TECHNISCHE UNIVERSITÄT
CHEMNITZ

Identifikation von Frageduplikaten für Q&A-Portale mittels Verfahren des maschinellen Lernens

Bachelorarbeit

Fakultät für Informatik
Professur Medieninformatik

Eingereicht von Stefan Taubert
Matrikelnummer: 369897
Chemnitz, den 21. Dezember 2017

Prüfer: Prof. Dr. Maximilian Eibl
Betreuer: M. Sc. Stefan Kahl

Taubert, Stefan
Identifikation von Frageduplikaten für Q&A-Portale mittels Verfahren des maschinellen Lernens
Bachelorarbeit, Fakultät für Informatik
Technische Universität Chemnitz, Dezember 2017

Zusammenfassung

Die vorliegende Bachelorarbeit gibt einen Einblick in bestehende und moderne Methoden der Identifikation ähnlicher Fragen. Auf Q&A Plattformen wie Quora werden täglich tausende Fragen gestellt und beantwortet. Dabei kommt es häufig vor, dass bereits gestellte Fragen in etwas abgewandelter Form erneut gestellt werden. Neue computerlinguistische Ansätze sollen durch klassische Textverarbeitung und maschinelles Lernen den Computer befähigen, ähnliche Fragen zu erkennen. Die gefundenen Methoden werden anhand von klassifizierten englischen Fragepaaren evaluiert.

Alle benutzten Internetquellen wurden am 21. Dezember 2017 zum letzten Mal auf Aktualität geprüft.

Inhaltsverzeichnis

Inhaltsverzeichnis . . . . . **i**

Abbildungsverzeichnis . . . . . **iv**

Tabellenverzeichnis . . . . . **v**

1 Motivation **1**
1.1 Kaggle als Online-Plattform 2
1.1.1 Aufbau einer Competition 2
1.1.2 Zur Verfügung gestellte Daten 2
1.1.3 Leaderboard 3
1.2 Quora Competition 4

2 Analyse der Daten **5**
2.1 Untersuchen der Trainingsdaten 5
2.1.1 Auszug aus den Duplikaten 6
2.1.2 Auszug aus den Nicht-Duplikaten 8
2.1.3 Auszug aus falschen Klassifizierungen 10
2.2 Untersuchen der Testdaten 11
2.2.1 Auszug aus den Duplikaten 12
2.2.2 Auszug aus den Nicht-Duplikaten 13
2.2.3 Auszug aus unüblichen Nicht-Duplikaten 13
2.3 Vergleich von Test- und Trainingsset 14

3 Grundlagen der Computerlinguistik **16**
3.1 Untersuchen der englischen Sprache 16
3.1.1 Morphologische Analyse 16
3.1.2 Aufbau englischer Sätze 17
3.2 Aspekte der syntaktischen Textvorverarbeitung 18
3.2.1 Segmentation und Tokenization 19
3.2.2 Part-Of-Speech Tagging 19
3.2.3 Normalization 20
3.2.4 Stemming 21
3.2.5 Lemmatization 21
3.2.6 Entfernen von Stoppwörtern 22
3.3 Grundlagen des Textvergleichs 22
3.3.1 Jaccard-Distanz 22

3.3.2 Levenshtein-Distanz . . . 23
3.3.3 Termfrequenz und Inverse Dokumentfrequenz . . . 23
3.4 Maschinelles Lernen . . . 24
3.4.1 Überwachtes Lernen . . . 24
3.4.2 Überanpassung und Unteranpassung . . . 25

4 Entwicklung eines Verfahrens . . . 26
4.1 Python Entwicklungsumgebung . . . 26
4.2 Baseline-Implementierungen . . . 26
4.2.1 Equal-Comparer . . . 27
4.2.2 Unequal-Comparer . . . 28
4.2.3 Random-Comparer . . . 28
4.2.4 Mean-Comparer . . . 28
4.2.5 Zusammenfassung . . . 29
4.3 Schritte zu einem Modell . . . 29
4.3.1 Vorverarbeitung . . . 29
4.3.2 Feature Extraction . . . 31
4.3.3 Modell Training . . . 35
4.3.4 Prediction . . . 36
4.3.5 Nachverarbeitung . . . 36

5 Evaluation . . . 37
5.1 Evaluationsmetriken . . . 37
5.1.1 Accuracy . . . 38
5.1.2 Precision . . . 38
5.1.3 Recall . . . 38
5.1.4 F1 . . . 39
5.1.5 Log-Loss . . . 39
5.2 Hyperparameter Optimization . . . 41
5.2.1 Grid Search . . . 41
5.2.2 Random Search . . . 41
5.3 Durchführung . . . 42
5.3.1 Feature-Selection . . . 42
5.3.2 Erstellen des Validierungssets . . . 42
5.3.3 Definieren der Scopes . . . 42
5.3.4 Ablauf . . . 43
5.4 Ergebnisse . . . 44
5.4.1 Baseline-Versuche . . . 44
5.4.2 Basisversuch . . . 45
5.4.3 Optimieren des Parameters num_boost_rounds . . . 47
5.4.4 Optimieren des Parameters rounding_boundary . . . 47
5.4.5 Optimieren der Tokenization . . . 49
5.4.6 Optimieren des Parameters max_depth . . . 50
5.4.7 Optimieren durch Normalization . . . 51

5.4.8 Finales Modell 52
5.5 Auszug aus den klassifizierten Validierungsdaten 57
5.5.1 Richtig vorhergesagte Fragepaare 57
5.5.2 Falsch vorhergesagte Fragepaare 57
5.5.3 Falsch klassifizierte Fragepaare 58
5.6 Fazit 59
5.6.1 Weitere Möglichkeiten 59

6 Ausblick 60

Literaturverzeichnis 61

Abbildungsverzeichnis

1.1 Öffentliches Leaderboard 3

2.1 Histogramm für die Auftrittswahrscheinlichkeit der Zeichen pro Frage 14
2.2 Histogramm für die Anzahl an Wörtern pro Frage 14
2.3 Wordcloud für das Trainingsset 15
2.4 Wordcloud für das Testset 15

3.1 Veranschaulichung von Überanpassung und Unteranpassung 25

4.1 Klassenstruktur der Baseline-Implementierungen 27

5.1 Diagramm der Log-Loss Funktion 40
5.2 Schematischer Ablauf der Evaluation 43
5.3 Ergebnisse der rounding_boundary-Versuche 48
5.4 Ergebnisse der Tokenization-Versuche 50
5.5 Ergebnisse aller Metriken der max_depth-Versuche 51
5.6 Ergebnisse der Normalization-Versuche 52
5.7 Ergebnisse aller Metriken des finalen Versuchs 54
5.8 Extrema der Accuracy für Anzahl an Features im finalen Versuch . . 54
5.9 Accuracies für alle Scopes des finalen Versuchs 55
5.10 Log-Loss für den finalen Versuch im Vergleich 55
5.11 Feature Prioritäten des finalen Versuchs 56

Tabellenverzeichnis

2.1 Satzanzahl im Trainingsset 6
2.2 Auszug aus Duplikaten des Testset 12
2.3 Auszug aus Nicht-Duplikaten des Testset 13
2.4 Auszug aus unüblichen Nicht-Duplikaten des Testset 13

3.1 Beispielfragen mit Kennzeichnung des Kopfes 17
3.2 POS-Tags des Stanford Taggers für eine Beispielfrage 20

4.1 Ergebnisse der Baseline-Implementierungen 29
4.2 Beispiele für die Normalization von Token 30
4.3 Parameter von XGBoost 35

5.1 Ergebnisse der Baseline-Versuche 44
5.2 Einstellungen der Hyperparameter im Basisversuch 45
5.3 Beste Ergebnisse aller Metriken für den Basisversuch 46
5.4 Beste Durchgänge aller Scopes im Basisversuch 46
5.5 Varianten der num_boost_rounds-Versuche 47
5.6 Varianten der rounding_boundary-Versuche 48
5.7 Varianten der Tokenization-Versuche 49
5.8 Varianten der max_depth-Versuche 50
5.9 Varianten der Normalization-Versuche 51
5.10 Beste Durchgänge des finalen Versuchs 53

