die Frage Q_3 verwendet werden kann. Der Datensatz Q_3U_3 hingegen besitzt zwei benachbarte Fragen, welche als negatives Beispiel verwendet werden können. Bei diesem wurde das negative Skalarprodukt als Distanzfuntion $d(\vec{a}, \vec{b})$ zwischen der aktuellen Frage und den benachbarten Fragen verwendet.

$$\begin{aligned} d(Q_3U_3, Q_2U_2) &= -(0.6 \cdot 0.1 + 0.2 \cdot 0.5 + 0.1 \cdot 0.1 + 0.4 \cdot 0.2) \\ &= -0.32 \Rightarrow Rang(1) \\ d(Q_3U_3, Q_4U_5) &= -(0.6 \cdot 0.0 + 0.2 \cdot 0.3 + 0.1 \cdot 0.3 + 0.4 \cdot 0.3) \\ &= -0.21 \Rightarrow Rang(2) \end{aligned} \tag{4.6}$$

Die Ergebnisse sind in 4.6 abgebildet. Die Distanz zu der Frage des nachfolgenden Datensatz ist höher gegenüber der vorherigen, weswegen diese auch den höchsten Rang bekommt und als negatives Beispiel verwendet wird. Mit diesem Vorgehen konnte somit für die bekannten positiven Datensätze jeweils für das hinterlegte Benutzerprofil ein passendes negatives Beispiel ermittelt werden und somit ein ausgeglichenes Verhältnis zwischen den positiven und negativen Klassen zum weiteren Trainieren der Klassifizierers.

5 EVALUIERUNG

In diesem Kapitel findet die Evaluierung des Recommender System (RS) statt. In einem ersten Schritt werden die verschiedenen Datensätze mit ihren Besonderheiten beschrieben, welche für die Evaluierung genutzt wurden. Anschließend werden geprüft, welche Parameter und Vorverarbeitungsschritte sinnvoll sind, um das Frageprofil zu erstellen. Dies wird sowohl mit der Durchführung der term frequency - inverse document frequency (Tf-idf)-Methode, sowie unterschiedlichen Doc2Vec Varianten ausgeführt. Aufbauend darauf werden die Erstellung der Benutzerprofile untersucht. Hierbei soll überprüft werden, ob die Methoden, welche Veränderungen des Benutzerinteresse im Laufe der Zeit berücksichtigen, bessere Ergebnisse liefern gegenüber der normalen Mittelwertberechnung und wie sich die Zuschaltung von externen Benutzerdaten auf die Empfehlungen auswirken.

5.1 VERWENDETE DATEN

Zur Evaluierung des Systems wurden für die Community Question Answering (CQA) Plattform öffentlich zugängliche Daten der Webseite Stackoverflow verwendet. Hierfür wurde die zur Verfügung gestellte Representational State Transfer (REST) Application Programming Interface (API)[1] verwendet. Diese wurde sich zur Nutze gemacht, um für einen vorgegebenen Zeitraum (Startdatum und Enddatum) alle gestellten Fragen inklusive ihrer Antworten, sowie Metadaten (verwendete Hastags, Zeitstempel der Erstellung, Benutzer-ID) zu extrahieren. Diese wurden in das im Abschnitt 2.2 beschriebene Schema 2.1 überführt, um eine eigene CQA-Plattform zu simulieren.

Für das Simulieren von externen Daten eines Benutzer außerhalb der CQA-Plattform wurde die auf Stack Overflow im Profil des Benutzers hinterlegte externe Webseite verwendet. Es wurde auch hierbei die REST API verwendet, um zu überprüfen, welche Benutzer, die eine Antwort in den zuvor extrahierten Daten auch eine Webseite angegeben haben. Mit dem Python Paket Scrapy[2], welches ein Webcrawling-Framework bereitstellt zum durchforsten und extrahieren von Webseiten wurde die angegeben Textinhalte der Webseite extrahiert. In einem ersten Versuch wurde nicht nur die eigentliche angegebene Webseite verwendet zum Anreichern der Benutzerprofile durch externe Daten, sondern auch benachbarte Webseite, welche eine ausgehende Verlinkung auf der Webseite besitzen. Nach einer stichprobenartige Überprüfung der Daten, konnte allerdings festgestellt werden, dass diese zusätzliche Webseiten oftmals keinen Bezug mehr zu dem eigentlichen Benutzer haben.

1 https://api.stackexchange.com/
2 https://scrapy.org

Dies war z.B. der Fall, wenn ein Benutzer seinen Lebenslauf auf einer öffentlichen Plattform als Webseite angegeben haben und die dort verlinkten Webseiten dann Inhalte des Webseitenbetreibers oder anderen Benutzern hatte. Daher wurde sich nur auf die direkt angegebene erste Webseite beschränkt, um das Benutzerinteresse anzureichern. Weiterhin wurden einige bekannte Webseiten ausgeschlossen, welche einen Login benötigen (z.B. facebook.com oder linkedin.com) oder wieder auf das eigene Stackoverflow-Profil verweisen, sowie offensichtlich falsch angegebene Webseiten (z.B. google.com).

Datensatz	**V1**	**V2**	**V3**
Anzahl Fragen	27729	5000	26172
Fragen von	01.01.2018	01.01.2018	01.01.2015
Fragen bis	06.01.2018	02.01.2018	31.12.2018
Mit Antworten	22982	4188	26172
Ohne Antworten	4747	812	0
Anzahl Antworten	34455	6416	49245
Antworten von	01.01.2018	01.01.2018	01.01.2015
Antworten bis	31.12.2018	31.10.2018	31.12.2018
Benutzer mit Antworten	16870	4114	15900
Benutzer mit Webseiten	1406	475	530
Anz. Antworten Top Benutzer	95	27	1248
Verwendete Hashtags	83650	15083	75862
Eindeutige Hashtags	9796	3576	7045

Tabelle 5.1: Zusammenfassung der verwendeten Daten

Dies ist der erste Datensatz, welcher 27729 Fragen und 34455 Antworten von 16870 verschiedenen Benutzern hat. Um zu überprüfen, ob die Methoden auch auf weniger Daten angewandt werden können, wurde aus diesen Datensatz ein kleinerer extrahiert, welcher nur die ersten 5000 Fragen beinhaltet, welche 6416 Antworten von 4114 verschiedenen Benutzer besitzen. Da diese zwei Datensätze sich nur auf einen kurzen Zeitraum konzentrieren, sollte noch ein weiteren Datensatz verwendet werden, welcher sich über einen längeren Zeitraum streckt, indem es auch Benutzer gibt, welche viele Antworten über die Zeit verteilt erstellt haben. Dadurch soll es möglich sein, ob sich die Interessen der Benutzer über diese Zeit verändert haben und ob dies berücksichtigt werden kann bei der Berechnung des Benutzerprofils. Um diesen Datensatz etwas einzuschränken von der Größe, wurde ausgehend von dem bestehenden Datensatz überprüft, welche Benutzer bereits dort mehrere Antworten gegeben haben und externe Daten vorhanden sind. Für diese Benutzer wurde ebenfalls über die REST API ermittelt, welche Fragen diese ab 01.01.2015 beantwortet haben und diese dann inklusive weitere Antworten extrahiert. Dieser Datensatz enthält 26172 Fragen, sowie 26172 Antworten von 15900 verschiedenen Benutzer. Eine komplette Auflis-

tung der Informationen zu den drei verschiedenen Datensätzen findet in der Tabelle 5.1 statt.

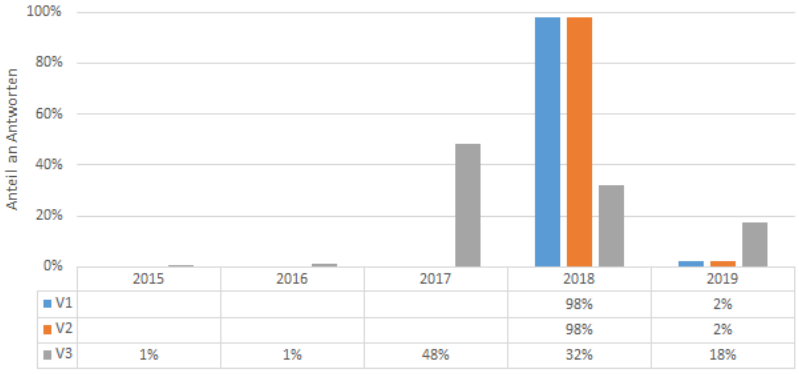


Abbildung 5.1: Verteilung der Antworten pro Jahr und Datensatz

In Abbildung 5.1 wird die Verteilung der Antworten für die einzelnen Datensätze auf die Jahre gezeigt. Die ersten beiden Datensätze konzentrieren sich dabei auf einen kurzen Zeitraum im Jahr 2018, während der dritte Datensatz genutzt werden kann für eine Verteilung über mehrere Jahre. In Abbildung 5.2 werden innerhalb der Datensätze der Anteil an Benutzern gezeigt und ihrer Anzahl an Antworten. Für eine einfachere Darstellung werden diese in Gruppe eingeteilt (2 Fragen, 3 bis 5 Fragen, etc.). Die Benutzer mit nur einer Antworten werden hierbei nicht berücksichtigt. Diese sind für die Trainingsphase und Evaluierung nicht relevant, da das Benutzerprofil mit der ersten beantworteten Frage beginnt und somit erst ab der zweiten Fragen in der Evaluierung berücksichtigt werden kann. Zu sehen ist in dieser Abbildung, dass in dem zweiten Datensatz (V2) der Großteil der Benutzer (89.34%) nicht mehr als 5 Antworten gegeben haben und es nur einen sehr geringen Anteil von 0.10% gibt, welcher mehr als 25 Antworten geschrieben haben, was nur einem Benutzer entspricht. Der erste Datensatz (V1) sieht diesem sehr ähnlich aus. Auch bei diesem haben lediglich 1.05% der Benutzer mehr als 25 Antworten gegeben. Dies ändert sich in dem dritten Datensatz (V3), welcher mehrere aktive Benutzer besitzt. In diesem haben 6.15% der Benutzer mehr als 25 Antworten geschrieben und 2.61% der Benutzer sind sehr stark aktiv mit mehr als 100 Antworten, was absolut 71 Benutzern entspricht.

Die drei Datensätze sollen somit folgende Szenarien abbilden, wobei für diese ein Alias für die spätere Identifizierung vergeben wurde.

- Standard Datensatz **(V1)**: Dies ist der große Datensatz, welcher sich auf einen kurzen Zeitraum beschränkt und eine aktive Community repräsentieren soll.
- Kleiner Datensatz **(V2)**: Dies ist die kleinere Variante des oberen Datensatz und soll dazu dienen, zu überprüfen, ob die Methoden auch auf

weniger Datensätze funktionieren und welche Besonderheiten dadurch entstehen.

- Historischer Datensatz **(V3)**: Der dritte Datensatz erstreckt sich über einen Zeitraum von vier Jahren und beinhaltet Benutzer, die während dieser Zeit aktiv waren und regelmäßig Antworten geschrieben haben. Dadurch soll es ermöglicht werden, das Benutzerinteresse über einen längeren Zeitraum berücksichtigen zu können.

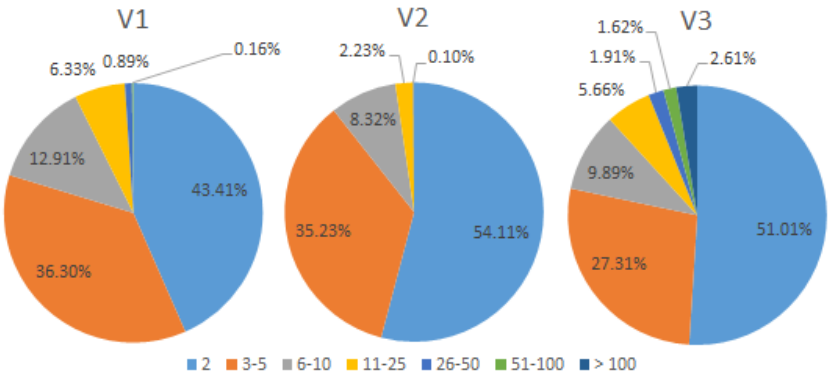


Abbildung 5.2: Anteil der Benutzer mit ihren Anzahl an Antworten innerhalb des Datensatzes

5.2 TF-IDF FRAGEPROFILE

In einer ersten Überprüfung wird Tf-idf zum Erstellen der benötigten Vektoren verwendet. Hierbei soll evaluiert werden, wie sich verschiedene Vorverarbeitungsschritte und die Reduzierung des Vokabulars auf die Ergebnisse des RS auswirken. Dies wird jeweils auf die drei verschiedenen Datensätze ausgeführt, um zusätzlich zu prüfen, ob die Größe des Datensatz, sowie die zeitliche Komponente eine Auswirkung auf die Auswahl der Parameter hat. Als Ausgangslage werden für die Berechnung der Benutzerprofile der einfache arithmetische Mittelwert verwendet, sowie ein einfaches Distanzmodell zwischen den Frage- und Benutzerprofil für das neuronale Netz zum Trainieren des Klassifizierers.

Die kompletten Ergebnisse für die verschiedenen Parametern können in der Anlage C eingesehen werden. Eine Zusammenfassung wird in Tabelle 5.2 dargestellt. Dort ist bereits zu sehen, dass ein zusätzliche Filterung auf bestimmte vorgegebene POS-Tags (Nomen, Verb, Adjektiv) eine leichte Reduzierung der Genauigkeit bewirkt. Dies ist bei dem Tf-idf darauf zurückzuführen, da dies bereits den Parameter *max_features* unterstützt, welches das verwendete Vokabular auf eine vorgegebene Größe reduziert. Da dies anhand der Wichtigkeit von Wörtern über den gesamten Textkorpus vorgenommen wird, ist dies das bessere Vorgehen zum reduzieren des verwendeten Vokabulars und somit die Vektorgröße eines Frageprofils.

