

STACKOVERFLOW DUPLICATE DETECTION USING EMBEDDINGS

Bachelorseminar im Sommersemester 2020

Universität zu Köln

Wirtschafts- und Sozialwissenschaftliche Fakultät

Köln, den 1. Juli 2020

Schlicht, Nicolas

7324997

Wirtschaftsinformatik

Bachelorseminar Information Management (Prof. Schoder)

Sven Stahlmann, Johannes Mehlbach

Inhaltsverzeichnis

Inhaltsverzeichnis	I
Abbildungsverzeichnis.....	III
Tabellenverzeichnis	IV
Abkürzungsverzeichnis.....	V
1 Einleitung.....	1
1.1 Motivation	1
1.2 Ziele	2
1.3 Aufbau der Arbeit.....	3
2 Theoretischer Hintergrund.....	3
2.1 Begriffserklärung.....	4
2.1.1 Word/Document Embeddings.....	4
2.1.2 Paragraph Vector.....	4
2.1.3 Paragraph Vector-Distributed Memory	4
2.1.4 Doc2Vec.....	4
2.1.5 K-Means.....	5
2.2 Stand der Forschung	5
3 Methodik.....	6
3.1 Wahl der Methodik.....	6
3.2 Vorgehensweise.....	7
3.2.1 Art der Implementation.....	7
3.2.2 Aufbau und Aufbereitung des Datensatzes.....	8
4 Ergebnisse.....	10
4.1 Code.....	10
4.2 Beantwortung der Forschungsfragen.....	11
4.2.1 Ist eine semantische Gruppierung der eingelesenen Fragen erkennbar?	11
4.2.2 Können neue Fragen durch die akzeptierte Antwort der nächsten Frage im Vektorraum zufriedenstellend beantwortet werden?.....	13
5 Diskussion.....	16
5.1 Reflexion der Ergebnisse.....	17
5.2 Kontext zur Forschung	18
5.3 Limitationen des Projekts	18

6	Fazit	19
	Literaturverzeichnis	21
A.	Anhang – Ausschlussliste Wörterreduzierung.....	24
B.	Anhang – Themengenerierung Cluster (Dim. 24)	24
C.	Anhang – Themengenerierung Cluster (Dim. 300)	25
D.	Anhang – Ergebnis Rückwärtssuche (Dim. 24).....	26
E.	Anhang – Ergebnis Rückwärtssuche (Dim. 300).....	26
	Erklärung	27

Abbildungsverzeichnis

Abbildung 1: Cluster (Dim. 24).....	10
Abbildung 2: Cluster (Dim. 300).....	10
Abbildung 3: Cluster (Dim. 1000).....	10

Tabellenverzeichnis

Tabelle 1: Parameter Doc2Vec	8
Tabelle 2: Clusterbildung bei verschiedenen Vektordimensionen	10
Tabelle 3: Themen der Cluster (Dim. 24).....	11
Tabelle 4: Themen der Cluster (Dim. 300).....	12
Tabelle 5: Vorwärtssuche Test-Set (Dim. 24)	14
Tabelle 6: Duplikate für Rückwärtssuche.....	15
Tabelle 7: gefundenes Duplikat Rückwärtssuche (Dim. 300)	16

Abkürzungsverzeichnis

PV-DM	Paragraph Vector-Distributed Memory
BOW	Bag-of-Words
Q&A	Question-and-Answer
PCQA	Programming Community-Based Question-Answering

1 Einleitung

Diese Arbeit ist eine Machbarkeitsstudie, daher legt sie bei entsprechendem Ergebnis nur den Grundstein für weitere Forschungsprojekte. Sie wird im Rahmen des Bachelorseminars am Lehrstuhl Prof. Schoder geschrieben und befasst sich mit der Berechnung von semantischer Nähe zwischen Fragen, die auf der Plattform *StackOverflow* gestellt wurden.

1.1 Motivation

Question-and-Answer (Q&A) Plattformen bilden einen immer wichtiger werdenden Teil der Wissensbildung bei Entwicklern, sie dienen als soziales Netzwerk, in dem sich ausgetauscht wird und bei Fragen und Problemen einander geholfen wird (Storey et al., 2010, S. 362). Dies spiegelt sich auch in der jährlichen Entwicklerumfrage von *StackOverflow* wider, in der 96,9% aller Befragten angaben, dass sie *StackOverflow* nutzen um Antworten zu spezifischen Fragen zu finden (Stack Exchange Inc., 2019). Treude et al. (2011, S. 804) fanden zudem heraus, dass How-To-Fragen am frequentesten gestellt werden.

Q&A Plattformen wie *StackOverflow* lassen sich sogenannten *programming community-based question-answering (PCQA)* Plattformen zuordnen (W. E. Zhang et al., 2017, S. 1221). Besonders wichtig für PCQA-Plattformen ist die Partizipation der Benutzer, sodass der Community-geführte Dialog aus Fragestellern und beantwortenden Benutzern so frustrationsfrei wie möglich funktioniert. Die Plattform baut auf dem Wissen ihrer Benutzer und aus genau diesen Dialogen auf, da sonst kein neues Wissen der Plattform hinzugefügt würde und die Relevanz (für neue) Fachgebiete generell abnehmen würde. Deswegen ist es wichtig die Benutzer auf der Plattform zu halten. Neben Incentives wie einem Belohnungssystem, welches die quantifizierbare Reputation des Benutzers auf der Plattform beeinflusst (P. Berger et al., 2016, S. 644), ist es ebenso relevant Zeit und Aufwand der Benutzer zu sparen, indem Fragenduplikate vermieden werden (Bian et al., 2008). Durch einen effizienten Umgang mit Fragenduplikaten wird die Benutzererfahrung insgesamt verbessert (W. E. Zhang et al., 2017, S. 1221).

Die Betreiber von *StackOverflow* haben Richtlinien für den Umgang mit Duplikaten aufgestellt. Benutzer sind angehalten, das Forum erst nach Fragen zu durchsuchen, die deren eigene Frage beantworten könnten. Dies stellt sich in der

Praxis aber als schwer heraus, da Fragen auf mehrere Arten und Weisen gestellt werden können. Die Betreiber unterscheiden grundsätzlich zwischen drei verschiedenen Duplikat-Arten:

1. Copy-and-Paste Duplikate
2. unbeabsichtigte Duplikate
3. grenzwertige Duplikate

Bisher wird die Entfernung oder Markierung von Duplikaten durch Benutzer und Moderatoren gehandhabt (Jeff Atwood, 2009). Duplikate der Kategorie 1 sind normalerweise schnell identifiziert und bearbeitet (Bogdanova et al., 2015, S. 123). Duplikate der Kategorien 2 und 3 sind schwerer zu identifizieren und zu bearbeiten, da es oftmals schwierig ist, Nuancen in der Fragestellung zu vergleichen und festzulegen, ob die Antwort einer Frage die andere zufriedenstellend beantworten kann (Bogdanova et al., 2015, S. 123). Trotz der bisherigen Bemühungen steigt die Anzahl von Duplikaten weiterhin ununterbrochen an (M. Ahasanuzzaman et al., 2016, S. 411).

Im Rahmen dieser Arbeit werden Modelle auf Basis von *Doc2Vec* entwickelt, welche Duplikate automatisch erkennen und klassifizieren sollen, um Benutzern auf *StackOverflow* Zeit und Aufwand einzusparen und gegen den stetigen Strom an neuen Duplikaten anzukommen.