Abkürzungsverzeichnis

FN False Negative

FP False Positive

IDF Inverse Dokumentfrequenz

POS Part-Of-Speech

TF Term Frequenz

TN True Negative

TP True Positive

1 Motivation

Auf Q&A–Portalen wie Quora[1], Yahoo! Anwers, Ask.fm, WikiAnswers oder Stack Exchange werden täglich tausende Fragen gestellt und beantwortet. Von den bis jetzt über 13 Millionen Fragen auf Quora (Stand März 2017)[2] wurden zirka 6 Millionen innerhalb des letzten Jahres gestellt. Das sind mehr als 16.000 neue Fragen pro Tag. Bei dieser Menge an Fragen ist es sehr wahrscheinlich, dass jemand eine Frage stellen will, die bereits in ähnlicher oder etwas abgewandelter Weise gestellt wurde. Durch das Finden neuer computerlinguistischer Ansätze, unter Anderem mit Hilfe von neuronalen Netzen, sollen ähnliche Fragen erkannt werden.
Durch einen Algorithmus, der zuverlässig Frageduplikate erkennt, können besonders Q&A–Portale profitieren. Der Vorteil an einer computergestützte Lösung liegt auf der Hand. Ohne einen zuverlässigen Algorithmus würde das Portal mit unzähligen Frageduplikaten überfüllt sein. Aus den redundanten Datenbankeinträgen resultiert ein erhöhter Speicherbedarf und höhere Speicherkosten. Alternativ dazu müssten die 16.000 Fragen pro Tag von Mitarbeitern auf Duplikate untersucht werden. Doch die Menge an Personen, die dafür nötig ist, würde die Personalkosten enorm in die Höhe treiben.
Auch Fragesteller können profitieren, da sie ihre Frage im Optimalfall gar nicht erst stellen müssen, wenn dies bereits geschehen ist. Nachdem eine Frage formuliert wurde, listet das Portal ähnliche Fragen auf, damit der Fragesteller schon Antworten finden kann. Ebenfalls Antwortsuchende, die keine Frage stellen wollen, erhalten bessere Antworten, da sich bei einzigartigen Fragen alle Meinungen an einer Stelle treffen. In einem großen Antwortpool kann eine tiefgründigere Diskussion der Antwortgeber über die beste Antwort stattfinden. Diese können sich so auf jede Frage konzentrieren und müssen nicht bei mehreren Fragen die gleiche Antwort geben.
Eine weitere Zielgruppe sind Data Scientists. Sie arbeiten täglich mit großer Datenmengen und wenden komplexe Algorithmen an, um Informationen zu extrahieren. Fast immer werden Methoden des maschinellen Lernens benutzt, um Modelle zu trainieren, wobei die Textverarbeitung dabei häufig eine große Rolle spielt. Die Fragenanalyse basiert auf textuellen Methoden und neue Verfahren könnten von Data Scientists auch in anderen Bereichen nützlich eingesetzt werden.
Um die Entwicklung eines Verfahrens zur Erkennung von Frageduplikaten soll es in dieser Arbeit gehen. Die dafür benötigten Daten sind auf Kaggle[3] gehostet und wurden von Quora zur Verfügung gestellt. Kaggle ist eine von Data Scientists geführte Internetseite, die in den folgendem Abschnitt kurz beschrieben wird.

[1] `http://quora.com`
[2] `http://quora.com/How-many-questions-are-on-Quora-answered-or-not`
[3] `http://kaggle.com`

1.1 Kaggle als Online-Plattform

Kaggle ist eine Online-Plattform, die Wettbewerbe, sogenannte Competitions, für prädiktive Modelle oder analytische Informationssysteme hostet. Jeder kann an diesen Competitions teilnehmen oder solche erstellen. Die für eine Competition benötigten Daten müssen der Community zur Verfügung gestellt werden. Kaggle richtet sich vorwiegend an Data Scientists, Machine Learning Engineers, Statistiker, Studenten oder Data Miner. Am Ende einer Competition muss jeder Teilnehmer ein Modell liefern und derjenige Teilnehmer, der das zuverlässigste Modell liefert, gewinnt. Neben den Competitions ist es möglich, auf Datensets und Kernels zuzugreifen oder Diskussionen zu führen. Diese Funktionalitäten sollen an der Stelle jedoch nicht weiter vertieft werden.

1.1.1 Aufbau einer Competition

Zu jeder Competition wird eine Beschreibung geliefert, die das zu lösende Problem erläutert und, wenn vorhanden, die aktuelle, zu verbessernde Lösung des Problems aufzeigt. Die Daten, mit denen gearbeitet werden soll, können separat heruntergeladen werden. Damit die Veranstalter der Competition einschätzen können, wie gut oder schlecht ein Algorithmus funktioniert, wird eine Evaluationsmetrik gegeben. Diese besteht aus einem Verfahren, welches die abgegebenen Daten der Teilnehmer mit den Idealdaten vergleicht (mehr dazu im Kapitel 5).
Da es sich bei den Veranstaltern der Competitions meist um Unternehmen handelt, die ein großes Interesse an einer guten Lösung haben, setzen viele von ihnen ein Preisgeld für die ersten Plätze aus. Diese können sogar im 6-stelligen Bereich liegen. Jede Competition hat zudem eine festgelegte Timeline, bis zu welcher der fertige Algorithmus vorliegen muss. Diese Zeitspanne kann jedoch bis zu 3 Jahre betragen.

1.1.2 Zur Verfügung gestellte Daten

Die bereits angesprochenen Daten, die von den Veranstaltern der Competitions zur Verfügung gestellt werden, können sich aus folgenden Daten zusammensetzen:

- **Testset**: Das Testset enthält die Daten, auf die das zu entwerfende Verfahren angewandt werden soll. Die Ideallösung zu den Testdaten besitzt lediglich der Veranstalter der Competition. Diese Daten werden außerdem in zwei Untermengen aufgeteilt: Zum einen das private Testset, in welchem sich der größere Anteil aller Testdaten befindet und zum anderen das öffentliche Testset. Der Grund für diese Aufteilung ist es, zu verhindern, dass der Algorithmus zu stark an das öffentliche Datenset angepasst wird. In solchen Fällen wird von Überanpassung gesprochen, bei welchen der Algorithmus zwar gut auf das öffentliche Testset anwendbar ist, aber im privaten Testset versagt (mehr dazu später in 3.4.2). Das private Testset wird jedoch erst veröffentlicht, wenn die Competition vorbei ist, um die Gewinne angemessen zu verteilen.

- **Trainingsset**: Das Trainingsset wird üblicherweise aus dem Testset abgespalten und enthält die Ergebnisse beziehungsweise Ausgaben für die gegebenen Dateneingaben. Es bietet die Grundlage zur Entwicklung eines Verfahrens, da mit diesem Trainingsset die generierten Ausgaben evaluiert werden. Es ist zudem nötig für das überwachte Lernen (siehe Kapitel 3.4.1) und wurde entweder teilweise computergestützt oder manuell erstellt.
- **Validierungsset**: Das Validierungsset ist ein, wie bei dem Trainingsset, mit Ausgaben markiertes Datenset. Es dient zur Evaluierung des entworfenen Verfahrens. Dieses Validierungsset wird aus dem Trainingsset abgespalten, wenn es nicht separat gegeben ist. Üblich ist es, 30% des Trainingssets nicht zum Trainieren des Algorithmus, sondern für das Validierungsset zu verwenden.

Damit eine Evaluation stattfinden kann, muss eine Datei erstellt werden, die die Ergebnisse des Algorithmus enthält. Diese Datei wird **Submission-File** genannt und beinhaltet die generierten Ausgabedaten zu allen Eingabedaten.

1.1.3 Leaderboard

Nachdem ein Teilnehmer ihr generiertes Submission-File erzeugt hat, wird sein Score durch Anwenden des Evaluationsverfahrens auf das Testset berechnet. Das Leaderboard listet die erreichten Scores aller Teilnehmer auf, damit jeder Teilnehmer seine Position in der Tabelle ablesen kann und sieht wie seine Konkurrenten bis zu diesem Zeitpunkt abschneiden. Dabei wird jeweils der beste Score aller hochgeladenen Submission-Files gewertet.
Ist die Competition vorbei, werden die Ergebnisse sowohl für das öffentliche, als auch für das private Testset angezeigt. Dabei kann es passieren, dass sich die Position der Teilnehmer verschiebt. Der Grund für die Verschiebung ist, dass die Teilnehmer die Parameter des Algorithmus nicht für das private Testset optimieren können. Somit wird sichergestellt, dass am Ende einer Competition universale Lösungen gefunden werden.

#	△priv	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	DL guys			0.11277	263	25d
2	—	Depp Learning			0.11367	196	25d
3	—	Jared Turkewitz & sjv			0.11446	178	25d
4	—	YesOfCourse			0.11450	189	25d
5	—	Qingchen | KazAnova | Faron			0.11482	219	25d

Abbildung 1.1: Das öffentliche Leaderboard zeigt für jedes Team die Platzierung und das Ergebnis der Evaluation (Score) an. Zusätzlich kann eingesehen werden, wann die letzte Submission war und wie viele Submissions ein Team abgegeben hat.

1.2 Quora Competition

Die Datensets, die für das Entwickeln eines Verfahren zur Frageduplikaterkennung nötig sind, liefert das Unternehmen Quora in ihrer Competition[4] auf Kaggle. Diese Competition hatte eine Timeline von 3 Monaten und startete am 16. März 2017. Für die ersten Plätze wurde ein Preisgeld von insgesamt 25.000 Dollar vergeben. Ziel dieser Competition war das gleiche Ziel dieser Bachelorarbeit, wobei Quora bereits ein Verfahren zur Duplikaterkennung entwickelt hatte. Dieses alte Verfahren sollte durch ein besseres neues Verfahren ersetzt werden. Die zur Entwicklung benötigten Daten liefert Quora in Form eines Test- und Trainingssets (siehe oben), die englische Fragepaaren enthalten. Die Aufteilung der Testdaten beträgt 35% und 65% zwischen öffentlichen und privaten Daten. Ein separates Validierungsset wird nicht geliefert, sodass 10% der Trainingsdaten entfernt und zum Validieren benutzt werden.
Das Submission-File besteht aus folgenden Spalten:

- **test_id**: Das Feld test_id enthält den eindeutigen Bezeichner für das jeweilige Fragepaar aus dem Testset.
- **is_duplicate**: Das is_duplicate-Feld enthält den Wert für das Fragepaar mit der jeweiligen test_id. Es muss im Intervall [0,1] liegen.