Datensatz	Vorverarbeitung	Training	Validierung	Test
V1	ohne POS	0.777 ± 0.017	0.784 ± 0.017	0.779 ± 0.026
	mit POS	0.773 ± 0.017	0.781 ± 0.019	0.774 ± 0.025
V2	ohne POS	0.745 ± 0.007	0.772 ± 0.020	0.766 ± 0.052
	mit POS	0.741 ± 0.011	0.771 ± 0.029	0.772 ± 0.041
V3	ohne POS	0.844 ± 0.011	0.844 ± 0.012	0.854 ± 0.014
	mit POS	0.839 ± 0.012	0.841 ± 0.013	0.853 ± 0.013

Tabelle 5.2: Zusammenfassung der Tf-idf Ergebnisse. Mittelwert und Standardabweichung der erzielten Genauigkeit über verschiedene Einschränkung der Vokabulargröße

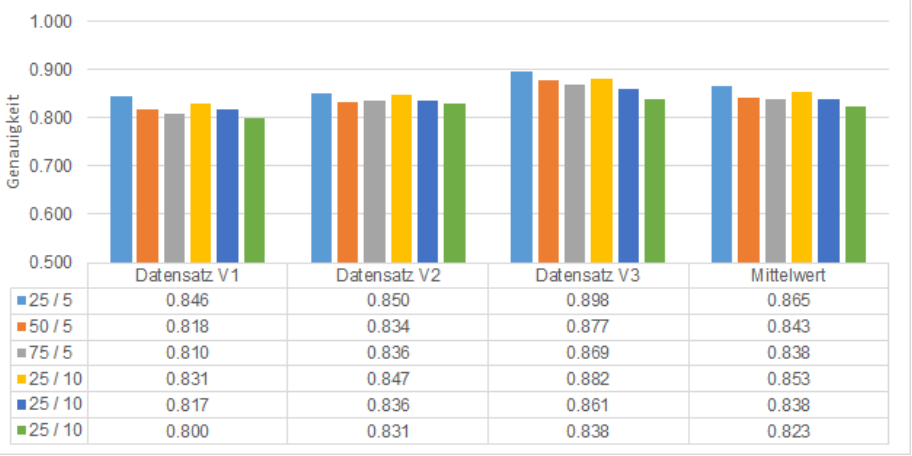
Die Auswirkung der Reduzierung des Vokabulars wird anhand der Werte für die Genauigkeit auf dem Datensatz V1 in Tabelle 5.3 detaillierter gezeigt, indem ausgehend von kompletten Vokabular (Nur Anwendung der Standard Vorverarbeitungsschritte bzw. zzgl. der POS Filter als Referenz) dieses sukzessive reduziert und die Genauigkeit gemessen wird. Es ist zu sehen, dass die Verwendung des kompletten Vokabulars das beste Ergebnis erzielt aber eine Reduzierung bis zu einem bestimmten Grad nur eine minimale Reduzierung der Genauigkeit bewirkt. Durch das Schrumpfen des Vokabulars auf ca. 50 Prozent (Vokabulargröße von 5000) reduziert sich die Genauigkeit auf den Trainings- und Validierungsdaten um maximal 0.2 Prozentpunkte und auf den Testdaten um maximal 0.8 Prozentpunkte. Selbst bei einer Reduzierung auf ca. 25 Prozent des Vokabulars (Vokabulargröße von 2500) findet eine Reduzierung von maximal 1.8 Prozentpunkte statt. Hierbei kann eine Abwägung zwischen der Genauigkeit und dem Speicherbedarf bei größeren Vektoren und damit verbunden die Laufzeit bei weiteren Rechenoperationen auf diese sinnvoll sein.

	Ohne POS			mit POS		
Vokabular	Training	Validierung	Test	Training	Validierung	Test
10029 / 9409[3]	**0.792**	**0.800**	**0.811**	**0.788**	**0.797**	**0.803**
5000	0.791	**0.800**	0.803	**0.788**	0.795	0.799
2500	0.788	0.794	0.792	0.785	0.796	0.788
1000	0.781	0.785	0.773	0.777	0.786	0.768
500	0.767	0.771	0.758	0.760	0.763	0.749
250	0.744	0.753	0.737	0.742	0.747	0.736

Tabelle 5.3: Ergebnisse für Tf-idf anhand des Datensatz V1 mit verschiedene Einschränkung der Vokabulargröße

5.3 DOC2VEC FRAGEPROFILE

In einem nächsten Schritt wird das Doc2Vec Verfahren evaluiert mit den im Kapitel 4.2.2 vorgestellten Varianten. Dies beinhaltet die ursprüngliche Form, welche jedem Dokument eine eindeutige ID vergibt während des Trainings, sowie die Erweiterung, welche als zusätzliche Features noch die hinterlegten Hashtags und/oder die Beantworter der Frage verwendet. Zusätzlich wird geprüft, wie sich die Wahl der Trainingsparameter (Die Größe des Vektors, sowie die Fenstergröße für die berücksichtigten benachbarten Wörter) auf die Ergebnisse auswirken. Wie bei dem Tf-idf Verfahren, werden in diesem Schritt ebenfalls der arithmetische Mittelwert für die Benutzerprofile verwendet.



	Datensatz V1	Datensatz V2	Datensatz V3	Mittelwert
25 / 5	0.846	0.850	0.898	0.865
50 / 5	0.818	0.834	0.877	0.843
75 / 5	0.810	0.836	0.869	0.838
25 / 10	0.831	0.847	0.882	0.853
25 / 10	0.817	0.836	0.861	0.838
25 / 10	0.800	0.831	0.838	0.823

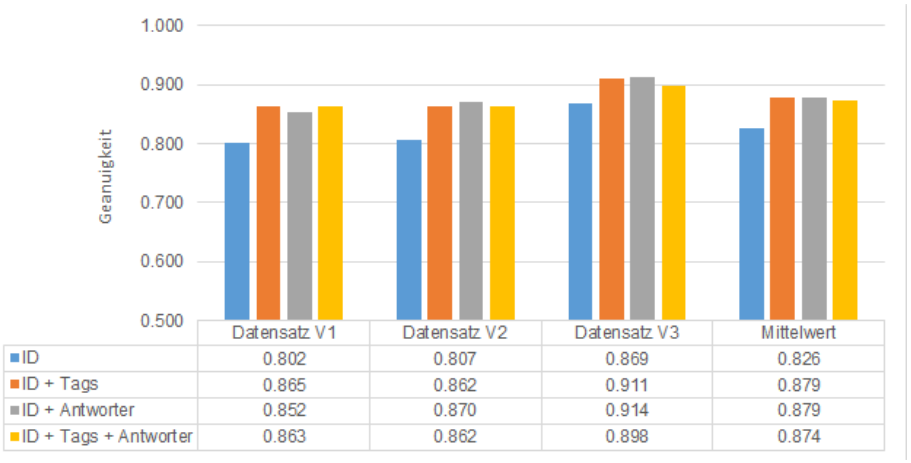
Abbildung 5.3: Mittelwert der Genauigkeit der verschiedenen Doc2Vec Parametereinstellung auf die Testdaten über die verschiedenen Verfahren.

Bei dem Doc2Vec Verfahren werden in der Regel eine Vektorgröße ab 100 empfohlen und je größer der Textcorpus ist während der Trainingsphase, desto besser kann die Semantik erkannt und dargestellt werden. Bei den drei verwendeten Datensätze wurden allerdings mit kleineren Vektoren bessere Ergebnisse erzielt. Die Abbildung 5.3 zeigt hierzu die durchschnittliche Genauigkeit über die verschiedenen Verfahren mit unterschiedlichen Parametern. Mit der Erhöhung der Vektorgröße nimmt die Genauigkeit auf den Testdaten leicht ab. Das beste Ergebnis erzielte hierbei eine Vektorgröße von 25 mit der Betrachtung der 5 benachbarten Wörtern zur Vorhersage des fehlenden Wortes. Bei einer Erhöhung der Vektorgröße auf 75 werden hierbei die Genauigkeit von 86.5% um 2.7 Prozentpunkte auf 83.8% reduziert. Auch eine Erhöhung der Fenstergröße für die benachbarten Wörter hat keinen positiven Effekt und reduziert bei allen Datensätzen die Genauigkeit, wobei dies weniger in das Gewicht fällt als die ausgewählte Vektorgröße.

Nachdem geprüft wurde, welche Vektorgröße Sinnvoll ist für die verwendeten Datensätze, werden noch auf die verschiedenen Varianten der zusätzlichen Features beim Trainieren des Doc2Vecs Modells überprüft. Diese wer-

den nun anhand der Vektorgröße von 25 überprüft und beinhalten folgende vier Kombinationen, welche bereits in 4.2.2 detaillierter erläutert wurden:

- **ID:** Pro Frage wird eine eindeutige ID vergeben.
- **ID + Tags:** Zusätzlich zu der eindeutigen ID werden noch die hinterlegten Hashtags der Frage berücksichtigt.
- **ID + Beantworter:** Zusätzlich zu der eindeutigen ID werden noch die Benutzer verwendet, welche die Frage beantwortet haben berücksichtigt.
- **ID + Tags + Beantworter:** Es werden sowohl die ID der Frage, die hinterlegten Hashtags und die Beantworter der Frage berücksichtigt.



	Datensatz V1	Datensatz V2	Datensatz V3	Mittelwert
ID	0.802	0.807	0.869	0.826
ID + Tags	0.865	0.862	0.911	0.879
ID + Antworter	0.852	0.870	0.914	0.879
ID + Tags + Antworter	0.863	0.862	0.898	0.874

Abbildung 5.4: Genauigkeit der verschiedenen Doc2Vec Verfahren auf die Testdaten für die Vektorgröße 25

Die ursprüngliche Idee von Doc2Vec ist, jedem Dokument eine eindeutige ID zu vergeben und diese beim Trainieren des Doc2Vec Modells zu berücksichtigen, um die Semantik des Dokumentes zu verstehen und daraus den Vektor abzuleiten. Durch das zusätzliche hinzufügen von weiteren Features wie die hinterlegte Hashtags der Fragen oder die Liste mit den Benutzern, welche die Frage beantwortet haben, kann die Genauigkeit weiter gesteigert werden. Dadurch wird nicht nur die Semantik der Frage widergespiegelt, sondern zusätzlich noch eine Relation zu den Hashtags hergestellt. Durch dieses Vorgehen ist es möglich, durch das Zuschalten der Hashtags gegenüber dem Standardvorgehen die Genauigkeit im Durchschnitt von 82.6% um 5.3 Prozentpunkte zu erhöhen auf 87.9%.

Der Unterschied zwischen dem Hinzufügen von Hashtags oder den Beantworter ist hingegen nur minimal und kann vernachlässigt werden. Auch die Kombination aus beiden Varianten bewirkt keine weitere Verbesserung. Es wird daher empfohlen, die Hashtags als zusätzliche Features zu verwenden, da diese pro Frage nur einmalig angelegt werden, wenn diese erstellt wird, während sich die Liste mit den Beantwortern für eine Frage sich ändern kann

durch neue Antworten. Ebenfalls spiegel das Frageprofil dadurch auch besser die eigentlich Frage wieder, da hier noch kein Bezug zu den Benutzern der Antworten hergestellt wird. Dadurch können Fragenprofile untereinander besser verglichen werden, um z.B. zusätzlich auch semantisch ähnliche Fragen zu finden.

	Ohne POS			mit POS		
Parameter	Training	Validierung	Test	Training	Validierung	Test
25 / 5	**0.856**	**0.854**	**0.865**	**0.837**	**0.838**	**0.844**
50 / 5	0.835	0.840	0.854	0.814	0.818	0.825
75 / 5	0.810	0.817	0.829	0.788	0.789	0.825
25 / 10	0.842	0.848	0.854	0.817	0.831	0.833
50 / 10	0.812	0.817	0.828	0.789	0.787	0.804
75 / 10	0.787	0.792	0.805	0.762	0.770	0.787

Tabelle 5.4: Ergebnisse für Doc2Vec mit Tags anhand des Datensatz V1 mit verschiedenen Trainingsparametern. Der erste Parameter steht für die Vektorgröße und der zweite Parameter für die Fenstergröße

Abschließend wird in Tabelle 5.4 nochmal anhand des Datensatz V1 gezeigt, wie die genauen Werte pro verwendeten Parametern sind anhand des Doc2Vec Verfahren mit der Verwendung der Hashtags. Weiterhin wird hierbei nochmal gegenübergestellt, wie sich die Verwendung der POS-Filter als zusätzliche Vorverarbeitungsschritt auf die Ergebnisse auswirkt. In allen Fällen bringt eine kleine Vektorgröße von 25 mit einem kleinen Fenster die besten Ergebnisse. Das hinzufügen des POS-Filtern hat auch bei der Verwendung von Doc2Vec keinen positiven Effekt und reduziert in allen Fällen die Genauigkeit.

5.4 VARIANTEN FÜR BENUTZERPROFILE

Nachdem in den zwei vorherigen Abschnitte geprüft wurde, was welche Parameter eine hohe Genauigkeit erzielt haben, um mit Tf-idf oder Doc2Vec das Frageprofil darzustellen, soll nun die Erstellung der Benutzerprofile genauer untersucht werden. Durch Anwendung der Maximum-Methode (4.3.1), des gewichteten arithmetisches Mittel (4.3.2), sowie Beschränkung auf die letzten N Fragen (4.3.3), soll untersucht werden, ob dies bessere Ergebnisse erzielt gegenüber den normalen arithmetischen Mittelwert. Hierbei wird auch eine genauere Betrachtung der vorhandenen Datensätze vorgenommen, da diese aufgrund ihrer verschiedenen Eigenschaften unterschiedlichen zeitlichen Verlauf der beantworten Fragen von Benutzern aufweisen.