1.2 Ziele

Das Ziel dieser Arbeit ist es, zu zeigen, ob mithilfe eines *Doc2Vec*-Modells ein Datensatz an Fragen der jeweiligen inhaltlichen Bedeutung nach gruppiert werden kann und ob darüber hinaus neue, dem implementierten Modell unbekannte, Fragen durch eine semantisch möglichst ähnliche Frage und deren Antwort aus dem bestehenden Datensatz beantwortet werden kann. Dafür wird im Rahmen dieses Forschungsprojektes an verschiedenen *Doc2Vec*-Modellen geforscht, welche auf Grundlage des öffentlichen Datensatzes des Forums *StackOverflow* arbeiten. Mithilfe der Modelle werden Fragen in Vektoren umgewandelt, die dann zusammen in einem Vektorraum dargestellt werden können. Die semantische Nähe zwischen zwei Fragen aus dem Datensatz soll anschließend durch die Distanz ihrer Vektoren im Vektorraum berechnet werden. Sobald eine hohe Ähnlichkeit zwischen zwei unterschiedlichen Fragen gefunden wird, wird evaluiert, ob sich die zweite, dem Modell zunächst unbekannte, Frage durch die Antwort der ersten Frage beantworten lässt. Mithilfe

dieser Methodik können so im weiterführenden Schritt Fragen im Datensatz als Duplikate klassifiziert werden. Folgende Forschungsfragen dienen als Qualitätskontrolle und definieren die Leistungsziele:

1. Ist eine semantische Gruppierung der eingelesenen Fragen erkennbar?
2. Können neue Fragen durch die akzeptierte Antwort der nächsten Frage im Vektorraum zufriedenstellend beantwortet werden?

Wichtig zu beachten ist hierbei, dass das zu implementierende Modell nicht optimiert wird, sondern nur hinlänglich der Machbarkeit in einfachem Umfang geprüft wird. Auf Basis der Evaluation der Ergebnisse dieser Arbeit soll sich abschätzen lassen, ob sich das vorgeschlagene Modell zur Erkennung von Duplikaten eignet.

1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in sechs Kapitel. Während das erste Kapitel einleitend die Motivation und die Forschungsziele der Arbeit darlegt, werden im zweiten Kapitel wichtige Begriffe und die Theorie hinter *Document Embeddings* erklärt. Besonders im Vordergrund steht hier die Erklärung der Implementation *Doc2Vec*, auf welcher die Implementation dieser Arbeit aufbaut. Im dritten Kapitel wird die Wahl der Methodik erläutert und die (technische) Vorgehensweise zur Beantwortung der Forschungsziele erklärt. Dazu wird der Aufbau des bearbeiteten Datensatzes von *StackOverflow* erklärt und die Aufbereitung desselben im Kontext dieser Arbeit. Das vierte Kapitel gibt einen Überblick über die technische Implementation und beantwortet die Forschungsfragen, die im Rahmen der Definition der Forschungsziele definiert wurden. Im fünften Kapitel werden die Ergebnisse, die in Kapitel vier besprochen wurden, kritisch diskutiert. Hierbei wird besonders auf die Erreichung der Forschungsziele eingegangen, Limitationen der Arbeit und Abweichungen von den Forschungsfragen werden transparent erläutert. Das sechste und letzte Kapitel fasst die wichtigsten Kernthesen und Erkenntnisse kurz zusammen und bildet so das Fazit der Arbeit.

2 Theoretischer Hintergrund

Der theoretische Hintergrund dieser Arbeit besteht vor allem aus dem Konzept der Berechnung von Vektoren auf Basis von Wörtern beziehungsweise Textabschnitten. Diese Vektoren repräsentieren die Bedeutung eines eingelesenen Textes im Vergleich mit anderen eingelesenen Texten und deren berechneten

Vektoren. Populäre Erkenntnisse lieferten auf diesem Gebiet vor allem Mikolov, Chen et al., 2013), Mikolov, Sutskever et al., 2013 und Le & Mikolov, 2014.

2.1 Begriffserklärung

2.1.1 Word/Document Embeddings

Word Embeddings ist der Sammelbegriff für eine Reihe von Feature-Lerntechniken in *Natural Language Processing (NLP)*, bei denen Wörter oder Sätze auf Vektoren reeller Zahlen abgebildet werden. Ziel dieser Vektoren ist es semantische Bedeutung von Wörtern einzufangen. Document Embeddings wenden ähnliche Verfahren auf Texte ungeachtet ihrer Länge an.

2.1.2 Paragraph Vector

Paragraph Vector ist ein Algorithmus, der durch Le und Mikolov publiziert wurde. Der Algorithmus ist unkontrolliert (deutsch für *unsupervised*), beziehungsweise ohne vorher definierte Ziele aufgebaut. Er klassifiziert Dokumente variabler Länge und repräsentiert diese als Vektoren festgelegter Größe, die nach initialem Training, Wörter in einem vorliegenden Dokument aufgrund vorheriger Wörterkombinationen vorhersagen (Le & Mikolov, 2014, S. 1188). Die beschriebene festgelegte Größe wird im weiteren Teil der Arbeit als Vektordimension bezeichnet.

2.1.3 Paragraph Vector-Distributed Memory

Paragraph Vector-Distributed Memory (PV-DM) betitelt eine Funktionsweise von *Paragraph Vector*, welche darauf beruht, dass das eindeutige Zeichen (der Token) des als Vektor dargestellten Paragraphen, als Speicher fungieren. Dieser Speicher merkt sich im aktuellen Kontext den fehlenden Inhalt beziehungsweise das (per Algorithmus berechnete) Thema des Paragraphen (Le & Mikolov, 2014, S. 1190). Dieses Modell bietet gegenüber der anderen Funktionsweise auf Grundlage von *Bag-of-Words* den Vorteil, dass es die Reihenfolge von eingelesenen Wörtern in den jeweiligen Dokumenten beibehält und bis zu einem gewissen Grad berücksichtigt (Le & Mikolov, 2014, S. 1191).

2.1.4 Doc2Vec

Doc2Vec ist eine Implementierung des Algorithmus *Paragraph Vector* (siehe Kapitel 2.1.2), veröffentlicht durch gensim¹. *Doc2Vec* lässt sich als den Nachfolger

¹ siehe <https://radimrehurek.com/gensim/>

beziehungsweise als die Erweiterung von *Word2Vec* ansehen, der ähnlich wie *Doc2Vec* inhaltliche Bedeutung von Wörtern durch Vektoren abbildet. Der wichtige Unterschied zu *Word2Vec* ist, dass *Doc2Vec* für die Umwandlung von ganzen Textabschnitten, im Gegensatz zu einzelnen Wörtern, in Vektoren optimiert wurde. Die Implementation von *Doc2Vec* lässt verschiedene Funktionsweisen des *Paragraph Vector* zur Auswahl zu. Im Rahmen dieser Arbeit wird die Implementation *PV-DM* (siehe Kapitel 2.1.3) gewählt.

2.1.5 K-Means

K-Means ist ein populärer Algorithmus zur Partitionierung beziehungsweise Clustererstellung von Datenpunkten. Er ist vergleichsweise schnell und findet lokal optimale Lösungen (unter Beachtung der Fehlerquote), um Datenpunkte in einzelne Cluster zu unterteilen. Dabei wird das Zentrum des jeweiligen Cluster während unterschiedlichen Iterationen immer wieder angepasst, um die Fehlerquote der Clustererstellung möglichst gering zu halten (Likas et al., 2003). *K-Means* ist nicht deterministisch.