Insgesamt nahmen 3307 Teams an der Competition teil, wobei ein Team auch aus einer Person bestehen konnte.

Herangehensweise

Die Herangehensweise zur Bearbeitung eines solchen Themas besteht üblicherweise aus den Schritten Datenanalyse, Verfahrensentwicklung und Evaluation. Auch in dieser Bachelorarbeit werden die einzelnen Etappen durchlaufen. Anhand der Vorangehensweise soll der Leser einen Überblick bekommen, welche Schritte bei der Lösung von Klassifizierungsproblemen nacheinander angewandt werden, um später selbst an einer Competition teilnehmen zu können. Die Bearbeitung des Themas erfolgt dabei unter Zuhilfenahme von Methoden des maschinellen Lernens, welche heutzutage in jeder Competition Verwendung finden. Dabei soll jedoch der Fokus auf klassischen Textverfahren liegen, um zu zeigen, dass es nicht immer nötig ist, auf komplexere Methoden, wie beispielsweise neuronale Netze, zurückzugreifen.

[4] `http://kaggle.com/c/quora-question-pairs`

2 Analyse der Daten

Die Grundlage für die Entwicklung eines Verfahrens zur Identifikation von Fragepaaren bilden bereits identifizierte Fragepaare. Anhand dieser Fragepaare kann das Verfahren später evaluiert werden. Quora liefert in ihrer Competition zwei Datensätze im CSV–Format, welche die Test- beziehungsweise Trainingsdaten in Form von Fragepaaren enthalten. Die Analyse der Daten spielt eine zentrale Rolle bei klassischen Klassifizierungsproblemen und wird in diesem Kapitel beschrieben.

2.1 Untersuchen der Trainingsdaten

Das Trainingsset enthält zu jedem Fragepaar einen Wert, der angibt, ob es sich bei dem Fragepaar um ein Duplikat handelt. Es liegt als CSV-Datei vor und setzt sich aus folgenden Spalten zusammen:

- **id**: Diese Spalte stellt einen eindeutigen Bezeichner für das jeweilige Fragepaar dar.
- **qid1** und **qid2**: Jede Frage bekommt zusätzlich einen eindeutigen Bezeichner, da bestimmte Fragen mehrfach mit anderen Fragen im Trainingsset als Fragepaar kombiniert vorkommen.
- **question1** und **question2**: In diesen Spalten befinden sich die entsprechenden Fragen, welche auch aus mehreren Sätzen bestehen können.
- **is_duplicate**: Dieser Wert gibt an, ob das Fragepaar mit der **id** ein Duplikat ist. Der Wert „0“ bedeutet, dass das Paar kein Duplikat ist und der Wert „1“ bedeutet, dass es sich bei dem Paar um ein Duplikat handelt.

Insgesamt besteht das Set aus 404.290 Fragepaaren, von denen 36,92% Duplikate sind. Das heißt es handelt sich nur bei rund einem Drittel aller Paare um ein Duplikat. Das Trainingsset ist demnach etwas unbalanciert. Diese ungleiche Verteilung muss später beachtet werden. Weiterhin treten 111.780 der in Summe 537.933 Fragen, also fast jede fünfte Frage, mehrfach auf. Die drei häufigsten Fragen im Trainingsset sind:

- What are the best ways to lose weight?
- How can you look at someone's private Instagram account without following them?
- How can I lose weight quickly?

Wird die Anzahl an Sätzen im Trainingsset betrachtet, fällt auf, dass über 90% aller Fragen aus nur einem Satz bestehen und weniger als 8% zwei Sätze haben. Daraus lässt sich schlussfolgern, dass der Hauptfokus bei der Verfahrensentwicklung auf Fragen mit einem Satz gelegt werden sollte:

Tabelle 2.1: Die meisten Fragen des Trainingssets bestehen aus einem Satz. Fragen mit mehren Sätzen sind selten, weswegen der Hauptfokus bei der Verfahrensentwicklung auf Fragen mit einem Satz gelegt wird.

Satzanzahl	**Häufigkeit %**
1	90,54
2	7,71
3	1,38
4-22	<0,27

Um einen Überblick zu bekommen, welche Fragen sich im Trainingsset befinden, wurden einige repräsentative Fragepaare herausgesucht. Im Folgenden werden zuerst Duplikate, anschließend Nicht-Duplikate und zuletzt Fragepaare, die von Quora falsch klassifiziert wurden, genauer betrachtet.

2.1.1 Auszug aus den Duplikaten

Nachfolgend sind Duplikate mit verschiedener Anzahl selber Worten aufgelistet, wobei ein Fragepaar umso weniger gemeinsame Worte besitzt, desto weiter unten es steht. Die Schwierigkeit des zu entwerfenden Verfahrens liegt darin, anhand von Wörtern, die nur in einer der beiden Fragen vorkommen, zu erschließen, ob es sich um ein Duplikat handelt.

Fragepaar 81640

- Frage 1: Why we should stop animal abuse?
- Frage 2: Why should we stop animal abuse?

Das erste Fragepaar enthält die gleichen Wörter und hat die gleiche Anzahl an Worten. Damit ist es sehr wahrscheinlich, dass dieses Paar ein Duplikat ist. Es ändert sich die Bedeutung der Fragen nicht, da nur Subjekt und Verb vertauscht sind. Trotzdem muss die gleiche Anzahl und Verwendung von Worten nicht zwangsläufig zu einem Duplikt führen, wie sich später noch zeigt.

Fragepaar 389453

- Frage 1: I can't lucid dream.Why?
- Frage 2: Why is lucid dreaming so difficult?

Im zweiten Fragepaar wird eine Frage in zwei Sätze aufgeteilt. Da jedoch das Fragewort alleine als Satz steht, könnte dieses später einfach vor den ersten Satz gesetzt werden, sodass sich „Why I can't lucid dream?" entsteht. Damit lassen sich beide Fragen besser vergleichen. Es ist einfach herauszufinden, dass es um „lucid dreaming" geht. Etwas schwerer wird es den Teilsatz „I can't" mit „so difficult" zu vergleichen, daher gehört dieses Fragepaar zu den etwas schwierigeren Paaren.

Fragepaar 58

- Frage 1: I was suddenly logged off Gmail. I can't remember my Gmail password and just realized the recovery email is no longer alive. What can I do?
- Frage 2: I can't remember my Gmail password or my recovery email. How can I recover my e-mail?

Das nächste Duplikat besteht jeweils aus mehreren langen Sätzen, in denen die Frage formuliert wird. Die Fragen am Ende haben jedoch keine Ähnlichkeit: „What can I do?" und „How can I recover my e-mail?". Somit können nicht einfach nur die Fragesätze der Frage auf Gleichheit geprüft werden. Frage 1 erklärt den Sachverhalt in zwei Sätzen, wobei in Frage 2 der Sachverhalt nur mit einem Satz erklärt wird. Der zweite Satz von Frage 1 und der erste Satz von Frage 2 haben jedoch eine hohe Ähnlichkeit. Ein guter Algorithmus könnte später herausfinden, welcher der angegebenen Sätze einer Frage am relevantesten ist und diesen jeweils miteinander vergleichen. Da Fragen, die mehr als einen Satz haben aber zu der Unterzahl gehören, werden solche Fragepaare nicht genauer untersucht.

Fragepaar 62

- Frage 1: How is the new Harry Potter book „Harry Potter and the Cursed Child"?
- Frage 2: How bad is the new book by J.K Rowling?

Das Fragepaar mit der Id 62 setzt ein gewisses Allgemeinwissen voraus, denn der Algorithmus müsste wissen, dass J.K Rowling unter anderem die Harry Potter Bände geschrieben hat und, dass der in der ersten Frage genannte Band der letzte Band ist. Da schätzungsweise nach Harry Potter und J.K. Rowling eher selten gefragt wird, haben diese Wörter mehr Relevanz gegenüber häufigeren Wörtern (siehe 3.3.3). Dadurch wird dieses Paar selbst von einem guten Algorithmus höchstwahrscheinlich als Nicht-Duplikat erkannt.

Fragepaar 71716

- Frage 1: How do parents feel when their kid has depression?
- Frage 2: What is it like to have a child with depression?

Ein weiteres schwieriges Fragepaar ist zudem das letzte Paar, da es zwischen beiden Fragen nur ein gemeinsames Wort gibt. In Frage 1 geht es eher darum wie sich die Eltern fühlen, wenn sie ein Kind mit Depressionen haben und im zweiten Satz eher darum, wie es ist ein Kind mit Depressionen zu haben. Der Algorithmus müsste erkennen, dass Eltern mindestens ein Kind haben und diese in der zweiten Frage gemeint sind. So ein Sachverhalt könnte sich für ein Verfahren als sehr schwierig bis unmöglich herausstellen.