Als Ausgangsbasis werden die Benutzerprofile verwendet, welche mit folgenden Parametern erstellt wurden, da diese gute Ergebnisse erzielt haben in der vorherigen Evaluierungsphase auf allen drei Datensätzen:

- Tf-idf mit einer maximalen Vokabulargröße von 5000.
- Doc2Vec mit Berücksichtigung der Hashtags, sowie einer Vektorgröße von 25 und die Fenstergröße für benachbarte Wörter von 5.

λ	Anzahl Fragen
0.01	46/50
0.025	45/50
0.05	41/50
0.10	28/50
0.25	11/50

Tabelle 5.5: Anzahl der Fragen, welche 95% der gesamten Gewichtung ausmachen für verschiedene λ Werten von einer Gesamtanzahl von 50 beantworteten Fragen des Benutzers.

Für die Berechnung der Benutzerprofile wurden dabei verschiedene λ-Werte getestet für den exponentiellen Zerfall, um herauszufinden, wie stark sich der Zerfalls und damit verbunden die Anzahl der vergangenen beantworteten Fragen, sich auf das Benutzerprofil und die Ergebnisse des RS auswirken. Die Tabelle 5.5 zeigt die für die Evaluierung gewählten λ Werte, sowie als Beispiel die Anzahl der Fragen, welche 95 Prozent der gewichteten Summe ausmachen bei 50 beantworten Fragen. Bei einem λ von 0.25 würden somit die zeitlich aktuellsten 11 der 50 Fragen 95 Prozent des gewichteten arithmetischen Mittelwert ausmachen.

Datensatz	Methode	Maximum	Gewichteter Mittelwert	Letzten N Fragen
V1	Tf-idf	0.7975	0.8039	**0.8054**
	Doc2Vec	0.8019	**0.8686**	0.8663
V2	Tf-idf	0.8150	**0.8228**	**0.8228**
	Doc2Vec	0.7953	0.8622	**0.8701**
V3	Tf-idf	0.8297	**0.8626**	0.8591
	Doc2Vec	0.7312	**0.9126**	0.9070
	Mittelwert	0.7951	**0.8555**	0.8551

Tabelle 5.6: Beste Genauigkeit auf den Testdaten der drei Berechnungsmöglichkeiten für das Benutzerprofil, ausgeführt für die verschiedenen Datensätze und Doc2Vec inkl. Hashtags und Tf-idf als Basis der Frageprofile

In Tabelle 5.6 wird eine Zusammenfassung der besten erzielten Genauigkeit unter der Verwendung von verschiedenen λ Werte gezeigt. Dies wurde

pro Datensatz und Methode durchgeführt. Hierbei ist bereits zu sehen, dass die Maximum-Methode eine schlechtere Genauigkeit erzielt gegenüber der Mittelwert-Methode. Vor allem bei der Verwendung von Doc2Vec macht sich dies bemerkbar, was aber auch zu erwarten war, da die Höhe eines einzelnen Vektorwertes gegenüber Tf-idf keine Aussagekraft über die Bedeutung eines Wortes besitzt. Die negativen Werte sind für Doc2Vec genauso wichtig, weshalb die Verwendung eines Mittelwertes bessere Ergebnisse erzielt. Auch mit der Nutzung der Methode, welche eine Einschränkung auf die letzten N Fragen vornimmt, kann eine hohe Genauigkeit erreicht werden. Im Durchschnitt ist allerdings der gewichtete Mittelwert am besten geeignet, was sich auch auf dem Datensatz V3 bemerkbar macht, welcher stärker den zeitlichen Benutzerverlauf beinhaltet.

Im weiteren wird nun der gewichteten Mittelwert genauer untersucht. In Abbildung 5.5 werden hierzu die einzelne Werte mit den verschiedenen λ Werte für das Doc2Vec Verfahren gezeigt. Ein λ von 0.0 entspricht keine Abschwächung historischer Daten und ist somit gleichzusetzen mit den normalen arithmetischen Mittelwert, wodurch jedes Frageprofil den gleichen Einfluss auf das Benutzerprofil erlangt. Zu sehen ist, dass durch die Verwendung eines λ und dem gewichteten Mittelwert eine Steigerung der Genauigkeit erzielt werden kann. Allerdings ist die Wahl des λ abhängig von den vorhandenen Daten.

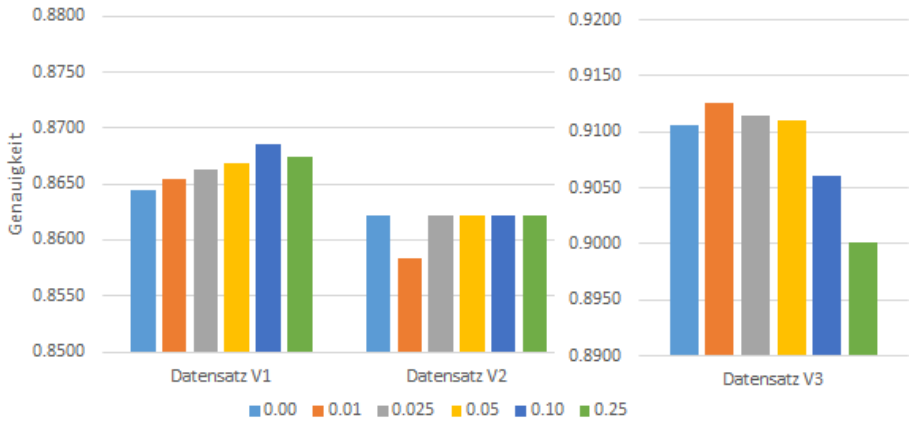


Abbildung 5.5: Genauigkeit der verschiedenen λ Werte auf die Testdaten für die Doc2Vec Variante und den unterschiedlichen Datensätzen

Bei dem kleinen Datensatz V2, welcher nur wenige beantworte Fragen besitzt und generell nur wenige Benutzer vorhanden sind, welche mehrere Fragen beantworten haben, ist keine Steigerung der Genauigkeit erreicht worden. Dies ist darauf zurückzuführen, dass sich durch die eher wenigen Antworten eines Benutzers noch keine wirkliche Historie aufgebaut hat und durch den kurzen Zeitraum die Fragen selbst sich zu ähnlich sind.

Bei dem größeren Datensatz V1 konnte mit der Steigerung des λ auch die Genauigkeit leicht erhöht werden. Das beste Ergebnis wurde hier mit einem λ von 0.10 erreicht. Bei dem Datensatz V3, welches die meisten Antworten und Poweruser mit vielen Antworten besitzt hingegen, wurde mit dem λ von 0.10 ein schlechteres Ergebnis gegenüber den normalen Mittelwert. Bei einer Steigerung auf 0.25 fällt die Genauigkeit dabei noch weiter ab. Bei einem kleineren Wert zwischen 0.01 und 0.05 konnte jedoch ebenfalls eine Steigerung ermittelt werden, wobei dies weniger als 1 Prozentpunkt ausmacht.

Bei dem gleichen Anwendungsszenario mit der Verwendung von Tf-idf als Basis konnte keine nennenswerte Verbesserung durch die Verwendung des Zerfallsterm und dem gewichteten Mittelwert gemessen werden. Wie in Abbildung 5.6 zu sehen ist, konnte lediglich in einem Fall eine minimale Erhöhung der Genauigkeit ermittelt werden mit einem λ von 0.25 auf den Datensatz V1. Dieser λ Wert verursacht allerdings bei den anderen beiden Datensätzen einen Verlust von ca. 2 Prozentpunkte. Bei dem Datensatz V3 ist, wie auch bereits bei Doc2Vec in Abbildung 5.5 zu sehen war aufgetreten, dass ab einem λ von 0.10 die Genauigkeit fällt. Durch den hohen λ Wert werden nur noch einige der zuletzt beantworteten Fragen berücksichtigt, wodurch dies evtl. zu ungenau wird für das Benutzerprofil. insbesondere bei Benutzer, welche sich für mehrere verschiedene Themengebiete interessieren und Fragen schubweise (z.B. 10 Fragen zu Thema Datenbanken beantworten, anschließend 10 Fragen zum Thema Statistik beantwortet, etc.) beantworten, kann dies zu Problemen führen, da dann zu wenige beantworte Fragen berücksichtigt werden, um mehrere Interessen gleichzeitig abzubilden.

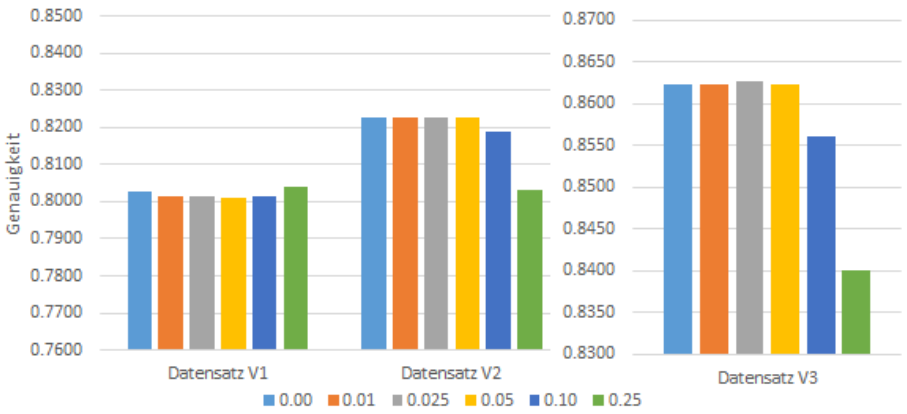


Abbildung 5.6: Genauigkeit der verschiedenen λ Werte auf die Testdaten für die Tf-idf Varianten und den unterschiedlichen Datensätzen

In Abbildung 5.7 wird angezeigt wie viele Fragen der vorhandenen Fragen aus dem jeweiligen Datensatz die Benutzer als Empfehlung bekommen, ausgehend von ihrem zeitlichen letzten Benutzerprofil. Diese sind zwar nicht direkt vergleichbar, da die Datensätze unterschiedliche Benutzer und Fragen

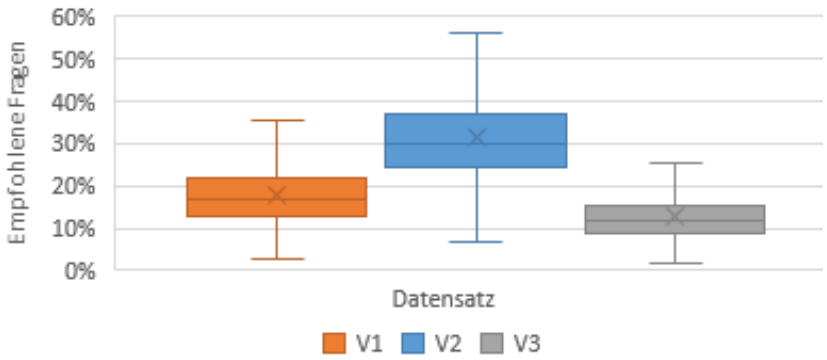


Abbildung 5.7: Anteil an positive Empfehlungen von Fragen für Benutzer

besitzen, sowie ihr eigenes trainierten RS aber dennoch ist ersichtlich, dass auf dem Datensatz V2, wo hauptsächlich Benutzer mit wenigen beantworten Fragen vorhanden sind, diese Benutzerprofile allgemeiner gehalten werden und dementsprechend mehr Fragen empfohlen werden. Die Hälfte der Benutzer bekommt in diesem Datensatz maximal 30% der Fragen empfohlen. In dem Datensatz V3, welcher am meisten Benutzerdaten besitzt, schrumpfen die Intervalle und die Hälfte der Benutzer bekommen in diesem Datensatz maximal 12% der Fragen empfohlen. Aufgrund der umfangreicheren Daten können detailliertere Benutzerprofile erstellt werden und gezielter Empfehlungen vorgenommen werden.

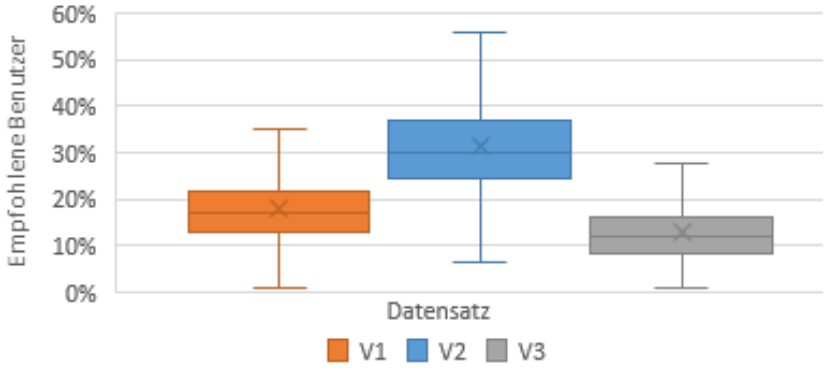


Abbildung 5.8: Anteil an positive Empfehlungen von Benutzer für Fragen

Als Gegenstück wird in Abbildung 5.8 abgebildet, wie viele Benutzer für eine Frage empfohlen werden. Dies Verhält sich ähnlich, wie bei der vorherigen Betrachtung für die Empfehlung von Fragen für Benutzer. Auch hier können im Datensatz V3 aufgrund der detaillierteren Benutzerprofile auch präzisere Empfehlungen vorgenommen werden.