2.2 Stand der Forschung

Die ersten Modelle zur semantischen Klassifizierung von Text werden als sogenannte *Feed-forward neural network language models* von Bengio et al. vorgestellt, spätere Forschung optimiert dies und gestaltet die Methoden einfacher und effizienter (Bengio et al., 2003; Lau & Baldwin, 2016). Mikolov, Chen et al. (2013), sowie Mikolov, Sutskever et al. (2013) erreichen 2013 große Durchbrüche im Bereich der Vektorisierung von Wörtern im Kontext eines Textabschnittes, was die Effizienz und Effektivität von semantischer Klassifizierung von Wörtern angeht. Die Implementation des vorgeschlagenen Algorithmus wird als *Word2Vec* vorgestellt, aus dem sich durch Forschung von Le und Mikolov (2014) der Algorithmus *Doc2Vec* entwickelt, der nun nicht nur einzelne Wörter, sondern ganze Sätze und Abschnitte in Vektoren umwandeln kann.

Es gibt bereits Forschungen im Bereich der automatischen Erkennung von Text-Duplikaten, auch in Bezug auf Fragen aus StackOverflow. So präsentieren beispielsweise Bogdanova et al. (2015, S. 125) eine Strategie zur Klassifizierung auf Basis eines neuronalen Netzwerks. Sie finden heraus, dass das vorgeschlagene *Convolutional Neural Network* in verschiedenen Themengebieten und mit kleineren Trainingsdatensätzen eine sehr hohe Genauigkeit aufweisen kann, während ein Ansatz

auf *Support Vector Machines* aufbauend erst bei großen Trainingsdatensätzen hohe Erfolgsraten zeigt (Bogdanova et al., 2015, S. 130).

M. Ahasanuzzaman et al. (2016, S. 411) stellen 2016 eine neue Methode mit dem Namen *Dupe* vor. *Dupe* übertrifft den früher vorgestellten *DupPredictor*, veröffentlicht von Y. Zhang et al. (2015), signifikant und kann bessere Ergebnisse in sechs verschiedenen Fragengruppen erzielen (M. Ahasanuzzaman et al., 2016, S. 410).

Die Forschung von Tao et al. (2013) im Bereich der Duplikat-Erkennung von Tweets ist für diese Arbeit ebenfalls relevant, da Tweets als Kurznachrichten einen ähnlichen Umfang haben, wie die in dieser Arbeit genutzten Fragentitel.

3 Methodik

3.1 Wahl der Methodik

Die Seminararbeit besteht aus zwei Teilen, zum einen aus der Datenaggregation und der Implementierung von Modellen auf Basis von *Doc2Vec* und zum anderen aus der Evaluation der gelieferten Ergebnisse. Wie bereits eingangs erläutert, stellt diese Arbeit eine Machbarkeitsstudie dar, weswegen die Optimierung des zu implementierenden Modells nicht im Vordergrund steht und das zu prüfende Konzept grundsätzlich vereinfacht werden kann. In den nachfolgenden Kapiteln 3.2.1 und 3.2.2 wird näher auf die Implementation und auf die Datenaggregation beziehungsweise Datenaufbereitung eingegangen. Zur Implementation des Softwareartefaktes wurde die Zielsprache Python gewählt. Das Softwareartefakt wird innerhalb eines Jupyter Notebook implementiert, welches unter der folgenden URL abgerufen werden kann:

https://github.com/NicolasSchlicht/stackoverflow_duplicate_detection

Das Modell dieser Arbeit benutzt zur Klassifizierung von Fragenduplikaten den Algorithmus *Doc2Vec*, da *Doc2Vec* vergleichsweise sehr gute Werte bezüglich der Klassifizierung von Dokumenten gezeigt hat (Lau & Baldwin, 2016; Le & Mikolov, 2014). Auf der anderen Seite ist die Forschung der Erkennung von Fragenduplikaten im Kontext zu PCQA-Plattformen auf Basis von *Doc2Vec* noch nicht sehr ausgereift und es ist Forschungspotenzial vorhanden.

3.2 Vorgehensweise

3.2.1 Art der Implementation

Wie in Kapitel 3.2.2 ausführlicher beschrieben wird, ist der zugrundeliegende Datensatz von Fragen, Antworten und weiteren Metadaten des öffentlichen Forums *StackOverflow* durch Stack Exchange Inc. veröffentlicht². Die Daten, die für diese Arbeit aufbereitet wurden, befinden sich in einer lokalen, relationalen *Postgre-SQL*-Datenbank³.

Für die Umsetzung von *Doc2Vec* wird die Implementation von *gensim*⁴ verwendet. Es ist möglich *Doc2Vec* mit *PV-DM* (siehe Kapitel 2.1.3) oder mit *Bag-of-Words (BOW)* zu initialisieren. Obwohl Lau und Baldwin (2016) herausfanden, dass *BOW* bei der Klassifizierung von Frageduplikaten in Stack Exchange-Foren im Vergleich überwiegend bessere Ergebnisse erzielt, wird im weiteren Verlauf dieser Arbeit mit *PV-DM* gearbeitet. In der Studie von Lau und Baldwin wurde mit einem signifikant größeren Datensatz gearbeitet und statt nur dem Fragentitel wurde ebenfalls die Beschreibung der Frage miteinbezogen, sodass die Datensätze durchschnittlich 130 Wörter enthielten (Lau & Baldwin, 2016). Die vorliegenden Datensätze der Fragentitel enthalten durchschnittlich 8 Wörter, weswegen die Reihenfolge der Wörter als semantischer Indikator wichtiger wird. Eine der Nachteile von *BOW* ist, dass die Informationen über die Reihenfolge nicht miteinbezogen wird (Le & Mikolov, 2014, S. 1188), weswegen *PV-DM* als die sinnvollere Variante erscheint.

Um die Fragen für die Modellerstellung durch *Doc2Vec* vorzubereiten, werden die (Fragen-) Dokumente in Arrays umgewandelt, sodass jedes Wort und jedes Zeichen eine eigene Position im Array innehalten. Als nächstes wird ein Ansatz zur Reduzierung der Datenmenge gewählt (Y. Mizobuchi & K. Takayama, 2017, S. 563), der die Bedeutung der einzelnen Wörter der Dokumente erhöht, indem Wörter, die für die inhaltliche Bedeutung der Frage unerheblicher sind, entfernt werden. Dazu werden die Fragmente gegen eine Ausschlussliste verglichen, in der häufig gebrauchte englische Wörter stehen, die wenig Einfluss auf den Inhalt des Dokumentes haben. Im Rahmen dieser Arbeit wurde die Bibliothek *nlTK* verwendet, welche von Bird et al.

² siehe <https://archive.org/download/stackexchange>

³ siehe <https://www.postgresql.org/>

⁴ siehe <https://radimrehurek.com/gensim/models/doc2vec.html>

(2009) beschrieben wird. Diese Ausschlussliste wird im Rahmen dieser Arbeit um weitere Zeichen ergänzt, die häufig auftreten, im normalen englischen Sprachgebrauch nicht verwendet werden und deswegen auch nicht in der Bibliothek *nltk* enthalten sind. Eine vollständige Auflistung der Ausschlussliste befindet sich im Anhang.