2.1.2 Auszug aus den Nicht-Duplikaten

Neben den Duplikaten existieren ebenfalls Nicht-Duplikate. Sie bilden die Mehrheit des Trainingssets und werden im Folgenden genauer untersucht. Hierbei vermehrt sich die Anzahl an gemeinsamen Worten, je weiter unten das Fragepaar beschrieben wird, da Paare mit mehreren gleichen Worten schwerer zu unterscheiden sind.

Fragepaar 23

- Frage 1: How much is 30 kV in HP?
- Frage 2: Where can I find a conversion chart for CC to horsepower?

Die Fragen dieses Paars haben kaum Gemeinsamkeiten, weder in der Satzlänge, noch in der Anzahl an gemeinsamen Worten. Daher sollte es sehr einfach als Nicht-Duplikat festzustellen sein.

Fragepaar 111

- Frage 1: Is USA the most powerful country of the world?
- Frage 2: *Why* is the USA the most powerful country of the world?

Bei diesem Nicht-Duplikat, wird die erste Frage in der zweiten Frage mit einem „Why“ ergänzt und erhält damit einen anderen Sinn. Die Fragen sind sich jedoch ansonsten sehr ähnlich und werden schwer als Nicht-Duplikat zu erkennen sein.

Fragepaar 64

- Frage 1: Where can I find a *European* family office database?
- Frage 2: Where do I find a *U.S.* family office database?

In diesem Paar unterscheiden sich die Fragen durch ein Wort und ergeben damit einen anderen Sinn. Dieses Paar sollte nicht so einfach zu markieren sein, da die Anzahl an Wörtern gleich ist und die Sätze sich stark ähneln. Das Entfernen von Stoppwörtern wie „can“ oder „do“ könnte sich gegebenenfalls negativ auswirken, da die Anzahl an unterschiedlichen Wörtern dadurch abnimmt (später mehr in 3.2.6).

Fragepaar 22

- Frage 1: What are the questions should *not* ask on Quora?
- Frage 2: Which question should I ask on Quora?

Bei dem vierten Fragepaar wird die erste Frage in der zweiten Frage durch ein „not“ negiert und ergibt damit einen komplett anderen Sinn. Das Entfernen von Stoppwörtern (siehe 3.2.6) hat ebenfalls eine negative Auswirkung, denn ohne „not“ geht der Sinn der Frage verloren. Eine Lösung könnte sein, bei kurze Fragen die Stoppwörter nicht zu entfernen.

Fragepaar 195

- Frage 1: What is the *work of* an executive recruiter like?
- Frage 2: What is it like to *work with* an executive recruiter?

Das nächste Paar ist etwas kniffelig, da sich die Präposition nach dem Hauptverb in beiden Fragen unterscheidet. Der Algorithmus muss erkennen, dass „work of“ einen anderen Sinn als „work with“ ergibt. Dafür muss er erst die Satzstruktur verstehen und solche Ansätze können sehr aufwendig sein, da jedes Wort nicht immer die den gleichen POS-Tag besitzt (mehr dazu in 3.2.2).

Fragepaar 21

- Frage 1: What's *causing someone* to be jealous?
- Frage 2: What can I do to *avoid being* jealous of someone?

Das vorletzte Fragepaar ist für den Menschen möglicherweise schwerer als Nicht-Duplikat zu erkennen als für den Computer. Da wenig Wörter gemeinsam vorkommen, wird der Algorithmus diese Frage eher als Nicht-Duplikat klassifizieren.

Fragepaar 6279

- Frage 1: Why is Google Chrome not working but Internet Explorer is?
- Frage 2: Why is Internet Explorer not working but Google Chrome is?

Das letzte Nicht-Duplikat enthält die gleichen Wörter, die gleiche Wortanzahl, jedoch nicht die gleiche Reihfolge der Wörter. Dies macht einen großen Unterschied, denn in der ersten Frage möchte der Fragesteller wissen, wieso der Internet Explorer nicht funktioniert und in der zweiten Frage aus welchem Grund Google Chrome nicht funktioniert. Das Paar wird schwer als Nicht-Duplikat zu erkennen sein.

2.1.3 Auszug aus falschen Klassifizierungen

Unter den Trainingspaaren existieren Duplikate beziehungsweise Nicht-Duplikate mit falschen Klassifizierungen. Das heißt es wurden im Trainingsset Duplikate als Nicht-Duplikate markiert und andersherum. Solche falschen Markierungen verfälschen die Evaluation und erschweren das maschinelle Lernen, es ist aber zu aufwändig, alle Fragen des Trainingssets auf ihre Richtigkeit zu prüfen. Einige dieser Fragepaare sind nachfolgend dargelegt.

Fragepaar 284

- Frage 1: How can I make money online with free of cost?
- Frage 2: How do I to make money online?
- Duplikat: ja

Das Fragepaar mit der Id 284 ist als Duplikat markiert, in Frage 1 ist jedoch die Spezifizierung „with free of costs“ vorhanden. Demnach ist dieses Paar kein Duplikat.

Fragepaar 365766

- Frage 1: What are some of the best romantic movies in English?
- Frage 2: What is the best romantic movie you have ever seen?
- Duplikat: ja

Bei diesem Fragepaar wird in der ersten Frage nach englischen romantische Filmen gesucht. In der zweiten Frage geht es jedoch allgemein um romantische Filme. Die Spezifizierung wurde nicht erkannt und deswegen ist dieses Paar als Duplikat markiert.

Fragepaar 172557

- Frage 1: What will be the output of the following code?
- Frage 2: What will be the output of following c++ code?
- Duplikat: nein

Das dritte Paar fragt nach der Ausgabe eines Programmcodes. In der zweiten Frage wird dieser auf C++ spezifiziert, jedoch ist die Sprache ebenfalls an dem Quellcode erkennbar. Demnach sind beide Fragen auf C++ spezifiziert, wenn der Quellcode in C++ gehalten ist. Da dieser Quellcode jedoch nicht verfügbar ist, kann dieses Fragepaar nicht eindeutig klassifiziert werden. Um effektiv mit den Daten arbeiten zu können, sollten solche Fragepaare im Trainingsset und im Testset nicht vorkommen.

Fragepaar 16682

- Frage 1: Why are natural resources important for us?
- Frage 2: Why natural resources are important for us?
- Duplikat: nein

Bei dem nächsten Duplikat sind nur Subjektiv und Verb vertauscht, der Sinn der Frage bleibt jedoch erhalten. Dieses Paar ist fälschlicherweise als Nicht-Duplikat markiert und wurde möglicherweise mit dem alten Verfahren von Quora klassifiziert. Es lässt sich vermuten, dass noch mehr solcher falscher Duplikate existieren.

Fragepaar 359464

- Frage 1: Which is better PS4 or Xbox one?
- Frage 2: Which has better exclusives, PS4 or Xbox One?
- Duplikat: ja

Im vorletzten Fragepaar wird in der ersten Frage nach der besseren Konsole gefragt. Die zweite Frage zielt jedoch auf die exklusiven Titel der jeweiligen Konsole ab. Damit handelt es sich, anders als angegeben, um kein Duplikat.

Fragepaar 34646

- Frage 1: Is the sun alive? Why or why not?
- Frage 2: Are stars alive?
- Duplikat: ja

Das letzte Fragepaar ist als Duplikat markiert, obwohl in der ersten Frage nach der Lebendigkeit von Sternen in der zweiten Frage nach der Lebendigkeit der Sonne gefragt wird. Außerdem möchte der Fragesteller von Frage 1 zusätzlich die Ursache wissen.

2.2 Untersuchen der Testdaten

Das Testset enthält die Daten, die der zu entwickelnde Algorithmus markieren soll und an welchem der Leaderboard Score errechnet wird. Die Markierungen für die Fragen werden nicht veröffentlicht, sondern nur von Quora selbst verwaltet. Das öffentliche Set enthält 2.345.796 Fragepaare. Das ist fast die 6-fache Menge an Fragepaaren der Trainingsdaten. Auf Grund dessen, dass das öffentliche Testset nur 35% der Daten vom gesamten Testset enthält, kann die Menge an privaten Test-Fragepaaren berechnet werden. Es ergeben sich circa 4.356.479 weitere Fragepaare für das private Set. In der Summe handelt es sich somit um 6.702.275 Paare im kompletten Testset. Bei dieser Anzahl an Paaren kann sicher davon ausgegangen werden,

dass diese bei Quora mit dem alten Algorithmus markiert worden sind, denn diese Menge kann keine Person per Hand klassifizieren. Demnach kann Gleiches auch für das Trainingsset geschlussfolgert werden, da es höchstwahrscheinlich eine Teilmenge des Testsets ist. Wie sich später noch zeigt, sind eine nicht unbeachtliche Menge an Fragen generiert wurden.
In der CSV-Datei sind folgende Spalten definiert:

- **test_id**: Diese Spalte stellt einen eindeutigen Bezeichner für das jeweilige Fragepaar dar.
- **question1** und **question2**: In diesen Spalten befinden sich die unmarkierten Fragen, welche jedoch keiner eigenständigen Id zugewiesen sind.