5.5 BENUTZERPROFILE VON TESTBENUTZERN

Benutzer ID	476	332059
Anzahl Antworten	1022	1245
Top 3 Hashtags	PHP (0.40) Python (0.26) Javascript (0.25)	Outlook (0.75) C# (0.29) VBA (0.25)

Tabelle 5.7: Informationen für die zwei verwendeten Testbenutzer

Anhand von zwei Testbenutzern, welche in dem Datensatz V3 am meisten Antworten erfasst haben, soll das Verhalten der Benutzerprofile etwas genauer untersucht werden. In Tabelle 5.7 werden einige Informationen zu diesen beiden Benutzer aufgelistet. Beide Benutzer haben jeweils über 1000 Fragen beantwortet in diesem Datensatz. Es wurden die Top 3 Hashtags der Benutzer ermittelt anhand der beantworteten Fragen mit den jeweiligen Anteil über alle seiner beantworteten Fragen, um dessen Interessengebiete darzustellen. Der zweite Benutzer ist gegenüber dem ersten Benutzer stärker auf ein Thema konzentriert. 75% seiner beantworten Fragen besitzen den Hashtag *Outlook* und anschließend folgen die zwei Programmiersprachen *C#* (29%) und *VBA* (25%). Bei dem ersten Benutzer ist der meistgenutzte Hashtag *PHP* nur mit 40% vertreten, gefolgt von *Python* mit 26%. Während PHP und Javascript in der Webentwicklung genutzt werden, ist Python dort eher selten vertreten. Es besteht also die Wahrscheinlichkeit, dass dieser Benutzer in verschiedenen Themen Interesse besitzt und weniger konzentriert auf einen Bereich, wie bei dem zweiten Benutzer.

In Abbildung 5.9 werden in den beiden Grafiken für die Benutzer der Verlauf angezeigt, wie hoch die Kosinus-Ähnlichkeit gegenüber dem zeitlichen vorherigen Benutzerprofils des Benutzers ist. Daraus lässt sich Analysieren, wie hoch die Veränderung der Benutzerprofile durch die Wahl des λ beeinflusst wird. In der Startphase ist der Verlauf bei den drei unterschiedlichen λ Werte noch ziemlich identisch, da es zu diesem Zeitpunkt noch nicht viele beantwortete Fragen gibt und somit die Zerfallformel nur eine geringe Wirkung zeigt. Mit steigender Anzahl von beantworteten Fragen stabilisiert sich die Ähnlichkeit gegen 100 Prozent bei Verwendung des normalen arithmetischen Mittelwert, was in diesem Fall einem λ von 0.0 entspricht. Das Benutzerprofil verändert sich immer weniger, da es die kompletten historischen beantworteten Fragen gleich gewichtet, wodurch es zwar weniger Anfällig wird gegen Ausreißer aber dies auch bedeutet, dass es für einen Benutzer schwieriger wird, sein Benutzerprofil anzupassen, falls sich das Interesse an Themen ändert. Dies ändert sich mit der Verwendung von höheren λ Werten. Je höher diese sind, desto stärker ändert sich auch das Benutzerprofil, da es jüngere beantworte Fragen priorisiert in der Gewichtung und sich auf das aktuelle Interesse des Benutzer konzentriert.

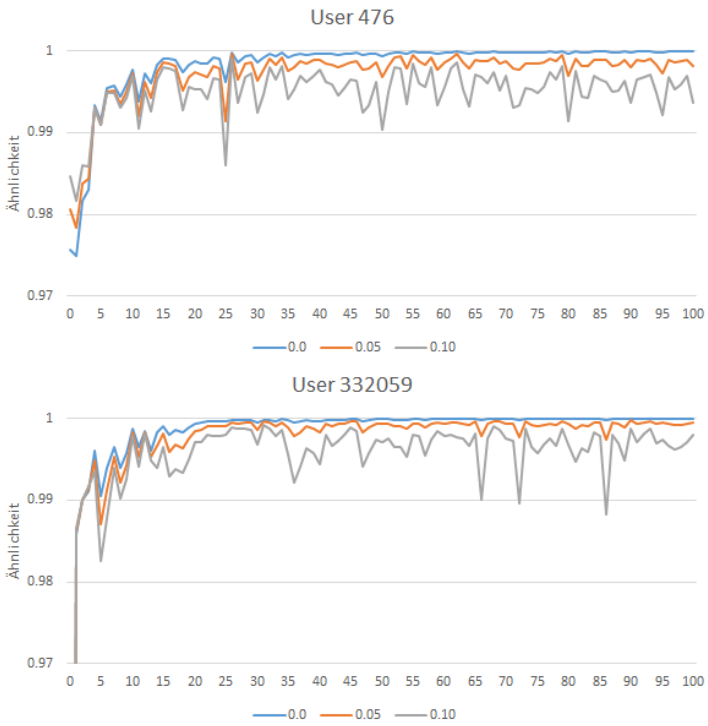


Abbildung 5.9: Ähnlichkeiten der Benutzerprofil gegenüber den vorherigen Benutzerprofils des Benutzers mit verschiedenen λ Werten

In der Abbildung 5.10 wird gezeigt, wie hoch die Kosinus-Ähnlichkeit während der Trainingsphase ist zwischen dem Frageprofil, welcher der Benutzer beantwortet hat und dem dazugehörigen Benutzerprofil, welches zu diesem Zeitpunkt aktiv war. Dies wird ebenfalls mit den negativen Daten angezeigt, was einer naheliegende Frage entspricht, welche der Benutzer aber nicht beantwortet hat. Zur besseren Übersicht wurden nur die ersten 100 beantworten Fragen der Benutzer verwendet. Zu sehen ist, dass bei dem ersten Benutzer die durchschnittliche Ähnlichkeit zwischen den beiden Vektoren bei 60% liegt für die positiven Daten, während dies bei dem zweiten Benutzer mit 80% deutlich höher liegt. Das Benutzerprofil ist somit besser auf seine beantworteten Fragen abgestimmt, was durch die bereits beschriebene Möglichkeit besteht, da sich dieser Benutzer mehr auf ein Thema beschränkt, während der erste Benutzer in mehren Themen Fragen beantwortet und das Benutzerprofil somit diese Vielfalt darstellen muss. Auch bei den negativen Daten macht sich dies bemerkbar, indem diese bei dem ersten Benutzer eine durchschnittliche Ähnlichkeit von 40% besitzen, während es bei dem zweiten Benutzer nur 20% sind. Dadurch liegen diese bei dem ersten Benutzer dichter zusammen, wodurch es auch zu Überschneidungen kommen kann und das negative Beispiel eine höhere Ähnlichkeit aufweist, als eigentlich positive Beispiel. Im Fall des zweiten Benutzer findet durch den höheren Abstand zwischen den positiven und negativen Daten eine bessere Trennung statt, was dem Klassifizierer erleichtert, die richtige Vorhersage zu erstellen.

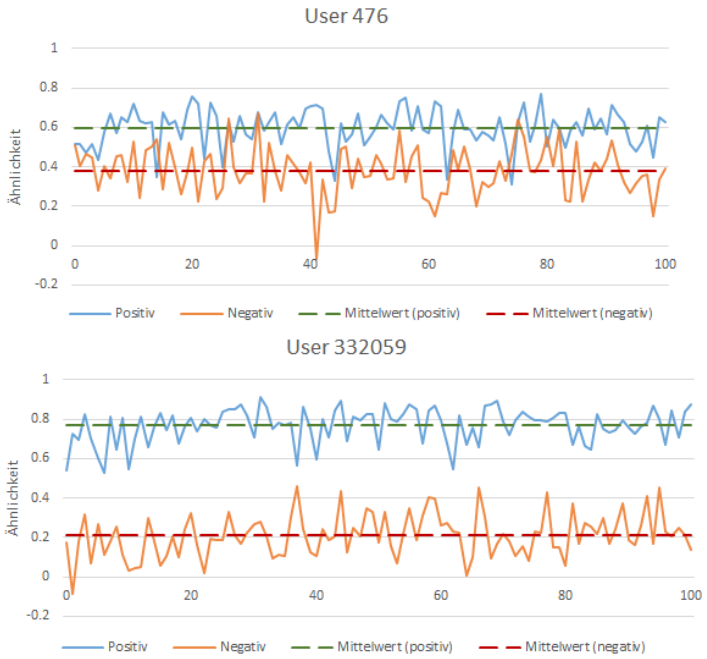


Abbildung 5.10: Ähnlichkeiten des Benutzerprofils mit dem korrespondierenden Fragenprofil während der Trainingsphase, sowie der durchschnittlichen Ähnlichkeit pro positiver und negativer Datensatz bei Verwendung von $\lambda = 0.05$

Benutzer	Label	Genauigkeit	Trefferquote	F1-Maß	Datensätze
476	Negativ	0.84	0.76	0.80	246
	Positiv	0.78	0.86	0.82	246
	Mittelwert	0.81	0.81	0.81	492
332059	Negativ	1.00	0.98	0.99	444
	Positiv	0.98	1.00	0.99	444
	Mittelwert	0.99	0.99	0.99	888

Tabelle 5.8: Ergebnisse für die beiden Testbenutzer auf deren hinterlegten Testdaten

Dies macht sich auch in den Ergebnisse in der Tabelle 5.8 bemerkbar. Hierzu wurde für das trainierte RS geprüft, welche Ergebnisse dieses auf die hinterlegten unbekannten Testdaten der beiden Benutzer erzielt. Wie erwartet konnte bei dem zweiten Benutzer aufgrund der breiteren Trennung zwischen den positiven und negativen Daten eine sehr hohe Genauigkeit von 99% erreicht werden. Bei dem ersten Benutzer wurde hingegen nur eine Genauigkeit von 81% erreicht, was dennoch ausreichend ist. 78% der Fragen, welcher das RS als Empfehlung vorhergesagt hat, waren auch tatsächliche Fragen, die der Benutzer beantwortet hat und von den tatsächlichen Fragen, welcher der Benutzer beantwortet hat wurden 86% der Fragen auch als Empfehlung gefunden. Das RS hat somit einen Großteil der zugeordneten Fragen gefunden und spricht mehr Empfehlungen aus als evtl. relevant sind.

Abschließend sollte noch betrachtet werden, wie sich die Genauigkeit auswirkt auf Benutzer, welche wirklich einen harten Schnitt machen und sich für neuen Themen interessieren. Hierzu wurde zu Testzwecken ein neuer Benutzer erstellt, welcher abwechselnd das Interesse von drei bekannten Benutzer übernimmt, welche Fragen in unterschiedlichen Themengebieten beantwortet haben. In der Abbildung 5.11 wird die Wahrscheinlichkeit für eine Empfehlung bei gegebenen Frageprofil und zu dem Zeitpunkt gültigen Benutzerprofil gezeigt. Auf der der Y-Achse zeigt die Hilfslinie von 0.5 an, ab wann eine Übereinstimmung für eine Empfehlung vorliegt (Werte oberhalb der 0.5 Linie) bzw. wenn es keine Übereinstimmung gibt (Werte unterhalb der 0.5 Linie). Je höher der Wert ist, desto sicherer ist sich das RS. Auf der X-Achse ist zu sehen, ab wann die Fragen eines neuen Benutzer chronologisch eingeordnet wurde. Hierbei ist erkennbar, dass bei einem Wechsel das RS die Fragen zu dem neuen Thema dem Benutzer nicht empfehlen wird. Dies ist aber auch zu erwarten, da das RS die neuen Benutzerinteresse erst wieder erlernen muss, was einige neue beantworte Fragen benötigt. Hierbei machen sich die verschiedenen Verfahren zum Erstellen des Benutzerprofils bemerkbar. Bei höheren λ Werten werden die jüngsten Fragen stärker betrachtet, weshalb wieder schneller das neue Benutzerinteresse abgebildet werden kann, während kleinere λ Werte bzw. das normale arithmetische Mittelwert weiter in die Vergangenheit schaut und somit länger benötigt sich wieder zu stabilisieren.

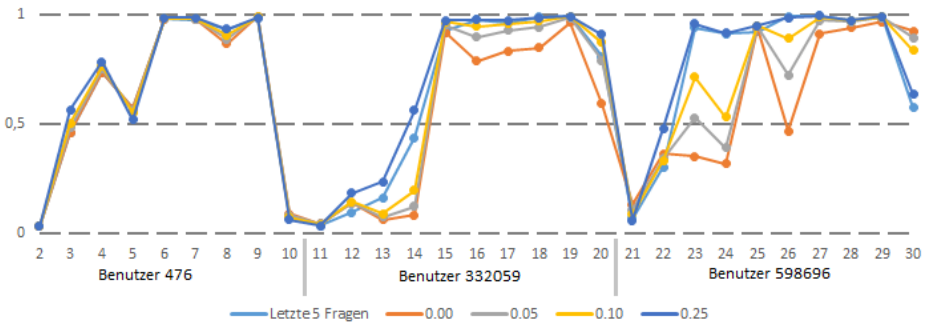


Abbildung 5.11: Verlauf der Genauigkeit des simulierten Benutzers

In der Tabelle 5.9 werden wie zuvor zweimal ein Wechsel des Benutzerinteresse simuliert. Dabei werden unterschiedliche Fenstergrößen verwendet, wie lange das Interesse besteht, bevor die beantworteten Fragen eines neuen Benutzers verwendet werden. Verwendet wurden die gleiche Benutzer wie in Abbildung 5.11. Zu sehen ist, dass bei einem solchen Benutzerverhalten, der normale arithmetische Mittelwert schlechter gegenüber den anderen Varianten abschneidet. Bei regelmäßigen Veränderungen, welches in Form einer kleinen Fenstergröße dargestellt ist, zielt ein hoher λ Wert die besten Ergebnisse, da dieser am besten auf größere Schwankungen reagiert und sich schneller an das neue Benutzerinteresse orientiert. Mit steigender Fenstergröße nimmt dieser Effekt aber etwas ab und kleinere λ Werte erzie-

len bessere Ergebnisse, da diese von normalen Schwankungen nicht zu stark beeinträchtigt werden.