Nachdem die Fragen in den reduzierten Arrays gespeichert sind, wird jedem Array ein eindeutiges Label zugewiesen, welches für das Trainieren des *Doc2Vec*-Modells wichtig ist. Grundsätzlich werden, wenn ein Modell im Bereich Data Science erstellt und trainiert wird, folgende Grundlagen beachtet. Die Daten zum initialen Erstellen und Training müssen von einem zweiten Set an Daten, welches zum Testen des Modells gebraucht wird, getrennt sein. Grundsätzlich wird also ein Modell trainiert und wird dann mit dem Modell unbekannten neuen Daten auf Funktionalität getestet. Im Weiteren werden diese beiden Sets Training-Set und Test-Set genannt. Zur Erstellung des Modells mithilfe von *Doc2Vec* werden die folgenden Parameter vergeben:

Tabelle 1: Parameter Doc2Vec

Parameter	Bedeutung
vector_size	legt die Dimensionalität der Vektoren fest
alpha	legt die Lernrate fest
min_alpha	legt die minimal erreichbare Lernrate fest
min_count	legt fest, welche Mindestanzahl Wörter haben müssen, um im Modell berücksichtigt zu werden
workers	legt die Verteilung auf Prozessorkerne fest
dm	legt fest, ob mit PV-DM oder BOW gearbeitet wird

3.2.2 Aufbau und Aufbereitung des Datensatzes

Zum Zeitpunkt der Veröffentlichung der zu bearbeitenden Daten (02.03.2020) sind 19.015.984 Fragen und 28.812.437 Antworten (9.926.109 akzeptierte Antworten) im Datensatz vorhanden. Alle Fragen und Antworten sind in englischer Sprache verfasst, dementsprechend baut diese Studie auf englischsprachigen Daten auf.

Für die Durchführung der Studie im Rahmen dieser Arbeit werden die ersten 50.000 Fragen, die mit *Java* getaggt sind und die ersten 50.000 Fragen, die mit *Python* getaggt sind, ausgewählt und verfügbar gemacht. 337 Fragen unter den ersten 100.000 Fragen im Datensatz sind gleichzeitig mit *Java* und *Python* getaggt, weswegen weitere 337 Fragen mit dem Tag *Python* hinzugefügt wurden, um insgesamt 100.000 Datensätze für das Training des Modells in der Datenbank zur Verfügung zu haben.

Es wurden Fragen im Zeitraum vom 01.08.2008 bis zum 08.10.2012 einschließlich für das Training-Set ausgewählt, da dies den Vorteil bietet, dass ein möglichst natürlicher Schnitt der Themen der gestellten Fragen vorhanden ist. Der Tag *Java* wurde ausgewählt, da es sich in dem Zeitraum um eine der meistgebrauchten Programmiersprachen auf *StackOverflow* handelt (Treude et al., 2011, S. 804). Der Tag *Python* hingegen bildet einen gewissen Gegenpol, da die Sprache im Zeitraum der extrahierten Fragen erst an Wachstum gewann und noch nicht so populär war wie heutzutage. Dadurch lässt sich vermuten, dass die gestellten Fragen vergleichsweise simpel gehalten sind und somit eventuell gute Ergebnisse in der Klassifizierung erzielen kann. Der Vorteil von zwei ausgewählten Programmiersprachen ist zusätzlich, dass einfache Probleme, die in beiden Sprachen ähnlich gelöst werden, dementsprechend semantisch zugeordnet werden könnten.

Fragen, die auf *StackOverflow* gestellt werden, bestehen aus einem Fragetitel, der die Frage beziehungsweise das Thema der Frage zusammenfasst und aus einer Beschreibung, in der der Fragesteller das Thema näher erläutern kann. Im weiteren Teil dieses Forschungsprojektes wird der Fragetitel mit der Frage an sich gleichgesetzt, um die Komplexität des zu bildenden Modells gering zu halten. Weiterhin wird durch den Gebrauch der Titel der Fragen, die im Training-Set durchschnittlich 8 Wörter enthalten, das statistische Rauschen minimiert, da die Titel möglichst eindeutig gewählt werden sollten und dadurch einzelne Begriffe im Titel mehr Aussagekraft haben. Zusätzlich werden, wie in Kapitel 3.2.1 beschrieben, häufig gebrauchte englische Wörter, entfernt. Dadurch ergibt sich die Anzahl von insgesamt 550.120 Wörtern, die in den 100.000 Fragen enthalten sind und im Weiteren verwendet werden.

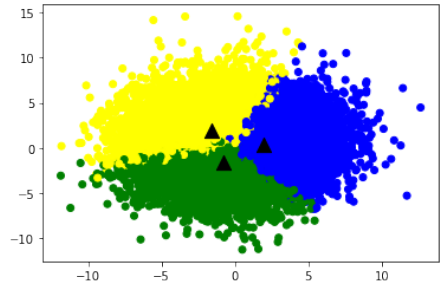
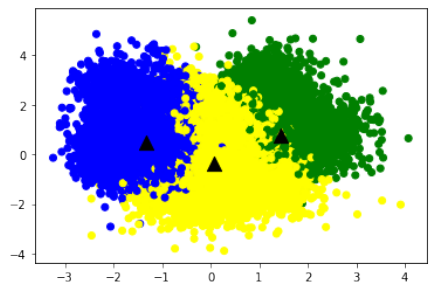
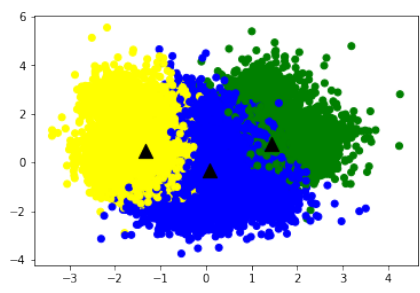
Das Test-Set enthält nach Beispiel des Training-Sets jeweils die nächsten 10.000 Fragen mit dem Tag *Java* beziehungsweise *Python* in einer separaten Tabelle der Datenbank. Das Test-Set ist für das Training des Modells nicht relevant und wird für die Evaluation des Modells gebraucht. Zur Evaluation des Modells werden zusätzlich, die im Forum durch den Fragesteller akzeptierten, Antworten der Fragen des Training-Sets in einer separaten Tabelle abgelegt.

4 Ergebnisse

4.1 Code

Im Rahmen dieser Forschungsarbeit wurden *Doc2Vec*-Modelle mit verschiedenen Parametern erstellt und miteinander verglichen. Mithilfe der Implementation der Bibliothek *nlTK* des Algorithmus *K-Means* lassen sich die aufgrund des *Doc2Vec*-Modells berechneten Vektoren in verschiedene Gruppen beziehungsweise Cluster unterteilen. Es hat sich gezeigt, dass die geringste Fehlerquote bei einer Dreiteilung des Datensatzes auftritt.

Tabelle 2: Clusterbildung bei verschiedenen Vektordimensionen

Dimension	Cluster	Vergleicht man
24		unterschiedliche Dimensionen von Vektoren, so erkennt man auch graphisch sehr eindeutig, dass die Clusterverteilung zwischen Dimension 24 und 300
300		sehr stark variiert, während sich die Clusterbildung zwischen 300 und 1000 nur noch wenig verändert. Mit diesem Hintergrund werden für den evaluierenden Teil dieser Arbeit die Vektordimensionen 24 (Modell 1) und 300 (Modell 2) ausgewählt, sodass die beiden Modelle hinsichtlich ihrer Performanz verglichen werden können. Alle weiteren Parameter, die in Kapitel 3.2.1 beschrieben wurden und für <i>Doc2Vec</i> notwendig sind
1000		wurden für die verschiedenen Modelle nicht verändert und weichen, bis auf die Wahl

wurden für die verschiedenen Modelle nicht verändert und weichen, bis auf die Wahl

von *PV-DM* statt *BOW*, in keiner relevanten Position von den Standardwerten von *Doc2Vec* ab.

4.2 Beantwortung der Forschungsfragen

Im Folgenden werden die zwei definierten Forschungsfragen anhand der beiden, in Kapitel 4.1 definierten, *Doc2Vec*-Modelle beantwortet.