Da es sich hierbei um Testdaten handelt, fehlt die is_duplicate-Klassifizierung und muss vom Algorithmus eigenständig generiert werden. In der Zieldatei müssen alle test_ids und ihre Klassifizierungen in der Spalte is_duplicate enthalten sein.
Nachfolgend wird ein kleiner Auszug aus den Fragepaaren des Testsets gegeben. Die Testdaten stellen jedoch eine unwichtige Rolle bei der Verfahrensentwicklung dar, da auf sie nur die von Kaggle vorgesehene Evaluations-Metrik angewandt werden kann. Aus diesem Grund sollen lediglich ein paar Fragepaare gezeigt werden. Die Hauptrolle spielt das Trainingsset, welches bereits im vorherigen Kapitel ausführlich beschrieben wurde.

2.2.1 Auszug aus den Duplikaten

Die Duplikate aus dem öffentlichen Testset sehen denen des Trainingssets sehr ähnlich. Ob diese Fragepaare tatsächlich als Duplikate klassifiziert sind, ist nicht erkennbar.

Tabelle 2.2: Ein Auszug aus den Duplikaten des Testsets zeigt, dass diese den Fragepaaren des Trainingssets sehr ähnlich sind und deutet darauf hin, dass beide Sets ursprünglich mit hoher Wahrscheinlichkeit einer gemeinsamen Menge angehörten.

Id	**Frage 1**	**Frage 2**
176	What is Economics?	What is the economics?
1415006	How do I my life happy?	What the best method to make life happy?
1415099	Who is the highest paid in Hollywood?	Who are the highest paid actors Hollywood right now?
1415133	How can I get into shape?	How can I get in shape quickly?
1415208	Where can I get real friends of opposite gender online?	What is the best way to make new friends with the opposite gender online?

2.2.2 Auszug aus den Nicht-Duplikaten

Bei den Nicht-Duplikaten des Testsets fällt ebenfalls kein Unterschied zu denen des Trainingsset auf, da zum Beispiel die Fragen den gleichen Themengebieten, wie Arbeit, Gesundheit oder Sport angehören.

Tabelle 2.3: Ein Auszug aus den Nicht-Duplikaten zeigt, dass die gleichen Themen wie im Trainingsset behandelt werden. Solche Themen sind beispielsweise Bildung (54), Arbeit (59), Beziehungen (242), Gesundheit (693047) oder Sport (1886411).

Id	Frage 1	Frage 2
54	Where do macadamia nuts cost from?	What the hangouts courses offered at AIIMS Bhopal?
59	How do I become a data scientist in Malaysia?	How can I become a data scientist?
242	What are best dating sites?	What are the top dating apps?
693047	Why are children so skinny?	Why are bikers skinny?
1886411	What was the worst Super Bowl halftime show?	Who pays for the Super Bowl halftime show?

2.2.3 Auszug aus unüblichen Nicht-Duplikaten

Interessant sind folgende Fragepaare, denn sie bestehen nicht aus richtigen Sätzen. Es scheint als seien die Paare durch Entfernen von Wörtern generiert wurden (siehe 58, 100, 330). Eine weitere Strategie war auch das zufällige Kombinieren von Stoppwörtern wie bei den Fragepaaren 330 und 1538055.

Tabelle 2.4: Einige Fragepaare des Testsets bestehen aus keinen grammatikalisch sinnvollen Sätzen oder setzen sich nur aus Stoppwörtern zusammen. Möglicherweise soll mit solchen Fragepaaren ein robusteres Verfahren entstehen.

Id	Frage 1	Frage 2
58	Why is glass a green in color?	What color say?
100	Which is the	Where was training founded?
330	Why movies?	Will I Why or why not?
1538055	How the world?	How

Da es sich jedoch bei diesen Fragepaaren immer um ein Nicht-Duplikat handelt, sollten diese zum Beispiel durch das Entfernen von Stoppwörtern schnell erkannt werden. Ein Grund dafür, dass solche Fragepaare bewusst eingefügt wurden, könnte sein, dass die Competition-Teilnehmer robustere Verfahren entwickeln sollen.

2.3 Vergleich von Test- und Trainingsset

Nachdem ein Überblick über die Fragen der beiden Sets verschafft wurde, sollen Test- und Trainingsset in dieser Sektion gegenübergestellt werden. Das erste Histogramm zeigt, wie häufig Fragen mit der jeweiligen Anzahl an Zeichen auftreten. Es lässt sich gut erkennen, dass die Mehrheit aller Fragen aus dem Test- und Trainingsset zwischen 30 und 70 Zeichen bestehen. Ab 150 Zeichen gibt es bei beiden Sets einen deutlichen Abfall an Fragen. Genauso sind Fragen unter 10 Zeichen sehr selten vertreten. Interessant wird es später zu sehen, wie gut der gefundene Algorithmus auf die einzelnen Fragelängen anzuwenden ist. Besonders für die Mehrheit aller Fragen sollte ein gutes Ergebnis erzielt werden.

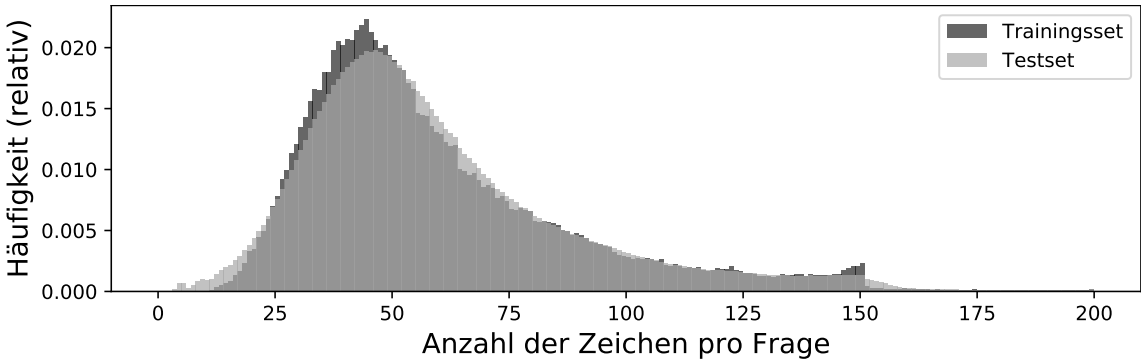


Abbildung 2.1: Das Histogramm für die Auftrittswahrscheinlichkeit der Zeichen pro Frage zeigt, dass in beiden Sets die meisten Fragen zwischen 30 und 70 Zeichen haben. Es kann ebenfalls geschlussfolgert werden, dass das Trainingsset das Testset sehr gut repräsentiert.

Werden Test- und Trainingsset in der Wortlänge verglichen, zeigt sich, dass die meisten Fragen aus 8 bis 12 Wörtern bestehen. Die wenigsten Fragen haben hingegen weniger als 3 oder mehr als 30 Wörter.

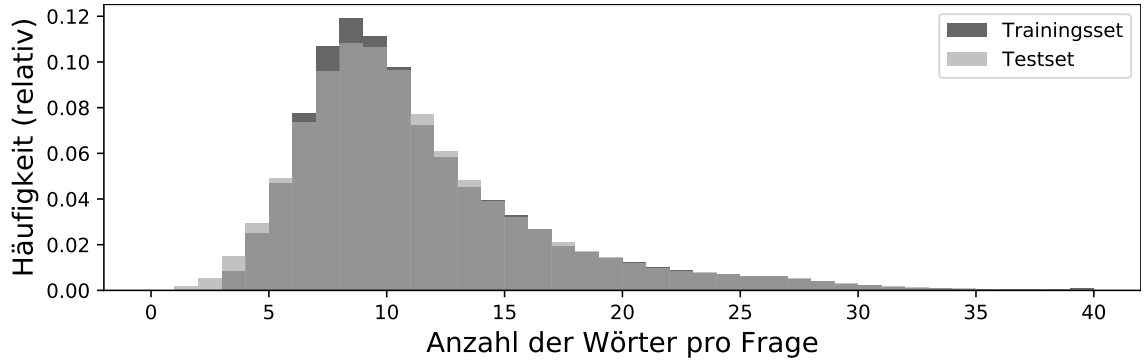


Abbildung 2.2: Das Histogramm für die Anzahl an Wörtern pro Frage lässt erkennen, dass die meisten Fragen zwischen 8 und 12 Wörter haben. Da beide Sets ähnliche Häufigkeiten besitzen, wird ein gutes Verfahren für das Trainingsset vermutlich ebenfalls gut für das Testset sein.

Nachfolgend sind Wordclouds für das Test- und Trainingsset zu sehen, welche die häufigsten Worte der Fragen darstellen. Dabei ist ein Wort umso häufiger, je größer es dargestellt ist. Besonders häufig sind dabei die Terme: „best way“, „difference“ und „will“. Sie spiegeln häufige Fragephrasen wie „What is the best way to. . . “ oder „What is the difference between. . . “ wieder. Außerdem zeigt die Wordcloud, dass die Fragen aus der heutigen Zeit stammen, da beispielsweise die Terme „Donald Trump“ und „Hillary Clinton“ vorkommen. Das Wort „India“ lässt suggerieren, dass viele Fragen aus Indien kommen und von Indern gestellt wurden. Die indische Währung „rupee“ taucht auch in der Wortcloud auf und unterstreicht diese Aussage.

Abbildung 2.3: Die Wordcloud für das Trainingsset zeigt, dass aktuelle Themen in den Fragen behandelt werden („Donald Trump“), dass viele Fragen aus Indien stammen („India“, „rupee“) und, dass klassische Fragen gestellt werden („best way“, „difference“).