	Fenstergröße für Interessenwechsel				
	5	10	20	50	100
Mittelwert	0.5714	0.5862	0.7458	0.7248	0.7759
Letzte 5 Fragen	0.5000	0.6897	0.7966	0.8121	0.8562
Letzte 10 Fragen	0.5714	0.6207	0.7627	**0.8658**	0.8863
Gew. Mittelwert ($\lambda = 0.05$)	0.5714	0.6552	0.7797	0.8456	0.8997
Gew. Mittelwert ($\lambda = 0.10$)	0.5714	0.7241	0.7966	**0.8658**	**0.9097**
Gew. Mittelwert ($\lambda = 0.25$)	**0.7857**	**0.7586**	**0.8305**	0.8523	0.8729

Tabelle 5.9: Genauigkeit bei verschiedenen Fenstergrößen für ein Interessenwechsel, welches zweimal durchgeführt wurde

5.6 VERWENDUNG VON EXTERNEN BENUTZERDATEN

In diesem Abschnitt soll untersucht werden, welche Auswirkung die externen Benutzerdaten auf die Empfehlungen haben. Hierbei wurde sich auch wieder auf das Doc2Vec Verfahren für Fragenprofile beschränkt, sowie ein λ von 0.05 für das Erstellen der Benutzerprofile. Die externen Benutzerdaten werden mit dem gleichen Doc2Vec Modell in einen Vektor transformiert, wie auch die Fragenprofile. Daher ist zu beachten, dass Vokabular, welches in den externen Daten verwendet wird aber während der Trainingsphase auf den Fragen unbekannt ist, dies ignoriert wird bei der Erstellung der Vektoren. Dadurch ist es z.B. möglich, dass eine Webseite, welche in einer anderen Landessprache erfasst wurde nicht gut abgebildet werden kann, da es nicht das bekannte englische Vokabular verwendet.

In Abbildung 5.12 wird gezeigt, wie hoch die Kosinus-Ähnlichkeit zwischen dem Vektor der externen Benutzerdaten und der ersten beantworteten Frage des Benutzers. Zwischen den einzelnen Datensätze gibt es hierbei keine große Abweichungen, obwohl diese unterschiedliche trainierte Doc2Vec Modelle verwenden, welcher nur die Fragen innerhalb des Datensatzes berücksichtigt hat. Im Durchschnitt liegt die Ähnlichkeit zwischen 32% und 34%. Der Bereich der Ähnlichkeiten ist dabei sehr breit abgedeckt. So gibt es zum einen Übereinstimmung mit einer sehr hohen Ähnlichkeit von über 75% aber auch einige mit keiner bzw. negativen Ähnlichkeit. Bei einer manuellen Betrachtung der Webseite der drei Benutzern mit der geringsten Ähnlichkeit fällt auf, dass diese keinen Bezug zu technischen Themen besitzen, wie sie in den Stackoverflow Datensatz vertreten sind. Bei den Benutzern

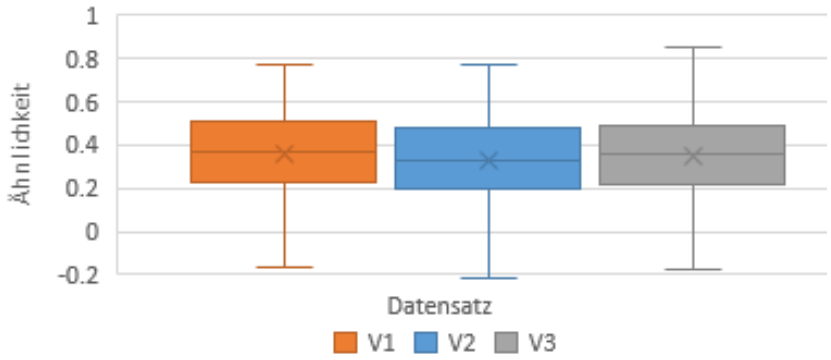


Abbildung 5.12: Ähnlichkeit zwischen dem externen Benutzerdaten und der ersten beantworten Frage des Benutzers

mit den höchsten positiven Ähnlichkeiten hingegen, ist einer der Webseiten ein Blog über die Verwendung von *pydev*, welches eine Entwicklungsumgebung für Python ist. Bei den anderen beiden Benutzern werden auf den Webseiten deren Lebenslauf dargestellt. Interessant ist hierbei, dass ein Benutzer seinen Lebenslauf in einer Fremdsprache verfasst hat aber aufgrund von Schlüsselwörtern, welche Produkte beschreiben (z.B. Python oder Java), diese dennoch ausreichen für eine hohe Übereinstimmung.

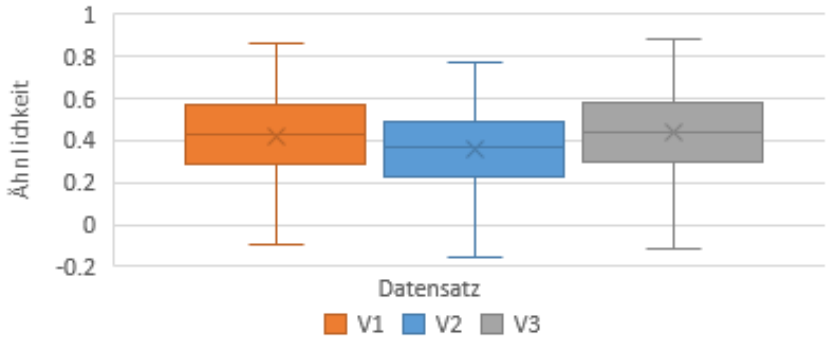


Abbildung 5.13: Ähnlichkeit zwischen dem externen Benutzerdaten und dem aktuellen Benutzerprofil

Zum Vergleich wird in Abbildung 5.13 noch die Ähnlichkeit zwischen dem Vektor der externen Benutzerdaten und dem aktuellen Benutzerprofil dargestellt. Gegenüber den Ähnlichkeiten zur ersten Frage, sind diese leicht gestiegen. Da die Webseiten zeitlich nicht vor der ersten Frage extrahiert wurden, kann es vorkommen, dass diese aktueller sind und somit näher an dem aktuellen Interessen des Benutzers liegt, als es damals war. Dies könnte z.B. der Fall sein, wenn sich die Lebensläufe auf den Webseiten verändert haben und neue aktuelle Interessen besitzen, welche dann auch das aktuelle Benutzerprofil widerspiegelt.

Um den Mehrwert der Anbindung von externen Daten zu messen, wird die Genauigkeit ermittelt, indem einem Benutzer seine erste Frage anhand des initialen externen Benutzerprofils als positive Empfehlung erhält.

- Datensatz V1: 35.95% der Benutzer mit externen Daten wurden ihre erste Frage positiv Empfohlen.
- Datensatz V2: 45.26% der Benutzer mit externen Daten wurden ihre erste Frage positiv Empfohlen.
- Datensatz V3: 25.77% der Benutzer mit externen Daten wurden ihre erste Frage positiv Empfohlen.

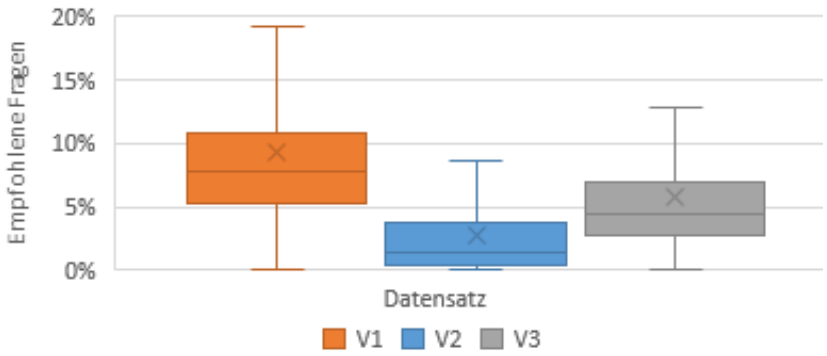


Abbildung 5.14: Empfohlene Fragen anhand des initialen externen Benutzerprofils

Zusätzlich kann es sinnvoll sein, zu betrachten, wie viele der vorhandenen Fragen im Datensatz einem Benutzer anhand des externen Benutzerprofils empfohlen wird. Hierzu wurden alle Fragen mit dem jeweiligen Benutzerprofil in das RS verwendet und berechnet, wie hoch der Anteil der Fragen ist, was in Abbildung 5.14 gezeigt wird. Gegenüber den gleichen Vorgehen mit den letzten aktiven Benutzerprofil in Abbildung 5.7 aus dem vorherigen Abschnitt, anstatt den initialen externen Benutzerprofil, haben sich die Verhältnisse verändert. Auf dem kleinen Datensatz V2, bei dem vorher die meisten Empfehlungen ausgesprochen wurden, werden nun am wenigstens Empfehlungen stattfinden. Der Datensatz V1 liegt hierbei zwar höher aber generell fällt der Anteil an Fragen, welche als Empfehlungen berücksichtigt werden gering aus. So ist der Median im Datensatz V1 7.71% (entspricht 2138 Fragen), bei dem Datensatz V2 1.38% (entspricht 69 Fragen) und in dem Datensatz V3 nur 4.50% (entspricht 362 Fragen).

6 ZUSAMMENFASSUNG UND AUSBLICK

In diesem abschließenden Kapitel wird eine Zusammenfassung über das Konzept des Recommender System (RS) erstellt und die Ergebnisse der durchgeführten Evaluierung. Anschließend wird noch ein Ausblick erstellt über weitere offene Themen, um das RS zu ergänzen für weitere Fragestellungen, sowie evtl. weitere Optimierung durch zusätzliche Maßnahmen.

6.1 ZUSAMMENFASSUNG

In dieser Arbeit wurde ein Konzept für ein RS erstellt, welches für eine interne Community Question Answering (CQA) Plattform der Firma PRODYNA SE eine Zusammenführung zwischen unbeantworteten Fragen und Benutzer, welche Interesse in dem Themengebiet der Frage besitzen. Dies ist sowohl aus Sicht des Benutzers möglich, welche eine Liste von interessanten Fragen bekommt, sowie aus Sicht einer neue Frage, welche an potentiellen Benutzern weitergeleitet wird. Ebenfalls sollte überprüft werden, ob durch die Verwendung zusätzlicher Benutzerdaten außerhalb dieses System eine Steigerung der Empfehlungsgenauigkeit vorgenommen werden kann bei neuen Benutzer, zu denen innerhalb des Systems noch nicht viel bekannt ist. Die Ausgangssituation der CQA-Plattform, sowie die Anforderungen an das RS wurde hierzu in Kapitel 2 genauer erläutert.

Im nachfolgenden Kapitel 3 wurden Grundlagen erläutert, welche für die Implementierung notwendig sind. Da es sich um Textdaten handelt, wurden auf notwendige Vorverarbeitungsschritte eingegangen, um diese zu bereinigen und anschließend in einen Vektor umzuwandeln, welcher für die weiteren Machine Learning (ML) Verfahren genutzt werden kann. Da in dieser Arbeit das RS als ein Klassifizierungsproblem betrachtet wird, wurden ebenfalls noch Metriken erläutert, welche für die Evaluierung des Systems relevant sind.

In dem Kapitel 4 wurde darauf eingegangen, was berücksichtigt werden sollte, um ein solches RS zu implementieren und an welchen Stellen Parameter verändert werden können, um evtl. bessere Ergebnisse zu erzielen bei den Empfehlungen. Für die Erstellung der Frageprofile wurde sich für term frequency - inverse document frequency (Tf-idf) mit verschiedenen Beschränkungen der Vokabulargröße als Ausgangsbasis entschieden, sowie Doc2Vec Verfahren mit verschiedenen Erweiterungen und Parametereinstellungen.

Für die Erstellung der Benutzerprofile werden die Frageprofile verwendet, welcher der Benutzer in der Vergangenheit beantwortet hat. Die Berechnung besteht dabei aus einer Zerfallfunktion und einer Aggregierungsfunktion. Bei der Zerfallfunktionen steht zur Auswahl der exponentielle Zerfall, kein Zerfall, sowie eine Einschränkung auf die letzten n Fragen. Bei dem exponentielle Zerfall werden jüngere beantwortete Fragen höher gewichtet gegenüber ältere, wodurch sich mehr auf aktuelle beantworte Fragen konzentriert wird aber die vergangene nicht komplett vergessen werden. Bei der Verwendung von keinem Zerfall, werden alle beantwortete Fragen des Benutzer gleich behandelt, unabhängig davon wie alt und relevant diese noch sind. Bei der dritten Variante wird ein harter Schnitt vorgenommen und es werden nur die letzten n beantwortete Fragen des Benutzers berücksichtigt. Es wird dadurch nur das aktuelle Interesse des Benutzers widergespiegelt und die Vergangenheit komplett ignoriert.

Die Aggregierungsfunktionen nutzt die gewichteten Fragenprofile und bilden daraus das fertige Benutzerprofil. Verwendet wurden hierbei das Maximum, welches über die Fragen hinweg den maximalen Wert pro Vektoreintrag ermittelt, sowie das gewichtete arithmetische Mittel, welches den Mittelwert über die Vektoren berechnet, ausgehend von der Zerfallfunktion bzw. das normale arithmetische Mittel, wenn keine Zerfallfunktion verwendet wurde.

Weiterhin wurde in dem Kapitel 4 noch auf die Architektur des verwendeten künstliches neuronales Netz (KNN) als Klassifizierungsmodell, sowie die Erstellung der Trainings- und Testdaten, um dieses Modell zu Trainieren und Evaluieren zu können. Es wurde ein einfaches KNN verwendet, welche zwei Vektoren (dem Frage- und Benutzerprofil) der gleiche Größe als Eingabe entgegennimmt, daraus die Kosinus-Ähnlichkeit berechnet und als Ausgabe die Wahrscheinlichkeit für eine positive Empfehlung ausgibt. Die Aufteilung zwischen den Trainingsdaten, welche das KNN zum trainieren seiner Gewichte verwendet und den unbekannten Testdaten, welche zur Evaluierung verwendet werden, findet anhand der Zeitachse des Antwortdatum statt.