4.2.1 Ist eine semantische Gruppierung der eingelesenen Fragen erkennbar?

Zunächst wird das erste Modell mit 24 Dimensionen pro Vektor betrachtet. Um die drei resultierenden Cluster auf ihre semantische Zugehörigkeit zu untersuchen, werden als erstes die Wörter extrahiert, die jeweils am häufigsten in den Fragentiteln vorkommen. Dabei zeigt sich, dass Begriffe wie „Java“ oder „Python“ in allen drei Clustern ähnlich oft vorhanden sind. Beschränkt man im nächsten Schritt die Cluster auf die Wörter, die in den hundert häufigsten Wörtern über die Cluster hinweg einzigartig sind, erhält man durchschnittlich etwa 27 Wörter pro Cluster. Durch die Analyse dieser Wörter lassen sich die übergeordneten Themen der Cluster beschreiben. In Tabelle 3 werden exemplarisch fünf aussagekräftige Wörter der einzelnen Cluster aufgelistet, eine vollständige Übersicht der Wörter befindet sich im Anhang.

Tabelle 3: Themen der Cluster (Dim. 24)

Cluster 1: Metaebene	Cluster 2: Umsetzung	Cluster 3: Bibliotheken
client	return	xml
problem	call	numpy
different	characters	regex
custom	methods	matplotlib
dynamic	variables	maven
[...]	[...]	[...]

Wie bereits in der Tabelle betitelt, lassen sich die drei Cluster durch die Überschriften *Metaebene*, *Umsetzung* und *Bibliotheken* beschreiben. Das Cluster *Metaebene* enthält zum Großteil Wörter, die Sachverhalte in der sogenannten Metaebene (griechisch von *meta* für *über*) beschreiben. Es geht hierbei nicht zwingend um inhaltliche Auseinandersetzung, sondern eher um allgemeine Konzepte. Das zweite Cluster beschrieben durch *Umsetzung*, gruppiert Fragen der inhaltlichen Auseinandersetzung mit beispielsweise programmiertechnischen Werkzeugen. So

sind die Themen *Variablen* oder *Methoden* vertreten. Das dritte Cluster beinhaltet größtenteils Wörter, die sich auf verschiedene Bibliotheken beziehungsweise Eigennamen beziehen. Beispielhaft zu nennen sind hier *numpy*, eine populäre Python-Bibliothek oder *XML*, der Name einer Markup-Sprache. Die Gruppierung der Fragen ist jedoch nicht zwingend eindeutig. Wie in Abbildung 1 zu sehen ist, überlappen die einzelnen Cluster teilweise und die Zentren der jeweiligen Cluster sind relativ nah im Zentrum der Punktwolke der Vektoren. Dies lässt darauf schließen, dass Fragen nicht immer eindeutig einem Cluster zugewiesen werden können. Zum Beispiel ließe sich die fiktive Frage „*How do I write custom clients calling variables or methods from modules like numpy or re?*“ anhand der in Tabelle 3 genannten Schlüsselwörter nicht eindeutig zu einem Cluster zuordnen.

Bei der Betrachtung des zweiten Modells mit 300 Vektordimensionen fällt auf, dass nun nicht die Art der Fragestellung, sondern die Unterteilung der beiden Sprachen ausschlaggebend für die Clustererstellung ist. Es fällt auf, dass die Begriffe *Python* und *Java* zwar in jedem der drei Cluster enthalten sind, jedoch jeweils zu über 90% nur in einem Cluster enthalten sind. Dementsprechend liegt es nahe, dass die beiden Cluster mit der jeweiligen dominierenden Sprache betitelt werden. Das dritte Cluster enthält, ähnlich wie im vorherigen Modell, vorwiegend Bibliotheken und Themen, die nicht zwingend eine eindeutige Zuordnung zu *Java* oder *Python* zulassen.

Tabelle 4: Themen der Cluster (Dim. 300)

Cluster 1: Bibliotheken	Cluster 2: Python	Cluster 3: Java
spring	dictionary	methods
numpy	functions	interface
url	csv	generic
query	lists	threads
sqlalchemy	json	client
[...]	[...]	[...]

Ein interessanter Vergleich zwischen Cluster 2 und Cluster 3 (siehe Tabelle 4) ist, dass *methods* im Cluster Java enthalten ist, während das programmiertechnische Äquivalent dazu, *functions* einzigartig im Cluster Python enthalten ist. *numpy* ist zwar eine Bibliothek, die nur unter Python benutzt werden kann, ist allerdings dem Modell nach enger verbunden mit *spring*, einem Java-Framework. *csv* als Importvariante von

externen Daten wird zwar nicht exklusiv im Python-Kontext verwendet, ist jedoch ein wichtiger Bestandteil im Bereich Data Science, in der Python als Programmiersprache populär ist. Die vollständige Übersicht der einzigartigen Worte unter den hundert häufigsten Wörtern der Cluster befindet sich im Anhang.

Abschließend lässt sich die Forschungsfrage bejahen. Es wurde gezeigt, dass es mithilfe von *Doc2Vec* möglich ist, Fragen, beziehungsweise allgemeiner gesprochen, Dokumente semantisch zu gruppieren. Weiterhin ist es sogar möglich über eine Beeinflussung der Vektordimensionen, unterschiedliche Arten der Gruppierung zu erhalten.

4.2.2 Können neue Fragen durch die akzeptierte Antwort der nächsten Frage im Vektorraum zufriedenstellend beantwortet werden?

Um die oben genannte Forschungsfrage beantworten zu können, muss zunächst der Begriff „zufriedenstellend“ im Rahmen dieser Arbeit definiert werden. Bogdanova et al. (2015, S. 123) definieren zwei Fragen als semantisch äquivalent, wenn beide Fragen mit derselben Antwort adäquat beantwortet werden können. Diese Definition wird im Weiteren für die Umschreibung einer „zufriedenstellenden“ Antwort als gegeben erachtet. Die Beantwortung dieser Forschungsfrage wird in zwei Teilen erfolgen. Zunächst wird eine *Vorwärtssuche* auf dem Test-Set angestellt, bei dem geprüft wird, ob und mit welcher Qualität das jeweilige Modell Duplikate im Test-Set erkennt. Der zweite Teil besteht aus einer *Rückwärtssuche*, bei der geprüft wird, ob das jeweilige Modell externe Duplikate von Fragen im Training-Set identifizieren kann.

Für die Vorwärtssuche wird eine minimale Übereinstimmung von 95 Prozent vorausgesetzt. Startet man die Abfrage auf Grundlage von Modell 1 (24 Vektordimensionen), erhält man als Ergebnis 18 Fragen, die im Test-Set als semantisch sehr nah zu einer Frage im Training-Set klassifiziert wurden.

Tabelle 5: Vorwärtssuche Test-Set (Dim. 24)

Ähnlichkeit (Dov2Vec)	Fragetitel (Test-Set)		Fragetitel (Training-Set)
95,47%	Capitalization NoClassDefFoundError ClassNotFoundException	and vs	ServletContext.getRequestDispatcher() vs ServletRequest.getRequestDispatcher()
95,07%	Replacing values with groupby means		Regular expression for urlpattern
95,8%	IntelliJ auto import for inner classes		How to do dependency injection python-way?