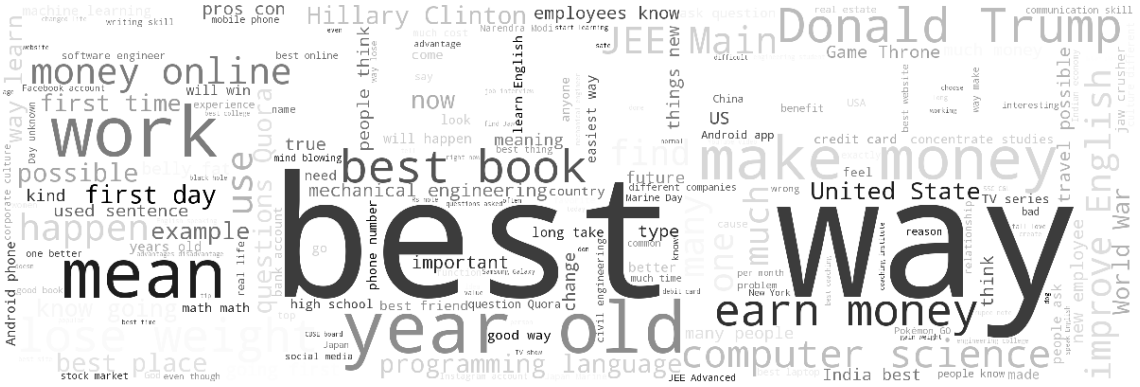


Abbildung 2.4: Die Wordcloud für das Testset hat viele gemeinsame Worte mit der Cloud des Trainingssets („best way“, „mean“, „money“) und dies spricht dafür, dass in beiden Sets Fragen mit gleichen Themen gestellt werden („Donald Trump“, „make money“, „lose weight“). Das Wort „difference“ taucht jedoch im Trainingsset viel häufiger auf.

Es kann schlussendlich behauptet werden, dass das Trainingsset mit großer Sicherheit aus dem Testset abgespalten wurde.

3 Grundlagen der Computerlinguistik

Dieses Kapitel beschäftigt sich mit den Grundlagen der Computerlinguistik, welche erforderlich sind, um Fragepaare zu identifizieren. Das erste Unterkapitel gibt einen kleine Einblick darüber, wie die englische Sprache aufgebaut ist. Das hilft später dabei zu verstehen, wieso ein Ansatz, verglichen mit einem anderen, besser funktioniert hat. Anschließend werden die wichtigsten Aspekte der syntaktischen Textvorverarbeitung dargelegt, deren Methoden heutzutage bei fast jeder computergestützten Textverarbeitung Einsatz finden. Diese Methoden reichen jedoch nicht aus, um Fragepaare eindeutig zu identifizieren, weshalb im Anschluss bekannte Textvergleichsverfahren erläutert werden. Das letzte Unterkapitel zeigt einen Einblick in das maschinelle Lernen, welches für den Textvergleich verwendet werden soll.

3.1 Untersuchen der englischen Sprache

Quora verwendet in ihren Datensets nur englische Fragen. Aus diesem Grund werden der Satzbau und einige Eigenschaften der englischen Sprache aufgezeigt, um zu verstehen, welche Wörter einer Frage maßgeblich für deren Inhalt sind. Über dieses Thema gibt es sehr viele Veröffentlichungen (zum Beispiel [All95, CEE+10, All95, Emo76, Mus16, Bre72]), die an der Stelle nicht näher betrachtet werden. Der Fokus dieser Arbeit soll auf klassischen Methoden der Textverarbeitung liegen. Das inhaltliche, computergestützte Verstehen eines Satzes erfordert aufwändige Verfahren, die wiederum viel Rechenkapazität benötigen. Diese Verfahren reichen jedoch über das Ziel der Bachelorarbeit hinaus.

3.1.1 Morphologische Analyse

Die Grundlage für einen Satz sind Wörter, welche unterschiedlich gebildet werden. Die Grundform eines Wortes wird **Morphem** genannt [All95, S. 10][CEE+10, S. 237]. So besteht zum Beispiel das zusammengesetzte englische Wort „policeman“ aus den beiden Morphemen „police“ und „man“.
Weiterhin können Wörter durch Bildung von Morphemen in **flektierte** und **abgeleitete** Wörter differenziert werden [CEE+10, S. 111]. Flektierte Wörter benutzen die Grundform eines Wortes und fügen meist eine Nachsilbe an. Im Englischen ist dies gut an Verben veranschaulichbar. Bei diesen werden oft Endungen wie „-s“, „-ing“ oder „-ed“ angehängt, um das Grundwort zu erweitern. Ein Beispiel ist das Verb „laugh“, welches zu den Wörtern „laughs“ oder „laughing“ umgewandelt werden kann. Jedes abgeleitete Wort hat dabei die gleiche Bedeutung wie das Grund-

wort. Für den Fragevergleich könnte es später vorteilhaft sein, von jedem Verb die Grundform zu bilden, um gleiche Bedeutungen herauszufinden.
Abgeleitete Wörter hingegen können in andere Wortformen umgewandelt werden und haben danach meist eine komplett andere Bedeutung [All95, S. 23]. Beispielsweise das Substantiv „day“ kann durch das Suffix „-ly“ zu „daily“ abgeleitet werden, erhält jedoch dadurch eine andere Bedeutung. Das Bilden des Wortstammes bei abgeleiteten Worten kann sich deswegen negativ auswirken.

3.1.2 Aufbau englischer Sätze

Linguisten unterscheiden vier verschiedene **Hauptklassen** in der englischen Sprache, welche die meiste Aussagekraft für den Bedeutung eines Satzes besitzen:

- Substantive,
- Adjektive,
- Verben und
- Adverben.

Jeder Satz enthält Vertreter dieser vier Hauptwortklassen. Neben ihnen existieren noch weitere Klassen wie Artikel, Pronomen, Präpositionen, Partikel, Konjunktionen et cetera. Diese Wortklassen sind jedoch in ihrer Wortbedeutung relativ fix, da selten neue Worte in diesen Klassen erschaffen werden. Aus diesem Grund werden solche Wortklassen auch **geschlossene Klassenwörter** genannt.
Die vier Hauptklassen auf der anderen Seite sind häufig im Wandel und erhalten neue Worte im Laufe der Entwicklung einer Sprache und werden deswegen als **offene Klassenwörter** bezeichnet. Ein Satz besteht im Englischen immer aus mindestens einer der offenen Hauptklassen. Jede Hauptklasse eines Satzes wird **Kopf** genannt. Diese Köpfe geben die Grundbedeutung des Satzes an und können im Fragevergleich als wichtigstes Element der Frage angesehen werden. Zusätzlich zum Kopf gibt es das **Komplement**, welches die Bedeutung des Satzes in Kombination mit dem Kopf erweitert. [All95, S. 24f.][Bak95, S. 41ff.][Cho02]
In nachstehender Tabelle sind die **Köpfe** einiger Fragen aus dem Trainingsset gekennzeichnet:

Tabelle 3.1: Ein Auszug aus den Fragen des Trainingssets zeigt, dass der Kopf der Frage maßgeblich für den Inhalt verantwortlich ist.

Id	Frage
442	Can I earn **money** on Quora?
444	What's the **PUK** for TF64SIMC4?
198592	How did you get **wealthy**?

Ferner kann ein Satz im Allgemeinen eine Behauptung, eine Frage oder ein Befehl sein. Bei Fragesätzen gibt es zwei sogenannte **Stimmungen** beziehungsweise Fragetypen [All95, S. 28]:

- Ja-/Nein-Fragen, die mit Ja oder Nein beantwortet werden können, wie „Can we time travel?“, „Are you satisfied with your life?“ oder „Are stars alive?“ und
- wh-Fragen, die nicht mit Ja oder Nein beantwortet werden können, wie „What do you think of online shopping?“, „Why are people not answering my questions on Quora?“ oder „Is the sun alive? Why or why not?“.

Ein möglicher Ansatz wäre es, den Fragetyp festzustellen, da es sich bei gleichen Typen innerhalb eines Fragepaares wahrscheinlicher um ein Duplikat handelt. Alternativ dazu können **Vorlagen** für Fragen definiert werden, die mit den gleichen Worten starten. Zum Beispiel könnte geschaut werden, ob beide Fragen mit „What is the“, „How do I“, „What are the“, „How can I“ oder „What are some“ beginnen. Enthalten beide Fragen eines Paares die gleiche Vorlage, könnte es sich mit höherer Wahrscheinlichkeit um ein Duplikat handeln.
Wie bereits erwähnt ist diese Art komplexer Methoden nicht Ziel dieser Arbeit, weswegen im nächsten Unterkapitel detailliert auf klassische Textverarbeitungs-Verfahren eingegangen wird.