Abschließend wurde in dem Kapitel 5 die Evaluierung des RS mit verschiedenen Einstellungen vorgenommen. Dies wurde auf drei verschiedene Datensätze angewandt, um unterschiedliche Szenarien zu berücksichtigen. Als Daten wurden Fragen und Antworten der öffentlichen Plattform Stackoverflow verwendet, sowie die dort hinterlegten Webseite von Benutzern zur Simulation der zusätzlichen externen Benutzerdaten. Die Evaluierung wurde sequentiell vorgenommen beginnend mit der Vorverarbeitungsschritte und Fragenprofilen, anschließend den verschiedenen Möglichkeiten zur Berechnung der Benutzerprofile, der zusätzlichen Anbindung von externen Benutzerdaten, sowie die detaillierte Betrachtung von ausgewählten Benutzern.

Bei der initialen Evaluierung der Frageprofile wurde festgestellt, dass die Verwendung des Doc2Vec Verfahren gegenüber dem Tf-idf bessere Ergebnisse erzielt. Durch die Verwendung von zusätzlichen Features während des Training des Doc2Vec Modells, konnte die Genauigkeit noch zusätzlich erhöht werden. So wurde mit der Berücksichtigung der hinterlegten Hashtags der Fragen über die drei Datensätze hinweg eine durchschnittliche Genauigkeit von 87.91% erreicht, was eine Steigerung von 4.83 Prozentpunkten ist gegenüber der Genauigkeit von 83.08% bei Verwendung von Tf-idf. Ebenfalls konnte in dieser Evaluierungsphase festgestellt werden, dass bei den verwendeten Daten für Doc2Vec eine kleine Vektorgröße von 25 ausreichend ist und bessere Ergebnisse erzielt gegenüber größeren zu trainierenden Vektoren. Durch die Verwendung von Doc2Vec wird somit nicht nur eine höhere Genauigkeit erzielt, sondern die Fragen können ebenfalls auf einen kleineren Vektorraum dargestellt werden gegenüber dem Tf-idf Verfahren.

Für die anschließende Evaluierung zur Erstellung der Benutzerprofile, wurde das Doc2Vec mit der Berücksichtigung der Hashtags verwendet, welche zuvor die beste Ergebnisse erzielt hat, sowie Tf-idf zum Vergleich. Dabei wurde festgestellt, dass die Verwendung des Maximum als Aggregierungsfunktion die schlechtesten Ergebnisse erzielt. Die Verwendung des gewichteten Mittelwert erzielt mit dem exponentiellen Zerfall ähnlich gute Ergebnisse, wie der harte Schnitt mit Einschränkung auf die zuletzt N beantworteten Fragen des Benutzers.

Bei dem Datensatz, welcher eine stärkere Benutzerhistorie abbildet gegenüber den anderen, macht sich die Differenz zwischen den beiden Verfahren stärker bemerkbar, wodurch mit dem exponentielle Zerfall bessere Ergebnisse erzielt wurden. Generell konnte mit der Verwendung der richtigen Wahl des exponentielle Zerfall und dem gewichteten Mittelwert bessere Ergebnisse erzielt werden gegenüber den normalen arithmetischen Mittelwert. Zu beachten ist hierbei allerdings, dass evtl. durch eine zu hohe Wahl des Zerfalls die Benutzerhistorie zu sehr eingeschränkt wird und das Benutzerprofil zu sehr auf die jüngsten Fragen ausgerichtet ist, was zu Problemen führen kann, wenn der Benutzer mehrere Interessen besitzt.

Der Effekt des Zerfalls zeigt am meisten Wirkung bei Benutzern, welche sich nach einer gewissen Zeit für ein neues Thema interessieren und das andere vernachlässigen. Durch die stärkere Gewichtung aktueller Fragen gegenüber älteren, wird sich schneller an das neue Interesse gerichtet, während der normale Mittelwert weiterhin alle alten Fragen gleich berücksichtigt und dementsprechend länger benötigt, sich an das neue Interesse auszurichten. Dies hat sich sehr stark bei einem Testbenutzer bemerkbar gemacht, welcher sein Interesse komplett geändert hat und mit der Anwendung des exponentiellen Zerfall und dem gewichteten Mittelwert eine Genauigkeit von 86.58% erzielt wurde gegenüber 72.48% bei Verwendung des normalen Mittelwert.

Abschließend wurde in der Evaluierung noch die Verwendung der externen Daten eines Benutzer betrachtet, inwiefern mit diesen eine Unterstützung bei den initialen Empfehlungen stattfindet. Diese liegen zwar nur für einen Teil der Benutzer vor und die Qualität der Webseite, welche als externe Daten des Benutzers extrahiert wurde, wurden nicht geprüft. Dennoch konnten im durchschnitt über die drei verschiedenen Datensätze bei 35.66% der Benutzer deren erste beantworte Fragen erfolgreich empfohlen werden.

Durch die Anwendung von Doc2Vec mit den hinterlegten Hashtags der Fragen als zusätzliche Features, während dessen Trainingsphase, konnten die Fragenprofile gegenüber den Tf-idf besser abgebildet werden, was eine höhere Genauigkeit bei den Empfehlungen erzielte und effizienter gespeichert werden, da diese einen sehr viel kleineren Vektorraum verwenden. Die Verwendung des gewichteten Mittelwert im Zusammenhang mit dem exponentiellen Zerfall kann Vorteile bringen aber ist Abhängig von den verwendeten Daten und der Stärke des Zerfalls. Bei der Verwendung sollten daher mit steigenden Anzahl an Daten in dem System auch regelmäßig die gewählten Parameter erneut evaluiert werden und gegebenenfalls angepasst werden. Auch die Verwendung der zusätzlichen externen Benutzerdaten hätte bei mindestens einen drittel der Benutzer einen positiven Effekt gehabt bei einer initialen Empfehlung von Fragen.

In Tabelle 6.1 wird die Genauigkeit auf die Testdaten für Tf-idf unter Verwendung des normalen arithmetischen Mittelwert und die optimierten Doc2Vec Werte angezeigt. Im Vergleich zum bekannten Tf-idf Ansatz, konnte mit den in dieser Arbeit vorgestellten Methoden eine Steigerung von bis zu 5.79 Prozentpunkten vorgenommen werden.

Datensatz	Tf-idf	Doc2Vec	Differenz
V1	0.8107	0.8686	0.0579
V2	0.8228	0.8622	0.0394
V3	0.8589	0.9126	0.0537

Tabelle 6.1: Genauigkeit der Tf-idf Variante mit dem normalen arithmetischen Mittelwert als Ausgangsbasis im Vergleich zum bester erzielten Genauigkeit von Doc2Vec mit Berücksichtigung der Hashtags

6.2 AUSBLICK

In einem nächsten Schritt sollte das aktuelle RS aktiv gesetzt werden in der bestehenden CQA-Plattform und als Online Experiment weiter beobachtet werden. Dadurch wird es möglich zu Evaluieren, wie gut dies auch in einem Echtzeitsystem funktioniert, anstatt auf die vorhandenen Offline Daten. Dadurch können auch neue vorteilhafte Daten entstehen, indem Benutzer bewerten können, ob eine Empfehlung des System ihrer Interesse entspricht

oder eine falsche Empfehlung ausgesprochen wurde. Diese vom Benutzer als schlecht empfundenen Empfehlungen sind sehr hilfreich, da diese als negative Beispiele zum Trainieren des Klassifizierers verwendet können. Gegenüber der aktuellen Methode, welche nur vermutet, dass sich ein Benutzer für eine zeitlich naheliegende Frage nicht interessiert hat, kann damit noch spezieller auf das Benutzerinteresse eingegangen werden und das System optimiert werden.

In dem Abschnitt 4.5 wurden einige weitere Methoden beschrieben, um bei der Erstellung der Trainingsdaten zusätzliche Manipulationen durchzuführen, welche aus zeitlichen Gründen in dieser Arbeit nicht mehr evaluiert werden konnten. Hierbei wäre zum einen interessant gewesen, wie sich die Ergebnisse auswirken, wenn die beantworten Fragen von Benutzern zum erstellen des Benutzerprofils zufällig sortiert gewesen wäre oder leicht angepasst Werte hätten. Dies würde dem Fall entsprechen, dass Benutzer ihre Fragen auch beantwortet hätten, wenn diese zu einem anderen Zeitpunkt gefragt wurde oder diese Frage auch in leicht abgewandelter Form beachtet hätten. Ebenfalls wäre in diesem Zusammenhang interessant zu prüfen, welche Auswirkung das zusätzliche Erstellen von künstlicher Trainingsdaten auf die Ergebnisse hat, was vor allem bei wenig vorhandenen Daten eine Rolle spielen kann, damit ausreichend Trainingsdaten verfügbar sind.

Auch die Verwendung von zusätzlichen Features (z.B. Qualität der Antwort, Aktivitätenlevel oder Motivation des Benutzers), welche in bekannte Ansätze verwendet werden, können sinnvoll sein. Diese spielen hauptsächlich eine Rolle bei der Weiterleitung von neuen Fragen an die Benutzer durch Push-Benachrichtigung, damit diese reagieren und weniger, wenn Benutzer aktiv im System angemeldet sind. Durch die Verwendung eines KNN können weitere Features einfach in die Architektur integriert werden. Dabei entsteht die Variante, dass die zusätzlichen Features in das Trainings integriert werden und ein komplettes Modell entsteht oder, dass das aktuelle Modell unabhängig bleibt für die Ermittlung von Benutzerinteresse und die Ausgaben dieses trainierte Modell dann als zusätzliches Feature in neuen Modellen verwendet wird.

In dieser Arbeit konnte gezeigt werden, wie eine CQA-Plattform erweitert werden kann um ein RS, welches dort gestellte Fragen und Benutzer zusammenbringen soll. Dies beinhaltet eine effektive Darstellung der Fragen, sowie die Ermittlung von Benutzerinteressen und das abschließende Zusammenbringen dieser als Empfehlungen. Dadurch konnte eine solide Ausgangsbasis für ein RS innerhalb der Plattform implementiert werden und bietet weitere Möglichkeiten zur Erweiterung.

Teil II

APPENDIX

A

VERWENDETE SOFTWARE

Die Implementierung und Evaluierung wurde mit der Programmiersprache Python 3.6.6 durchgeführt. In der unten angezeigten Tabelle werden die verwendete Python Module mit Version aufgelistet.

Modul	**Version**	**Webseite**
Python	3.6.6	https://www.python.org
Numpy	1.17.0	https://numpy.org
Scipy	1.3.0	https://scipy.org
Pandas	0.25.0	https://pandas.pydata.org
Scikit-Learn	0.20.3	https://scikit-learn.org
Spacy	2.1.6	https://spacy.io
Gensim	3.8.0	https://radimrehurek.com/gensim
Tensorflow	1.12.0	https://www.tensorflow.org
Keras	2.2.4	https://keras.io
Scrapy	1.6.0	https://scrapy.org
Flask	1.0.2	http://flask.pocoo.org

Tabelle A.1: Übersicht der verwendeten Python Module

B

BEILIEGENDE DATEIEN

In diesem Kapitel wird die Struktur des beiliegenden Python-Projekt erläutert, welches für diese Arbeit zur Evaluierung implementiert wurde.

```
CQA/ ..................................Python Projekt Ordner
├── cqa/cqa.api .......................Verbindung zur MongoDB
├── data/ .............................Verwendete Datensätze
├── logs/ .............................Logdateien
├── notebooks/ ........................Jupyter Notebook Vorlagen
├── recommender/
│   ├── model/ ........................Speicherpfad für trainierter
│   │                                  Modelle
│   ├── engine.py .....................Recommender Klasse für
│   │                                  Vorhersagen
│   ├── keras_models.py ...............Templates für NN Architektur
│   │                                  und Evaluierungsmethode
│   ├── preprocessing.py ..............Methoden zur Vorverarbeitung
│   │                                  von Texten
│   ├── trainer.py ....................Benötigten Trainingsmethoden
│   │                                  zum Erstellen der Modelle
│   └── vectorizer.py .................Verschiedene Klassen zum
│                                      Transformieren von Text in
│                                      Vektoren
├── requirements.txt ..................Benötigte Python Module
└── setup.py ..........................Globale Parametereinstellungen
```

C

DETAILLIERTE ERGEBNISSE DER EVALUIERUNG

In diesem Kapitel werden die detaillierte Ergebnisse aus der Evaluierung aufgelistet, welche in dem Kapitel 5 aus Übersichtsgründen nur als Teilausschnitt oder zusammengefasst verwendet wurden. Die Ergebnisse zeigen die Genauigkeit während der Trainings-, Evaluierungs- und Testphase unter der Verwendung von verschiedenen Parametereinstellungen und den drei Datensätzen.