Wie man in Tabelle 5 exemplarisch beobachten kann, lassen sich gewisse Ähnlichkeiten zwischen den jeweiligen Fragetiteln herleiten. Allerdings ist keine der oben aufgeführten Fragentitel inhaltlich so nah aneinander, dass die Frage des Test-Sets durch die akzeptierte Antwort der Frage aus dem Training-Sets beantwortet werden könnte. Tatsächlich zeigt sich die Art der Gruppierung der Fragen des *Doc2Vec*-Modells, wie in Kapitel 4.2.1 beschrieben, auch in dieser Abfrage. Das *Doc2Vec*-Modell gewichtet die Art, wie eine Frage gestellt wird, mehr als den für Fragesteller verständlichen Inhalt der Frage. So wird beispielsweise die erste Frage in Tabelle 5 als semantisch sehr nah gewertet, weil der Syntax der beiden Fragen sehr ähnlich ist: In beiden Fragen werden zwei jeweils ähnliche Wörter mit einem „vs“ in der Mitte verglichen. Die beiden anderen Abfragen in Tabelle 3 lassen sich ebenfalls als thematisch sehr nah verorten, beispielsweise ist eine *regular expression* (englisch für *regulärer Ausdruck*) ein Hilfsmittel, um bestimmte Zeichenkombinationen in Zeichenketten zu finden. Oftmals werden durch reguläre Ausdrücke auch Zeichen ersetzt – Zeichen beziehungsweise Werte zu ersetzen, spiegelt sich in der Frage aus dem Test-Set wider. Der zur besseren Lesbarkeit gekürzte Fragentext der gestellten Frage ist der folgende:

“I would like to replace values < 0 with the mean of the group that they are in.

For missing values as NAs, I would do:

```
data = df.groupby(['GroupID']).column
```

```
data.transform(lambda x: x.fillna(x.mean()))
```

But how to do this operation on a condition like $x < 0$?”

Die akzeptierte Antwort der Frage im Training-Set ist die folgende:

*“Parameter after # character is not send to server, so
cannot catch in server side script like django.”*

Diese unterstreicht zusätzlich den fehlenden inhaltlichen Bezug, da die Antwort die Frage nicht und besonders nicht zufriedenstellend beantwortet. Grundsätzlich lässt sich also sagen, dass die Fragen des Test-Sets durch semantisch nahe Fragen des Training-Sets nicht zufriedenstellend beantwortet werden können. Zusätzlich enthalten die genannten 18 Ergebnisse Fehler, bei denen kein Zusammenhang zwischen den Fragen erkennbar ist. Eine vollständige Auflistung der Ergebnisse ist angehängen.

Wendet man die *Vorwärtssuche* auf Modell 2 (300 Vektordimensionen) an, erhält man keine Ergebnisse bei vorausgesetzten 95 Prozent in der Übereinstimmungsrate, auch eine Abfrage mit 90 Prozent Übereinstimmung liefert keine Ergebnisse. Die höchste Übereinstimmungsrate ist auf das ganze Test-Set bezogen 71,08 Prozent.

Für die *Rückwärtssuche* werden zu zehn zufällig ausgewählten Fragen des Training-Sets manuell Duplikate erstellt, die sich inhaltlich nicht unterscheiden, sondern andere Wörter und Wortreihenfolgen verwenden. Anhand der Quote dieser Stichprobe, wie viele der Duplikate erkannt und richtig zugeordnet werden, lässt sich beurteilen wie gut das jeweilige Modell Duplikate erkennt.

Tabelle 6: Duplikate für Rückwärtssuche

ID	Frage Training-Set	Duplikat
12590131	How to slice multindex columns in pandas DataFrames?	How do I select from multidimensional DataFrame in Python?
2145826	Convert Unicode/UTF-8 string to lower/upper case using pure & pythonic library	How to convert a UTF-8/Unicode String to lower or upper case using only Python?
4629131	restart local computer from python	Is Python able to restart my computer?
4415133	Where do we put the Servlets in the directory structure of Tomcat?	In which Tomcat folder do I have to put the Servlets?
6331497	An elegant way to get hashtags out of a string in Python?	Is there any good way to remove hastags from strings in Python?

2261650	How to parse just the text from a Word Doc using Python?	How can I access text inside a Word Document with Python?
5361971	Replace all double quotes within String	How to replace double quotes in a string?
4527921	upgrade to javaserver faces. Easy task?	Is it easy to upgrade to javaserver faces?
11895768	Python: string to a list of lists	How do I convert a string to a list of lists in Python?
2850534	Producing a static HTML site from XML content	How to produce a static HTML website from XML?

Modell 1 (24 Vektordimensionen) liefert für alle zehn Abfragen Prozentwerte zwischen 80 und 90 Prozent Übereinstimmung, jedoch ist keine der Fragen aus Tabelle 6 zurückgegeben worden. Keine der zurückgegebenen Fragen ist inhaltlich nah an der Ausgangsfrage. Modell 2 (300 Vektordimensionen) liefert Werte zwischen 53 und 70 Prozent zurück. Im Vergleich zu Modell 1, sind zwar die Prozentwerte geringer, jedoch lassen sich bei einzelnen Abfragen inhaltliche Zusammenhänge zwischen den beiden Fragen erkennen. Die Abfrage, die mit 70,61 Prozent am besten bewertet wurde, hat ein Duplikat im Training-Set gefunden. Dieses Duplikat ist allerdings nicht die Vorlage aus Tabelle 6.

Tabelle 7: gefundenes Duplikat Rückwärtssuche (Dim. 300)

70,61%	How can I convert this string to list of lists?
--------	---

Der Rest der Ergebnisse ist ähnlich wie bei Modell 1 inhaltlich inkohärent zu den abgefragten Fragen.

Abschließend lässt sich die Forschungsfrage verneinen. Mit den vorgelegten Modellen ist es nicht uneingeschränkt möglich Duplikate zwischen zwei Fragen zu erkennen. Daraus ergibt sich, dass auch die zugehörige Antwort der ersten Frage nicht die nächste Frage im Vektorraum (die Frage mit der höchsten Übereinstimmung) zufriedenstellend beantworten kann.

5 Diskussion

Im Rahmen dieses Kapitel werden die Ergebnisse aus Kapitel 4 diskutiert und kritisch begutachtet. Es lässt sich einleitend feststellen, dass nur die erste der beiden

Forschungsfragen bejaht werden konnte. Grundsätzlich wurde das Ziel der Arbeit, die Erstellung eines *Doc2Vec*-Modells, welches Duplikate in Fragen aus dem Datensatz von StackOverflow klassifizieren kann, nicht erreicht. Dies liegt verschiedenen Faktoren zugrunde. Der wichtigste Faktor ist jedoch, dass die ausgewählten und geprüften Modelle nicht imstande waren, paraphrasierte Duplikate zu erkennen.

5.1 Reflexion der Ergebnisse

Um die Ergebnisse dieser Arbeit angemessen zu reflektieren, muss zunächst auf die Ergebnisse der einzelnen Forschungsfragen eingegangen werden.

Wie in Kapitel 4.2.1 ausführlich beschrieben lässt sich die erste Forschungsfrage „Ist eine semantische Gruppierung der eingelesenen Fragen erkennbar?“ positiv beantworten. Es ist gelungen anhand von zwei Modellbeispielen zu zeigen, dass sich der jeweilige Vektorraum in verschiedene Gruppen beziehungsweise Cluster einteilen lässt. Die Methode zur Themenfindung innerhalb der Cluster ist ihrem Anspruch gerecht geworden und hat dazu beigetragen die Cluster zu benennen. Es ist jedoch zu erwähnen, dass unter Verwendung dieser Methode manueller Aufwand für die Interpretation des Themas des jeweiligen Clusters aufzubringen ist. Weiterhin ist ebenfalls zu erwähnen, dass obwohl sich ein deutliches übergeordnetes Thema pro Cluster in beiden Modellen durchgesetzt hat, es trotzdem Ausreißer gegeben hat. Solche Ausreißer waren Wörter dessen inhaltliche Zuordnung zu einem der Cluster nicht komplett zu bestätigen waren. In Form eines Gegenbeispiels wurde bereits in Kapitel 4.2.1 aufgezeigt, dass es Fragen geben kann, die auf Basis der vorhandenen Clustererstellung keinem Cluster eindeutig zugewiesen werden können.