3.2 Aspekte der syntaktischen Textvorverarbeitung

Die Syntax beschreibt die Prinzipien, wie Wörter in einer Frage miteinander kombiniert werden können und definiert, welche strukturelle Rolle ein Wort spielt. Zudem gibt sie an, welche Teilsätze in Sätzen vorkommen. Die syntaktische Zerlegung einer Frage ist nötig, um den Zusammenhang zwischen Wörtern einer Frage herauszufinden. Sie gibt an, wie Wörter gruppiert werden und sich gegenseitig beeinflussen. Zusätzlich kann die Syntax aussagen, wie bestimmte Satzteile miteinander in Beziehung stehen. [All95, S. 10ff.][Cho02, S. 1][Mit04, S. 758]
Ein Satz kann in seine syntaktischen Einzelteile zerlegt werden. Die Vorverarbeitung von Text durchläuft typischerweise nacheinander die folgenden Schritte:

1. Segmentation/Tokenization
2. Normalization
3. Stemming/Lemmatization
4. Entfernung von Stoppwörtern (optional)

Nachfolgend wird auf diese Schritte und deren Grundlagen eingegangen. Zur Veranschaulichung der einzelnen Verfahren soll folgende Beispiel-Frage dienen: „What is the most valuable thing you have?“.

3.2.1 Segmentation und Tokenization

Die Extraktion von sprachlich relevanten Einheiten eines Textes wird Segmentierung genannt. Diese Einheiten sind beispielsweise Sätze oder Wörter. Dadurch wird es möglich, die Texte anschließend mit Hilfe von informationstechnologischen Werkzeugen zu bearbeiten. [CEE+10, S. 264, S. 282]
Der Begriff Tokenization beschreibt den Prozess des Aufteilens eines Textes in seine Token. Ein Token stellt in diesem Zusammenhang ein Wort dar, also eine Anreihung alphanumerischer Zeichen, die beidseitig durch ein Leerzeichen oder Interpunktation von einander getrennt sind. Hierbei muss jedoch beachtet werden, dass unter anderem in Zahlen Punkte enthalten sein können, die nicht als Satzende bestimmt sind. Die geschrieben Zahl „2.000“ darf zum Beispiel nicht in die Token {2, ., 000} umgewandelt werden. [CEE+10, S. 264f][Bir06, S. 70][ID10, S. 15ff.]
Die Umsetzung solcher Verfahren geschieht durch Tokenizer. Ein bekannter Tokenizer ist der Stanford Tokenizer, welcher in der englischen Sprache als PTBTokenizer[1] bekannt ist. Er separiert, so wie die meisten Tokenizer, die Token eines Textes anhand von Leerzeichen und Interpunktion.

Beispiel

Folgendes Beispiel stellt die Tokenization der Beispiel-Frage dar und soll den Prozess verdeutlichen:
Original: „What is the most valuable thing you have?“
Tokenisiert: {What, is, the, most, valuable, thing, you, have, ?}
Wie zu sehen ist, wird das Fragezeichen am Ende als eigenständiges Token separiert. Ein Teilen der Token nur anhand von Leerzeichen hätte das Fragezeichen nicht erkannt.

3.2.2 Part-Of-Speech Tagging

Das Part-Of-Speech Tagging (POS Tagging) beschreibt ein Verfahren, das zu jedem Wort eines Satzes die Wortart annotiert und ist ein wichtiger Bestandteil moderner Information Retrieval Systeme [Bri00, S. 403ff.][Bir06, S. 70].
Ein Beispiel ist der 2003 von Kristina Toutanova implementierte **Log-Linear Part-Of-Speech Tagger**, welcher im **Stanford Log-linear Part-Of-Speech Tagger** Anwendung findet [TKMS03]. Solche Tagger werden bei Methoden wie der Lemmatization verwendet. Außerdem können Fragen nicht nur auf ihre Wörter untersucht werden, sondern es können ebenso die POS-Tags der Fragen verglichen werden.

Beispiel

Den Wörtern der Beispiel-Frage werden mit Hilfe des Stanford Taggers auf der nächsten Seite ihre jeweiligen POS-Tags zugewiesen.

[1] `https://nlp.stanford.edu/software/tokenizer.shtml`

Tabelle 3.2: Der Stanford Tagger liefert die POS-Tags des Satzes „What is the most valuable thing you have?“, welche die Wortart der einzelnen Wörter beschreiben. Die Verben erhalten dabei sogar die Angabe, um welche grammatikalische Person es sich handelt.

Wort	**POS-Tag**	**Bedeutung**
What	WP	Wh-Pronomen
is	VBZ	Verb, Präsens, dritte Person Singular
the	DT	Bestimmungswort
most	RBS	Adverb, Superlativ
valuable	JJ	Adjektiv
thing	NN	Substantiv, Singular
you	PRP	Personalpronomen
have	VBP	Verb, Präsens, nicht dritte Person Singular
?	Punkt	Satzende

Neben den gezeigten POS-Tags existieren noch viele weitere, die in der Penn Treebank[2] zu finden sind.

3.2.3 Normalization

Nachdem die Token aus einem Text extrahiert wurden, bietet es sich an, diese durch Normalization zu vereinheitlichen, indem beispielsweise jedes Token durch Case-Folding in Kleinbuchstaben umgewandelt wird oder alle Satzzeichen entfernt werden [Man08, S. 30]. Es ist ebenso möglich, Umlaute durch Digraphen zu ersetzen, Token aus dem nicht-lateinischen Schriftsystem zu transliterieren oder sehr seltene Wörter mit Platzhaltern auszutauschen. Ein weiteres Konzept zur Normalization sind Äquivalenzklassen, in welcher bestimmte Token als äquivalent definiert und durch einen Repräsentanten der Klasse ersetzt werden [Man08, S. 28][SBC+01]. Die Token 'U.S.A' oder 'US' könnten etwa mit dem Wort 'america' eine Äquivalenzklasse bilden. Das Anwenden dieser Methoden, führt möglicherweise zu einem besseren Ergebnis im Fragevergleich.

Beispiel

Das obige Beispiel könnte durch Case-Folding zu Kleinbuchstaben transformiert werden, woraus sich „what is the most valuable thing you have?“ ergibt. Hierbei ändert sich jedoch nur das Wort „What“. Ob es sinnvoll ist für englische Fragen Case-Folding anzuwenden, zeigt sich noch später.

[2] `http://ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

3.2.4 Stemming

Stemmingverfahren werden benutzt, um die Wortstämme von Worten zu bilden. Dies kann unter anderem durch das wiederholte Anwenden von Regeln geschehen. Ein bekanntes Verfahren, das diese Methodik verwendet, ist der Porter Stemmer von M.F. Porter aus dem Jahr 1980 [Por80, S. 130]. Er definierte bestimmte Regeln, die Wörter kontinuierlich auf ihren Stamm reduzieren. Die Wörter „connected“, „connecting“, „connection“ dezimieren sich beispielsweise zu „connect“. Weiterhin wird zum Beispiel die Endung „ational“ zu „ate“ umgewandelt [Por80, S. 135]. Das Wort „relational“ transformiert der Stemmer infolgedessen zu „relate“. Krovetz fand 1993 heraus, dass diese Regeln jedoch öfter zu falschen Wortstämmen führen, wie bei dem Wort „organization“, welches zu „organ“ gestemmt wird [Kro93, S. 192ff].

Beispiel

Wird das Eingangsbeispiel gestemmt, ergeben sich folgende Token:
Tokenisiert: {What, is, the, most, valuable, thing, you, have, ?}
Gestemmt: {What, is, the, most, valuabl, thing, you, have, ?}
Das Wort „valuable“ verliert den letzten Buchstaben und es ergibt sich ein Wort, welches nicht in der englischen Sprache vorkommt. Aufgrund solcher Vorkommnisse wird in der Praxis oft Lemmatization anstelle von Stemming verwendet.

3.2.5 Lemmatization

Lemmatization verfolgt das gleiche Ziel wie Stemming, soll jedoch zuverlässigere Ergebnisse liefern und bezieht deswegen für die Stammformbildung Wörterbücher und Wortanalysen ein. Am Ende kommt die kanonische Form eines Wortes oder ein orthografisch korrektes Wort heraus. [Man08, S. 32]
Im NTLK ist ein Lemmatizer mit dem Namen WordNetLemmatizer implementiert, der neben dem Wort optional die entsprechende Wortart als Eingabe fordert. Diese kann durch POS-Tagging bestimmt werden. Das Wort „organization“ wurde vom Porter Stemmer falsch transformiert, bleibt mit dem WordNetLemmatizer unter Verwendung der POS-Tags aber erhalten.

Beispiel

Das Beispiel von oben sieht nach der Lemmatization wie folgt aus: {What, be, the, most, valuable, thing, you, have, ?}. Das Wort „is“ wird zu „be“ transformiert und vereinfacht den Fragevergleich. Anders als bei der Ausgabe des Porter-Stemmers bleibt jedes Wort lexikalisch korrekt. Ob die Verwendung eines Lemmatizers bessere Ergebnisse liefert, wird sich später in der Evaluation herausstellen.

3.2.6 Entfernen von Stoppwörtern

Ein optionaler Schritt am Ende der Segmentierung ist es, Stoppwörter zu entfernen. Dies sind Wörter, die sehr häufig in einer Sprache auftreten und deswegen wenig dazu beitragen, Texte beziehungsweise Fragen voreinander zu differenzieren. Sie werden in Stoppwortlisten zusammengefasst und enthalten im Englischen Wörter wie „I“, „me“, „my“, „myself“, „we“ oder „now“. [Man08, S. 27][WS92]
Eine große Liste an Stoppwörtern ist unter dieser Adresse[3] zu finden.