In den Tabellen C.1, C.2 und C.3 wurde jeweils für die verschiedene Datensätze eine Tabelle erstellt, welche die initiale Betrachtung der Verwendung von Tf-idf mit der gewünschten Vorgabe, wie groß das Vokabular maximal sein sollte (*max_features*), sowie die Filterung auf die Part-of-Speech (POS)-Tags. Zur Berechnung der Benutzerprofile wurde hierbei der arithmetische Mittelwert verwendet, da diese Evaluierung sich auf die Darstellung der Frageprofile konzentrieren sollte. In den Tabellen C.4, C.5, C.6, C.7 und C.8 wurde das gleiche Verfahren angewandt aber unter der Betrachtung von Doc2Vec und seinen verschiedenen Varianten. Anstatt der maximalen Vokabulargröße wurden allerdings bei Doc2Vec dessen verschiedenen Parameter Vektorgröße (*vector_size*) und die Anzahl der benachbarten Wörter (*window*) betrachtet. In den drei letzten Tabellen C.9, C.10 und C.11 findet die Betrachtung der Berechnungslogiken für das Benutzerprofil unter der Anwendung eines verschieden starken Zerfalls älterer Fragen.

max_features	POS-Filter	Vektorgröße	Training	Validierung	Test
None		10029	0.7921	0.8000	0.8107
5000		5000	0.7911	0.8003	0.8025
2500		2500	0.7884	0.7939	0.7916
1000		1000	0.7805	0.7854	0.7728
500		500	0.7671	0.7706	0.7575
250		250	0.7442	0.7531	0.7367
None	x	9409	0.7880	0.7971	0.8034
5000	x	5000	0.7881	0.7953	0.7986
2500	x	2500	0.7848	0.7958	0.7878
1000	x	1000	0.7770	0.7857	0.7684
500	x	500	0.7601	0.7634	0.7493
250	x	250	0.7424	0.7469	0.736

Tabelle C.1: Tf-idf Ergebnisse für Datensatz V1

max_features	POS-Filter	Vektorgröße	Training	Validierung	Test
None		3589	0.7495	0.7916	0.8228
2500		2500	0.7482	0.7916	0.8189
1000		1000	0.7515	0.7738	0.7717
500		500	0.7441	0.7649	0.7283
250		250	0.7325	0.7381	0.689
None	x	3327	0.7495	0.7887	0.8189
2500	x	2500	0.7459	0.7976	0.8189
1000	x	1000	0.7518	0.7857	0.7638
500	x	500	0.7376	0.7679	0.7362
250	x	250	0.7209	0.7173	0.7205

Tabelle C.2: Tf-idf Ergebnisse für Datensatz V2

max_features	POS-Filter	Vektorgröße	Training	Validierung	Test
None		8777	0.8497	0.8506	0.8589
5000		5000	0.8518	0.8493	0.8623
2500		2500	0.8532	0.8513	0.8642
1000		1000	0.8511	0.8506	0.8642
500		500	0.8354	0.8437	0.8498
250		250	0.8233	0.8179	0.8246
None	x	8199	0.8449	0.8470	0.8571
5000	x	5000	0.8481	0.8444	0.8590
2500	x	2500	0.8490	0.8493	0.8616
1000	x	1000	0.8449	0.8476	0.8621
500	x	500	0.8335	0.8429	0.8505
250	x	250	0.8158	0.8130	0.8247

Tabelle C.3: Tf-idf Ergebnisse für Datensatz V3

Datensatz	vector_size	window	Training	Validierung	Test
V1	25	5	0.7917	0.8051	0.8022
V1	50	5	0.7625	0.7770	0.7848
V1	75	5	0.7439	0.7539	0.7716
V1	25	10	0.7702	0.7801	0.7810
V1	50	10	0.7393	0.7464	0.7643
V1	75	10	0.7179	0.7297	0.7428
V2	25	5	0.7433	0.7619	0.8071
V2	50	5	0.7137	0.7292	0.7835
V2	75	5	0.6936	0.7232	0.7874
V2	25	10	0.7064	0.7738	0.8150
V2	50	10	0.6840	0.7321	0.7402
V2	75	10	0.6724	0.7292	0.7559
V3	25	5	0.8533	0.8638	0.8685
V3	50	5	0.8400	0.8420	0.8586
V3	75	5	0.8267	0.8306	0.8464
V3	25	10	0.8265	0.8336	0.8440
V3	50	10	0.8085	0.8170	0.8304
V3	75	10	0.7924	0.8010	0.8147

Tabelle C.4: Ergebnisse für Doc2Vec Standardansatz

Datensatz	vector_size	window	Training	Validierung	Test
V1	25	5	0.8557	0.8540	0.8645
V1	50	5	0.8346	0.8399	0.8539
V1	75	5	0.8100	0.8170	0.8292
V1	25	10	0.8420	0.8476	0.8542
V1	50	10	0.8115	0.8173	0.8275
V1	75	10	0.7865	0.7918	0.8048
V2	25	5	0.8284	0.8274	0.8622
V2	50	5	0.8000	0.8185	0.8858
V2	75	5	0.7719	0.7917	0.8740
V2	25	10	0.8044	0.8244	0.8504
V2	50	10	0.7696	0.7708	0.8661
V2	75	10	0.7631	0.8006	0.8661
V3	25	5	0.9052	0.9082	0.9106
V3	50	5	0.8890	0.8880	0.8944
V3	75	5	0.8758	0.8769	0.8800
V3	25	10	0.8910	0.8866	0.8959
V3	50	10	0.8722	0.8763	0.8784
V3	75	10	0.8567	0.8522	0.8606

Tabelle C.5: Ergebnisse für Doc2Vec mit Hashtags als zusätzlicher Trainingsvektor

Datensatz	vector_size	window	Training	Validierung	Test
V1	25	5	0.8557	0.8540	0.8645
V1	50	5	0.8346	0.8399	0.8539
V1	75	5	0.8100	0.8170	0.8292
V1	25	10	0.8420	0.8476	0.8542
V1	50	10	0.8115	0.8173	0.8275
V1	75	10	0.7865	0.7918	0.8048
V2	25	5	0.8284	0.8274	0.8622
V2	50	5	0.8000	0.8185	0.8858
V2	75	5	0.7719	0.7917	0.8740
V2	25	10	0.8044	0.8244	0.8504
V2	50	10	0.7696	0.7708	0.8661
V2	75	10	0.7631	0.8006	0.8661
V3	25	5	0.9052	0.9082	0.9106
V3	50	5	0.8890	0.8880	0.8944
V3	75	5	0.8758	0.8769	0.8800
V3	25	10	0.8910	0.8866	0.8959
V3	50	10	0.8722	0.8763	0.8784
V3	75	10	0.8567	0.8522	0.8606

Tabelle C.6: Ergebnisse für Doc2Vec mit Beantworter als zusätzlicher Trainingsvektor

Datensatz	vector_size	window	Training	Validierung	Test
V1	25	5	0.8430	0.8550	0.8524
V1	50	5	0.8322	0.8486	0.8551
V1	75	5	0.8277	0.8391	0.8586
V1	25	10	0.8289	0.8391	0.8322
V1	50	10	0.8180	0.8266	0.8354
V1	75	10	0.8106	0.8221	0.8313
V2	25	5	0.8131	0.8571	0.8700
V2	50	5	0.8258	0.8542	0.8858
V2	75	5	0.8286	0.8542	0.9016
V2	25	10	0.8041	0.8244	0.8583
V2	50	10	0.7889	0.8333	0.8740
V2	75	10	0.7814	0.8125	0.8504
V3	25	5	0.9050	0.9054	0.9135
V3	50	5	0.8845	0.8785	0.8912
V3	75	5	0.8684	0.8757	0.8830
V3	25	10	0.8865	0.8922	0.8988
V3	50	10	0.8585	0.8582	0.8737
V3	75	10	0.8220	0.8177	0.8341

Tabelle C.7: Ergebnisse für Doc2Vec mit Hashtags und Beantworter als zusätzlicher Trainingsvektor

Datensatz	vector_size	window	Training	Validierung	Test
V1	25	5	0.8374	0.8383	0.8442
V1	50	5	0.8138	0.8176	0.8245
V1	75	5	0.7878	0.7894	0.8000
V1	25	10	0.8173	0.8311	0.8330
V1	50	10	0.7886	0.7873	0.8042
V1	75	10	0.7624	0.7695	0.7866
V2	25	5	0.8142	0.7887	0.8228
V2	50	5	0.7900	0.7768	0.8543
V2	75	5	0.7567	0.7679	0.8307
V2	25	10	0.7869	0.7738	0.8268
V2	50	10	0.7343	0.7530	0.7874
V2	75	10	0.7309	0.7530	0.8189
V3	25	5	0.8811	0.8901	0.8910
V3	50	5	0.8700	0.8748	0.8850
V3	75	5	0.8521	0.8560	0.8677
V3	25	10	0.8710	0.8722	0.8804
V3	50	10	0.8502	0.8530	0.8674
V3	75	10	0.8254	0.8248	0.8409

Tabelle C.8: Ergebnisse für Doc2Vec mit Hashtags als zusätzlicher Trainingsvektor und POS-Filter

		Tf-idf			Doc2Vec		
λ	Methode	Training	Validierung	Test	Training	Validierung	Test
0.25	Mean	0.7940	0.8030	0.8039	0.8549	0.8537	0.8674
0.10	Mean	0.7928	0.8035	0.8013	0.8554	0.8550	0.8686
0.00	Mean	0.7911	0.8003	0.8027	0.8557	0.8540	0.8645
0.01	Mean	0.7914	0.8003	0.8016	0.8557	0.8542	0.8654
0.025	Mean	0.7913	0.8016	0.8016	0.8555	0.855	0.8663
0.05	Mean	0.7916	0.8011	0.8010	0.8559	0.8548	0.8668
0.01	Max	0.7770	0.7710	0.7890	0.7866	0.7751	0.8004
0.025	Max	0.7780	0.7710	0.7870	0.7864	0.7764	0.7998
0.05	Max	0.7800	0.7750	0.7890	0.7877	0.7764	0.8007
0.10	Max	0.7803	0.7793	0.7913	0.7867	0.7783	0.8019
0.25	Max	0.7821	0.7884	0.7975	0.7820	0.7785	0.7998
0.00	Max	0.7780	0.7710	0.7890	0.7756	0.7711	0.7960
0.00	Letzten 5	0.7940	0.8000	0.7992	0.8551	0.8521	0.8657
0.00	Letzten 10	0.7939	0.8062	0.8054	0.8561	0.8537	0.8663
0.00	Letzten 20	0.7916	0.8011	0.8028	0.8559	0.8555	0.8654

Tabelle C.9: Ergebnisse für verschiedene Berechnungsmethoden des Benutzerprofils für Datensatz V1

		Tf-idf			Doc2Vec		
λ	Methode	Training	Validierung	Test	Training	Validierung	Test
0.25	Mean	0.7513	0.7768	0.8031	0.8310	0.8214	0.8622
0.10	Mean	0.7523	0.7800	0.8189	0.8244	0.8244	0.8622
0.00	Mean	0.7495	0.7917	0.8228	0.8284	0.8274	0.8622
0.01	Mean	0.7495	0.7917	0.8228	0.8289	0.8274	0.8583
0.025	Mean	0.7492	0.7917	0.8228	0.8284	0.8274	0.8622
0.05	Mean	0.7495	0.7857	0.8228	0.8281	0.8304	0.8622
0.01	Max	0.7425	0.7738	0.8110	0.7657	0.8006	0.7953
0.025	Max	0.7479	0.7738	0.7992	0.7644	0.8006	0.7953
0.05	Max	0.7487	0.7738	0.8031	0.7652	0.8006	0.7874
0.10	Max	0.7459	0.7800	0.7992	0.7665	0.8006	0.7913
0.25	Max	0.7464	0.7857	0.8150	0.7539	0.7887	0.7756
0.00	Max	0.7392	0.7738	0.8110	0.7652	0.8006	0.7953
0.00	Letzten 5	0.7528	0.7798	0.8150	0.8312	0.8214	0.8701
0.00	Letzten 10	0.7510	0.7887	0.8228	0.8302	0.8830	0.8622
0.00	Letzten 20	0.7495	0.7917	0.8228	0.8284	0.8274	0.8622

Tabelle C.10: Ergebnisse für verschiedene Berechnungsmethoden des Benutzerprofils für Datensatz V2

		Tf-idf			Doc2Vec		
λ	Methode	Training	Validierung	Test	Training	Validierung	Test
0.25	Mean	0.8360	0.8375	0.8401	0.8911	0.8981	0.9001
0.10	Mean	0.8493	0.8500	0.8560	0.9017	0.9032	0.9061
0.00	Mean	0.8519	0.8493	0.8623	0.9052	0.9082	0.9106
0.01	Mean	0.8546	0.8493	0.8622	0.9079	0.9078	0.9126
0.025	Mean	0.8550	0.8530	0.8626	0.9072	0.9084	0.9114
0.05	Mean	0.8545	0.8545	0.8623	0.9054	0.9080	0.9111
0.01	Max	0.8181	0.8188	0.8286	0.7023	0.6987	0.6921
0.025	Max	0.8197	0.8207	0.8295	0.7086	0.7065	0.7003
0.05	Max	0.8215	0.8216	0.8297	0.7159	0.7149	0.7105
0.10	Max	0.8200	0.8222	0.8263	0.7209	0.7224	0.7144
0.25	Max	0.8105	0.8134	0.8190	0.7309	0.7284	0.7312
0.00	Max	0.8142	0.8157	0.8199	0.6856	0.6791	0.6640
0.00	Letzten 5	0.8199	0.8274	0.8280	0.8893	0.8950	0.8930
0.00	Letzten 10	0.8429	0.8425	0.8498	0.8978	0.9007	0.9019
0.00	Letzten 20	0.8521	0.8511	0.8591	0.9029	0.9047	0.9070

Tabelle C.11: Ergebnisse für verschiedene Berechnungsmethoden des Benutzerprofils für Datensatz V3

[Agg18] Charu C. Aggarwal. "An Introduction to Neural Networks". In: *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018, S. 1–52. ISBN: 978-3-319-94463-0. DOI: 10.1007/978-3-319-94463-0_1. URL: https://doi.org/10.1007/978-3-319-94463-0_1.

[Ben+03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent und Christian Janvin. "A Neural Probabilistic Language Model". In: *J. Mach. Learn. Res.* 3 (März 2003), S. 1137–1155. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944966.