Um die Ergebnisse der zweiten Forschungsfrage „Können neue Fragen durch die akzeptierte Antwort der nächsten Frage im Vektorraum beantwortet werden?“ zu evaluieren, muss zunächst erneut klar gestellt werden, dass die Ergebnisse unzureichend sind und die Forschungsfrage nicht positiv beantwortet werden konnte. Dies hat unterschiedliche Gründe. Zum einen ist die Forschungsfrage an sich nicht explizit genug gestellt. Im Nachhinein macht es mehr Sinn die Frage neu zu stellen und zuerst abzufragen, ob Duplikate eindeutig identifiziert werden. Im Laufe der Forschung im Rahmen dieser Arbeit hat sich eben jene Frage als negativ zu beantworten herausgestellt. Zusätzlich zu der Formulierung der Frage wurde die Evaluationsmethodik mit Mängeln gewählt. So hat sich herausgestellt, dass im bearbeiteten Teil der Fragen im Datensatz von StackOverflow keine markierten

Duplikate vorhanden waren, sodass die Güte der Modelle in Form einer manuell geführten Stichprobe getestet werden mussten. Eine Stichprobe lässt die Beurteilung eines Sachverhaltes allerdings nur bis zu einem gewissen Grad zu und kann daher ein Modell auch nur bis zu einem gewissen Grad validieren. Im vorliegenden Fall hat sich die Stichprobe allerdings als nützlich bewiesen, die Funktionstüchtigkeit der getesteten Modelle zu widerlegen, da es in diesem Fall hinreichend ist, Gegenbeispiele aufzudecken.

5.2 Kontext zur Forschung

Die Ergebnisse dieser Arbeit haben gezeigt, dass sich Fragen von *StackOverflow* durch verschiedene *Doc2Vec*-Modelle gruppieren und kategorisieren lassen. Dies sollte in weiterführenden Forschungen als Kernthema aufgegriffen werden, um herauszufinden, welche weiteren Arten von Gruppierungen in Anwendungsfällen Sinn machen. So könnte es möglicherweise sinnvoll sein, Suchergebnisse auf Plattformen wie *StackOverflow* nach Fragenarten zu gruppieren.

Um die Performanz von *Doc2Vec*-Modellen bezüglich der Identifikation von Duplikaten in zukünftiger Forschung zu verbessern, kann in verschiedene Richtungen geforscht werden. So kann beispielsweise die Anzahl der Datensätze des Training-Sets stark erhöht werden. Es kann ebenso geprüft werden, ob die Performanz zukünftiger Modelle verbessert wird, wenn neben den Fragetiteln auch noch die Beschreibung und eventuell die Tags der Frage mitberücksichtigt werden. Zu guter Letzt gibt es innerhalb der Erstellung von Modellen mit *Doc2Vec* die Möglichkeit *BOW*, statt *PV-DM* auszuwählen und die Veränderung der Performanz zu untersuchen.

Es sollte auf jeden Fall darauf geachtet werden, dass in zukünftigen Datensätzen markierte Duplikate enthalten sind, damit die Modelle gegen echte Duplikate getestet werden können und dadurch die Validation der Modelle stark verbessert werden kann.

5.3 Limitationen des Projekts

Die Forschung im Rahmen dieses Projekts ist mehreren Limitationen ausgesetzt, die die Ergebnisse verfälschen können. Wie bereits in vorangegangenen Kapiteln erwähnt wurde, ist der Datensatz vergleichsweise klein und sollte für weitere Forschung angepasst werden.

Der Gebrauch des Algorithmus *K-Means* zur Clustererstellung sorgt ebenfalls für Limitationen, da die Ausführung nicht deterministisch ist und es bei jedem

Durchlauf des Algorithmus zu verschiedenen Ergebnissen kommen kann. Im Rahmen dieser Arbeit wurde jedoch jede Clustererstellung mehrfach wiederholt und die Abweichungen sind vergleichsweise klein und daher nicht signifikant für die Bewertung der Ergebnisse.

Es hat sich zudem herausgestellt, dass *Doc2Vec* gerade in Modellen mit einer geringen Anzahl von Vektordimensionen Syntax der Semantik vorzieht. Dies lässt sich anhand der Kategorisierung von Modell 1 nachvollziehen. Um allerdings die zweite Forschungsfrage positiv zu beantworten ist die Semantik deutlich wichtiger als der Syntax der Fragen. Diese Abweichung von der Forschungsfrage lässt sich in der Natur von *Doc2Vec* verorten, da der Algorithmus *unsupervised* ist. Es gibt bei der Erstellung keine Möglichkeit die Erstellung des Modells direkt zu manipulieren, um den Bezug zum Forschungsziel zu wahren.

6 Fazit

Das Forschungsziel dieser Arbeit ist es, Duplikate in einem Datensatz von Fragen der Plattform *StackOverflow* zu erkennen. Dazu wurden zunächst zwei Forschungsfragen definiert, die das insgesamt Forschungsvorhaben in zwei Teilen zusammenfassen und die Evaluation der Erreichung des Forschungsziels unterstützen. Grundsätzlich lässt sich sagen, dass das allgemeine Forschungsziel nur zu Teilen erreicht wurde.

Im Rahmen dieser Arbeit wurden zwei *Doc2Vec*-Modelle vorgestellt, die aufgrund unterschiedlicher Vektordimensionen andere Ergebnisse auf die gestellten Forschungsfragen geliefert haben. Während die zweite Forschungsfrage „Können neue Fragen durch die akzeptierte Antwort der nächsten Frage im Vektorraum zufriedenstellend beantwortet werden?“ auf Basis der vorgestellten Modelle verneint werden muss, kann die erste Forschungsfrage „Ist eine semantische Gruppierung der eingelesenen Fragen erkennbar?“ bejaht werden. Es ist zusätzlich zum Forschungsvorhaben gelungen zu zeigen, dass verschiedene Modelle verschiedene valide Gruppierungen des Datensatzes erzeugen können. So lässt sich beispielsweise mithilfe eines Modells mit wenigen Vektordimensionen eine Gruppierung erstellen, die gegebene Fragen der Plattform *StackOverflow* nach der Art der Frage sortiert. Es wurde unterschieden zwischen fachspezifischen Fragen zu Bibliotheken und übergeordneten Themen, zwischen praktischen Fragen zur programmiertechnischen

Umsetzung und zwischen Fragen, die auf der Metaebene stattfinden, beispielsweise der Vergleich zwischen unterschiedlichen Programmier Techniken. Unter der Verwendung von höheren Vektordimensionen wurde zudem herausgefunden, dass sich die Fragen auch nach inhaltlichem Thema sortieren lassen – so wurden Fragen in Modell 2 beispielsweise entweder dem Themenblock *Java* oder *Python* zugeordnet.