[BDW08] Mohamed Bouguessa, Benoundefinedt Dumoulin und Shengrui Wang. "Identifying Authoritative Actors in Question-Answering Forums: The Case of Yahoo! Answers". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: Association for Computing Machinery, 2008, S. 866–874. ISBN: 9781605581934. DOI: 10.1145/1401890.1401994. URL: https://doi.org/10.1145/1401890.1401994.

[BMA16] Georgios-Ioannis Brokos, Prodromos Malakasiotis und Ion Androutsopoulos. "Using Centroids of Word Embeddings and Word Mover's Distance for Biomedical Document Retrieval in Question Answering". In: *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, S. 114–118. DOI: 10.18653/v1/W16-2915. URL: https://www.aclweb.org/anthology/W16-2915.

[Che+14] Tianjian Chen, Jing Cai, Hao Wang und Yu Dong. "Instant Expert Hunting: Building an Answerer Recommender System for a Large Scale QA Website". In: *Proceedings of the 29th Annual ACM Symposium on Applied Computing*. SAC '14. Gyeongju, Republic of Korea: ACM, 2014, S. 260–265. ISBN: 978-1-4503-2469-4. DOI: 10.1145/2554850.2554915. URL: http://doi.acm.org/10.1145/2554850.2554915.

[Cho+15] Jason H. D. Cho, Yanen Li, Roxana Girju und Chengxiang Zhai. "Recommending Forum Posts to Designated Experts". In: *Proceedings of the 2015 IEEE International Conference on Big Data (Big Data)*. BIG DATA '15. USA: IEEE Computer Society, 2015, S. 659–666. ISBN: 9781479999262. DOI: 10.1109/BigData.2015.7363810. URL: https://doi.org/10.1109/BigData.2015.7363810.

[CJMS17] Edilson Anselmo Corrêa Júnior, Vanessa Queiroz Marinho und Leandro Borges dos Santos. "NILC-USP at SemEval-2017 Task 4: A Multi-view Ensemble for Twitter Sentiment Analysis". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017, S. 611–615. DOI: 10.18653/v1/S17-2100. URL: https://www.aclweb.org/anthology/S17-2100.

[DL05] Yi Ding und Xue Li. "Time Weight Collaborative Filtering". In: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. CIKM '05. Bremen, Germany: ACM, 2005, S. 485–492. ISBN: 1-59593-140-6. DOI: 10.1145/1099554.1099689. URL: http://doi.acm.org/10.1145/1099554.1099689.

[Don+15] H. Dong, J. Wang, H. Lin, B. Xu und Z. Yang. "Predicting Best Answerers for New Questions: An Approach Leveraging Distributed Representations of Words in Community Question Answering". In: *2015 Ninth International Conference on Frontier of Computer Science and Technology*. 2015, S. 13–18. DOI: 10.1109/FCST.2015.56.

[Dro+11] Gideon Dror, Yehuda Koren, Yoelle Maarek und Idan Szpektor. "I Want to Answer; Who Has a Question?: Yahoo! Answers Recommender System". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '11. San Diego, California, USA: ACM, 2011, S. 1109–1117. ISBN: 978-1-4503-0813-7. DOI: 10.1145/2020408.2020582. URL: http://doi.acm.org/10.1145/2020408.2020582.

[GPJ18] Palash Goyal, Sumit Pandey und Karan Jain. "Word Vector Representations". In: *Deep Learning for Natural Language Processing: Creating Neural Networks with Python*. Berkeley, CA: Apress, 2018, S. 75–118. ISBN: 978-1-4842-3685-7. DOI: 10.1007/978-1-4842-3685-7_2. URL: https://doi.org/10.1007/978-1-4842-3685-7_2.

[Jam+13a] Gareth James, Daniela Witten, Trevor Hastie und Robert Tibshirani. "Classification". In: *An Introduction to Statistical Learning: with Applications in R*. New York, NY: Springer New York, 2013, S. 127–173. ISBN: 978-1-4614-7138-7. DOI: 10.1007/978-1-4614-7138-7_4. URL: https://doi.org/10.1007/978-1-4614-7138-7_4.

[Jam+13b] Gareth James, Daniela Witten, Trevor Hastie und Robert Tibshirani. "Support Vector Machines". In: *An Introduction to Statistical Learning: with Applications in R*. New York, NY: Springer New York, 2013, S. 337–372. ISBN: 978-1-4614-7138-7. DOI: 10.1007/978-1-4614-7138-7_9. URL: https://doi.org/10.1007/978-1-4614-7138-7_9.

[JM09] Dan Jurafsky und James H. Martin. *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition, 2nd Edition*. Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. ISBN: 9780135041963. URL: http://www.worldcat.org/oclc/315913020.

[JA07] Pawel Jurczyk und Eugene Agichtein. "Discovering Authorities in Question Answer Communities by Using Link Analysis". In: *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*. CIKM '07. Lisbon, Portugal: Association for Computing Machinery, 2007, S. 919–922. ISBN: 9781595938039. DOI: 10.1145/1321440.1321575. URL: https://doi.org/10.1145/1321440.1321575.

[Kam+18] Ms Kamble, Rajani Ravaso, Mr Chetan, Chetan Awati und Sonam Kharade. "Review Paper on Answers Selection and Recommendation in Community Question Answers System". In: *International Journal on Future Revolution in Computer Science and Communication Engineering (IJFRSCE)* 4.1 (2018), S. 298–302.

[Kub15] Miroslav Kubat. "Performance Evaluation". In: *An Introduction to Machine Learning*. Cham: Springer International Publishing, 2015, S. 213–233. ISBN: 978-3-319-20010-1. DOI: 10.1007/978-3-319-20010-1_11. URL: https://doi.org/10.1007/978-3-319-20010-1_11.

[LM14] Quoc V. Le und Tomas Mikolov. "Distributed Representations of Sentences and Documents". In: *CoRR* abs/1405.4053 (2014). arXiv: 1405.4053. URL: http://arxiv.org/abs/1405.4053.

[LG14] Omer Levy und Yoav Goldberg. "Dependency-Based Word Embeddings". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Juni 2014, S. 302–308. DOI: 10.3115/v1/P14-2050. URL: https://www.aclweb.org/anthology/P14-2050.

[Li+14] Lei Li, Li Zheng, Fan Yang und Tao Li. "Modeling and Broadening Temporal User Interest in Personalized News Recommendation". In: *Expert Syst. Appl.* 41.7 (Juni 2014), S. 3168–3177. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2013.11.020. URL: http://dx.doi.org/10.1016/j.eswa.2013.11.020.

[LLY10] Mingrong Liu, Yicen Liu und Qing Yang. "Predicting Best Answerers for New Questions in Community Question Answering". In: *Web-Age Information Management*. Hrsg. von Lei Chen, Changjie Tang, Jun Yang und Yunjun Gao. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 127–138. ISBN: 978-3-642-14246-8.

[LD13] Rushi Longadge und Snehalata Dongre. "Class Imbalance Problem in Data Mining Review". In: *CoRR* abs/1305.1707 (2013). arXiv: 1305.1707. URL: http://arxiv.org/abs/1305.1707.

[LS68] H.P. Luhn und C.K. Schultz. *Pioneer of Information Science: Selected Works*. London: Spartan Books, 1968.

[Luo+14] Lin Luo, Fei Wang, Michelle X. Zhou, Yingxin Pan und Hang Chen. "Who Have Got Answers?: Growing the Pool of Answerers in a Smart Enterprise Social QA System". In: *Proceedings of the 19th International Conference on Intelligent User Interfaces*. IUI '14. Haifa, Israel: ACM, 2014, S. 7–16. ISBN: 978-1-4503-2184-6. DOI: 10.1145/2557500.2557531. URL: http://doi.acm.org/10.1145/2557500.2557531.

[Mac+17] Jakub Macina, Ivan Srba, Joseph Jay Williams und Maria Bielikova. "Educational Question Routing in Online Student Communities". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys '17. Como, Italy: ACM, 2017, S. 47–55. ISBN: 978-1-4503-4652-8. DOI: 10.1145/3109859.3109886. URL: http://doi.acm.org/10.1145/3109859.3109886.

[MRS08] Christopher D. Manning, Prabhakar Raghavan und Hinrich Schütze. *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008. ISBN: 0521865719, 9780521865715.

[Mel+16] Oren Melamud, David McClosky, Siddharth Patwardhan und Mohit Bansal. "The Role of Context Types and Dimensionality in Learning Word Embeddings". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, Juni 2016, S. 1030–1040. DOI: 10.18653/v1/N16-1118. URL: https://www.aclweb.org/anthology/N16-1118.

[Mik+13a] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado und Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality". In: *CoRR* abs/1310.4546 (2013). arXiv: 1310.4546. URL: http://arxiv.org/abs/1310.4546.

[Mik+13b] Tomas Mikolov, G.s Corrado, Kai Chen und Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: Jan. 2013, S. 1–12.

[NM12] Ani Nenkova und Kathleen McKeown. "A Survey of Text Summarization Techniques". In: *Mining Text Data*. Hrsg. von Charu C. Aggarwal und ChengXiang Zhai. Boston, MA: Springer US, 2012, S. 43–76. ISBN: 978-1-4614-3223-4. DOI: 10.1007/978-1-4614-3223-4_3. URL: https://doi.org/10.1007/978-1-4614-3223-4_3.

[Por80] Martin F Porter. "An algorithm for suffix stripping". In: *Program* 14.3 (1980), S. 130–137.

[Ria+12] Fatemeh Riahi, Zainab Zolaktaf, Mahdi Shafiei und Evangelos Milios. "Finding Expert Users in Community Question Answering". In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12 Companion. Lyon, France: ACM, 2012, S. 791–798. ISBN: 978-1-4503-1230-1. DOI: 10.1145/2187980.2188202. URL: http://doi.acm.org/10.1145/2187980.2188202.

[SW17] Claude Sammut und Geoffrey I. Webb, Hrsg. *Encyclopedia of Machine Learning and Data Mining*. Springer, 2017. ISBN: 978-1-4899-7685-7. DOI: 10.1007/978-1-4899-7687-1. URL: https://doi.org/10.1007/978-1-4899-7687-1.

[SPK14] Jose San Pedro und Alexandros Karatzoglou. "Question Recommendation for Collaborative Question Answering Systems with RankSLDA". In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys '14. Silicon Valley, California, USA: ACM, 2014, S. 193–200. ISBN: 978-1-4503-2668-1. DOI: 10.1145/2645710.2645736. URL: http://doi.acm.org/10.1145/2645710.2645736.

[Sar16] Dipanjan Sarkar. "Processing and Understanding Text". In: *Text Analytics with Python: A Practical Real-World Approach to Gaining Actionable Insights from your Data*. Berkeley, CA: Apress, 2016, S. 107–165. ISBN: 978-1-4842-2388-8. DOI: 10.1007/978-1-4842-2388-8_3. URL: https://doi.org/10.1007/978-1-4842-2388-8_3.

[Shp19] Gidi Shperber. *A gentle introduction to Doc2Vec*. Shibumi. 7. Okt. 2019. URL: https://www.shibumi-ai.com/post/a-gentle-introduction-to-doc2vec (besucht am 20. 10. 2019).

[SGB15] Ivan Srba, Marek Grznar und Maria Bielikova. "Utilizing Non-QA Data to Improve Questions Routing for Users with Low QA Activity in CQA". In: *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*. ASONAM '15. Paris, France: ACM, 2015, S. 129–136. ISBN: 978-1-4503-3854-7. DOI: 10.1145/2808797.2809331. URL: http://doi.acm.org/10.1145/2808797.2809331.

[Sri+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), S. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[SMP13] Idan Szpektor, Yoelle Maarek und Dan Pelleg. "When Relevance is Not Enough: Promoting Diversity and Freshness in Personalized Question Recommendation". In: *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13. Rio de Janeiro, Brazil: ACM, 2013, S. 1249–1260. ISBN: 978-1-4503-2035-1. DOI: 10.1145/2488388.2488497. URL: http://doi.acm.org/10.1145/2488388.2488497.

[Tia+14] Yuan Tian, Pavneet Singh Kochhar, Ee-Peng Lim, Feida Zhu und David Lo. "Predicting Best Answerers for New Questions: An Approach Leveraging Topic Modeling and Collaborative Voting". In: *Social Informatics*. Hrsg. von Akiyo Nadamoto, Adam Jatowt, Adam Wierzbicki und Jochen L. Leidner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, S. 55–68. ISBN: 978-3-642-55285-4.

[Wik19] Wikipedia contributors. *Weighted arithmetic mean*. 2019. URL: https://en.wikipedia.org/wiki/Weighted_arithmetic_mean (besucht am 02. 10. 2019).

[YM14] Baoguo Yang und Suresh Manandhar. "Tag-Based Expert Recommendation in Community Question Answering". In: *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ASONAM '14. Beijing, China: IEEE Press, 2014, S. 960–963. ISBN: 9781479958764.

[ZAA07] Jun Zhang, Mark S. Ackerman und Lada Adamic. "Expertise Networks in Online Communities: Structure and Algorithms". In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: Association for Computing Machinery, 2007, S. 221–230. ISBN: 9781595936547. DOI: 10.1145/1242572.1242603. URL: https://doi.org/10.1145/1242572.1242603.

[ZLK12] Tom Chao Zhou, Michael R. Lyu und Irwin King. "A Classification-based Approach to Question Routing in Community Question Answering". In: *Proceedings of the 21st International Conference on World Wide Web*. WWW '12 Companion. Lyon, France: ACM, 2012, S. 783–790. ISBN: 978-1-4503-1230-1. DOI: 10.1145/2187980.2188201. URL: http://doi.acm.org/10.1145/2187980.2188201.