Literaturverzeichnis

- Bengio, Y., Ducharme, R., Vincent, P. & Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb), 1137–1155.
- Bian, J., Liu, Y., Agichtein, E. & Zha, H. (2008). Finding the right facts in the crowd. In J. Huai (Hg.), *Proceedings of the 17th international conference on World Wide Web* (S. 467). ACM. <https://doi.org/10.1145/1367497.1367561>
- Bird, S., Klein, E. & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."
- Bogdanova, D., dos Santos, C., Barbosa, L. & Zadrozny, B. (2015). Detecting Semantically Equivalent Questions in Online User Forums. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning* (S. 123–131). Association for Computational Linguistics. <https://doi.org/10.18653/v1/K15-1013>
- Jeff Atwood. (2009). *Handling Duplicate Questions*. <https://stackoverflow.blog/2009/04/29/handling-duplicate-questions/#:~:text=Either%20they're%20not%20satisfied,borderline%20abuse%20of%20the%20system.>
- Lau, J. H. & Baldwin, T. (2016). An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation. *Proceedings of the 1st Workshop on Representation Learning for NLP, Berlin, Germany*, pp. 78-86. <http://arxiv.org/pdf/1607.05368v1>
- Le, Q. & Mikolov, T. (2014). Distributed representations of sentences and documents. In *International conference on machine learning*.
- Likas, A., Vlassis, N. & J. Verbeek, J. (2003). The global k-means clustering algorithm. *Pattern Recognition*, 36(2), 451–461. [https://doi.org/10.1016/S0031-3203\(02\)00060-2](https://doi.org/10.1016/S0031-3203(02)00060-2)
- M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy & K. A. Schneider (2016). Mining Duplicate Questions of Stack Overflow. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*.
- Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*.
- P. Berger, P. Hennig, T. Bocklisch, T. Herold & C. Meinel (2016). A Journey of Bounty Hunters: Analyzing the Influence of Reward Systems on StackOverflow Question Response Times. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*.
- Stack Exchange Inc. (Hg.). (2019). *Developer Survey Results 2019*.
<https://insights.stackoverflow.com/survey/2019>
- Storey, M.-A., Treude, C., van Deursen, A. & Cheng, L.-T. (2010). The impact of social media on software engineering practices and tools. In G.-C. Roman (Hg.), *Proceedings of the FSESDP workshop on Future of software engineering research* (S. 359). ACM. <https://doi.org/10.1145/1882362.1882435>
- Tao, K., Abel, F., Hauff, C., Houben, G.-J. & Gadiraju, U. (2013). Groundhog day. In D. Schwabe (Hg.), *Proceedings of the 22nd International Conference on the World Wide Web: May 13 - 17, 2013, Rio de Janeiro, Brazil* (S. 1273–1284). ACM. <https://doi.org/10.1145/2488388.2488499>
- Treude, C., Barzilay, O. & Storey, M.-A. (2011). How do programmers ask and answer questions on the web? In R. N. Taylor, H. Gall & N. Medvidović (Hg.), *33rd International Conference on Software Engineering (ICSE), 2011: 21 - 28 May 2011, Waikiki, Honolulu, HI, USA* (S. 804). IEEE.
<https://doi.org/10.1145/1985793.1985907>
- Y. Mizobuchi & K. Takayama (2017). Two improvements to detect duplicates in Stack Overflow. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*.
- Zhang, W. E., Sheng, Q. Z., Lau, J. H. & Abebe, E. (2017). Detecting Duplicate Posts in Programming QA Communities via Latent Semantics and Association Rules. In R. Barrett, R. Cummings, E. Agichtein & E. Gabrilovich (Hg.), *WWW '17: Proceedings of the 26th International Conference on World Wide Web : May 3-7, 2017, Perth, Australia* (S. 1221–1229). International World Wide Web Conferences Steering Committee. <https://doi.org/10.1145/3038912.3052701>

Zhang, Y., Lo, D., Xia, X. & Sun, J.-L. (2015). Multi-Factor Duplicate Question Detection in Stack Overflow. *Journal of Computer Science and Technology*, 30(5), 981–997. <https://doi.org/10.1007/s11390-015-1576-4>

A. Anhang – Ausschlussliste Wörterreduzierung

i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, him, his, himself, she, she's, her, hers, herself, it, it's, its, itself, they, them, their, theirs, themselves, what, which, who, whom, this, that, that'll, these, those, am, is, are, was, were, be, been, being, have, has, had, having, do, does, did, doing, a, an, the, and, but, if, or, because, as, until, while, of, at, by, for, with, about, against, between, into, through, during, before, after, above, below, to, from, up, down, in, out, on, off, over, under, again, further, then, once, here, there, when, where, why, how, all, any, both, each, few, more, most, other, some, such, no, nor, not, only, own, same, so, than, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren, aren't, couldn, couldn't, didn, didn't, doesn, doesn't, hadn, hadn't, hasn, hasn't, haven, haven't, isn, isn't, ma, mightn, mightn't, mustn, mustn't, needn, needn't, shan, shan't, shouldn, shouldn't, wasn, wasn't, weren, weren't, won, won't, wouldn, wouldn't, ?, :, (,), ;, ,, `', -, 's, '

B. Anhang – Themengenerierung Cluster (Dim. 24)

Cluster 1: Metaebene	Cluster 2: Umsetzung	Cluster 3: Bibliotheken
without	equivalent	xml
problem	call	numpy
different	creating	name
simple	program	database
vs	return	static
+	write	html
&	implement	memory
within	key	url
like	date	regex
working	generate	http
inside	strings	model
question	package	maven
via	process	swing
dynamic	characters	matplotlib
good	element	default
used	elements	jpa
help	variables	jar
2	calling	@
based	implementation	based

large	change	instance
issue	threads	servlet
problems	remove	source
output	[binary
need]	unicode
3	lists	first
nested	query	jsp
vs.	install	system
client	format	directory
instead		logging
single		

C. Anhang – Themengenerierung Cluster (Dim. 300)

Cluster 1: Bibliotheken	Cluster 2: Python	Cluster 3: Java
spring	dictionary	#
hibernate	lists	methods
numpy	`	date
add	strings	interface
maven	unicode	generic
matplotlib	import	collection
custom	loop	arraylist
query	functions	client
model	html	threads
jpa	command	source
key	output	good
@	parsing	framework
user	print	performance
field	json	int
test	arguments	byte
change	characters	map
url	directory	
form	reading	
jar	dict	
	process	
	remove	

D. Anhang – Ergebnis Rückwärtssuche (Dim. 24)

Übereinstimmung	Frage mit höchster Übereinstimmung (Training-Set)
83,22%	Django select max id
89,34%	Cross platform keylogger
88,63%	Python/PySerial and CPU usage
87,73%	How should I unit test a code-generator?
84,18%	Equivalent of C# OnDeserialized/OnSerialized in Java?
84,74%	pygettext.py and msgfmt.py on Mac OS X
86,59%	How to replace $i_6^{1/2}$ in a string
87,65%	pygettext.py and msgfmt.py on Mac OS X
84,78%	Converting a String to a List of Words?
88,06%	pygettext.py and msgfmt.py on Mac OS X

E. Anhang – Ergebnis Rückwärtssuche (Dim. 300)

Übereinstimmung	Frage mit höchster Übereinstimmung (Training-Set)
65,33%	java.lang.OutOfMemoryError: PermGen space
61,05%	Using multiprocessing pool of workers
59,50%	Unable to locate tools.jar
67,44%	How to properly put JSPs in the WEB-INF folder?
53,61%	Remove empty strings from a list of strings
59,27%	How do I access ModelMap in a jsp?
59,90%	Case insensitive replace
58,88%	Bypass GeneratedValue in Hibernate
70,61%	How can I convert this string to list of lists?
68,54%	Website to computer communications

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht.

Nicolas Schlicht

Köln, den 1. Juli 2020