Beispiel

Die Stoppwort-Entfernung wiederum angewandt auf das Musterbeispiel ergibt folgende Menge an Token: {What, valuable, thing. ?}. Die Wörter „is“, „the“, „most“, „you“ und „have“ fallen weg. Ob es sich bei dem Fragevergleich lohnt, die Stoppwörter zu löschen, wird sich später bei der Entwicklung eines guten Verfahrens herausstellen.

3.3 Grundlagen des Textvergleichs

Diese Sektion soll die Grundlagen des Textvergleichs nahe bringen. Dazu gehören Verfahren wie die Levenshtein-Distanz und die Jaccard-Distanz, welche benutzt werden, um auszusagen, wie ähnlich sich zwei Zeichenketten sind. Außerdem wird die Termfrequenz und die Inverse Dokumentfrequenz erläutert, mit welchen die Relevanz einer Zeichenkette eingeschätzt werden kann.

3.3.1 Jaccard-Distanz

Der Botanikprofessor Paul Jaccard entwickelte 1901 den Gleichheits-Koeffizienten, der später zum Jaccard Index oder Jaccard-Distanz umbenannt wurde [Jac01, S. 564]. Er gibt die Ähnlichkeit zweier Mengen an und berechnet sich aus der Kardinalität der Schnittmenge dividiert durch die Kardinalität der Vereinigungsmenge. Dieser Wert liegt im Intervall $[0, 1]$, wobei eine hohe Jaccard-Distanz eine große Ähnlichkeit ausdrückt.

Beispiel

Es soll die Jaccard-Distanz der Fragen ’How I can speak English fluently?’ und ’How can I learn to speak English fluently?’ errechnet werden. Dafür werden die Fragen zuerst durch Trennung der Wörter in Mengen umgewandelt.
So ergeben sich:

$$M_1 = \{\text{How, I, can, speak, English, fluently?}\}$$
$$M_2 = \{\text{How, can, I, learn, to, speak, English, fluently?}\}$$

[3] `http://xpo6.com/list-of-english-stop-words/`

Aus den beiden Mengen kann die Schnittmenge und die Vereinigungsmenge gebildet werden:

$M_1 \cap M_2 = \{\text{How, I, can, speak, English, fluently?}\}$ (Kardinalität 6)
$M_1 \cup M_2 = \{\text{How, I, can, speak, English, fluently?, learn, to}\}$ (Kardinalität 8)

Die Jaccard-Distanz beträgt damit $6 \div 8 = 0{,}75$.

3.3.2 Levenshtein-Distanz

Ein weiteres Maß zur Bestimmung der Ähnlichkeit zweier Texte ist die Levenshtein-Distanz, die Vladimir Levenshtein im Jahre 1965 einführte. Dieser Wert wird anhand der benötigten Operationen berechnet, die nötig sind, um von einem Text zum anderen zu gelangen. Eine Operation erhöht das Maß um Eins und umfasst das Löschen, Einfügen oder Ersetzen einzelner Zeichen. [Lev66]

Beispiel

Zur Verdeutlichung der Levenshtein-Berechnung sei folgendes Beispiel gegeben: Es soll die Distanz zwischen 'married' und 'marry' bestimmt werden. Dafür sind nachstehende Schritte notwendig:

1. married
2. marryed (i mit y ersetzen)
3. marrye (d löschen)
4. marry (e löschen)

Die Levenshtein-Distanz zwischen married und marry beträgt demnach 3 (1x ersetzen, 2x löschen). Für Fragen kann das gleiche Prinzip angewendet werden.

3.3.3 Termfrequenz und Inverse Dokumentfrequenz

Die Termfrequenz gibt an, wie häufig ein Term in einem Dokument vorkommt, wobei beim Fragevergleich die Häufigkeit eines Tokens pro Frage gemeint ist. Da die meisten Fragen relativ wenig Worte besitzen, ist dieser Wert nicht sonderlich aussagekräftig, weswegen die im Information Retrieval eher die Inverse Dokumentfrequenz Verwendung findet. Sie gibt die Anzahl der Dokumente an, die einen bestimmten Term enthalten, also die Anzahl an Fragen aus dem jeweiligen Set, die ein bestimmtes Token beinhalten [Man08, S. 117f.]. Je kleiner der Wert ist, desto relevanter ist der entsprechende Term für den Fragevergleich, da er eine Frage sehr deutlich von den anderen Fragen abgrenzt. Hierbei muss jedoch beachtet werden, dass die Token mit einer extrem niedrigen Inversen Dokumentfrequenz höchstwahrscheinlich Rechtschreibfehler, Eigennamen, Symbole oder ähnliches sind und sich deswegen nicht zur Differenzierung von Fragen eignen. Die Top 3 der häufigsten Wörter des Testsets sind: „the“, „What“ und „I“, wohingegen unter anderem die folgende Wörter nur einmal vorkommen: „sabine“, „manchurian“ und „sanam“.

3.4 Maschinelles Lernen

Der Begriff maschinelles Lernen umfasst Methoden, die es einem Computer möglich machen, aus Erfahrungen zu lernen. Das Ziel solcher Methoden ist es zum Beispiel, das menschliche Lernverhalten nachzubilden und in der natürlichen Sprachverarbeitung anzuwenden. Besonders gut eignet sich maschinelles Lernen bei Klassifizierungsproblemen. Dabei kann grundsätzlich in überwachtes und unüberwachtes Lernen unterschieden werden, je nachdem, ob der Algorithmus ein Performanz-Feedback erhält. Da in der Praxis jedoch am häufigsten überwachtes Lernen angewandt wird, soll im Folgenden unüberwachtes Lernen nicht weiter Betrachtung finden. [Lan96, S.ix,4,8] [Mit04, S. 376f.] [JM00, S. 118]

3.4.1 Überwachtes Lernen

Das Ziel des maschinellen Lernens ist es, ein Modell zu erzeugen, das ein Problem durch Anwenden von Regeln löst. Für das Lösen eines Klassifizierungsproblems, wie das Problem des Fragevergleichs, wird beim überwachten Lernen häufig von „learning from example“ gesprochen. Das heißt, anhand eines Beispiels versucht der Algorithmus zu lernen, zu welcher Klasse dieses Beispiel gehört. Ziel ist es, zu jedem Eingangswert den korrekte Ausgangswert vorherzusagen. Die Grundlage dafür bildet das Trainingsset, da es aus korrekt klassifizierten Beispielen besteht, welche zuvor per Hand oder durch ein existierenden maschinelles Verfahren erstellt wurden. Neben dem Trainingsset sind Attribute nötig, die die festzustellenden Klassen beschreiben, sogenannte **Features**. Das Finden der Features stellt den Aufwändigsten Teil der Problemlösung dar und wird im Kapitel 4.3.2 genauer untersucht. [Lan96, S. 8][HTF09, S. 9,29][Mit04, S. 377][JM00, S. 118]
Angewandt auf den Fragevergleich bedeutet überwachtes Lernen, dass die disjunkten Klassen *Duplikat* beziehungsweise *kein Duplikat* den Fragepaaren mit Hilfe von Features durch ein Modell zugewiesen werden müssen. Dieses Modell lernt anhand der Features im sogenannten **Modelltraining**. Die Fragen an sich spielen an dieser Stelle keine Rolle mehr, da alle nötigen Informationen in Features extrahiert wurden.

Extreme Gradient Boosting (XGBoost) XGBoost stellt ein **skalierbares** System für maschinelles Lernen dar, das auf dem Modell der **Greedy Function Approximation** und basiert, dessen Funktionsweise hier nicht weiter vertieft werden soll. Es wurde in den Top 3 Lösungen von 29 Kaggle-Competitions benutzt und auch bei dem Netflix Prize verwendet [BL+07]. In diesem ging es darum, anhand von Filmvorlieben der Nutzer vorherzusagen, welche Filme ihnen ebenfalls gefallen könnten. Außerdem nutzte Kaggle in 17 ihrer Blogs[4] XGBoost als Lösung für ein Problem und XGBoost fand 2015 im KDDCup, einem jährlichen Wettbewerb der ACM Special Interest Group on Knowledge Discovery and Data Mining, bei jedem der Top-10 Gewinner Anwendung. [CG16, S. 1]

[4] `http://blog.kaggle.com`

3.4.2 Überanpassung und Unteranpassung

Von Überanpassung wird gesprochen, wenn das Modell zu stark an das Trainingsset angepasst wird, sodass die Performanz auf das Testset im Vergleich dazu wesentlich schlechter ist [ES02, S. 314][SF95, S. 129f.]. Die Ursache können zu viele Parameter sein, die im Worst-Case für jedes Trainings-Paar die Ausgabe exakt vorherbestimmen, aber für die Testdaten nur falsche Ergebnisse liefern. Um Überanpassung zu verhindern, können **Early Stopping Rounds** angegeben werden, mit welchen nach einer bestimmten Anzahl an Trainingsiterationen das Training abbricht, sofern es nur noch marginale Verbesserungen gibt [YRC07, RWY11, ZY05]. Weitere Methoden zur Reduktion von Überanpassung sind **Cross-Validation** oder **Dropout**, auf welche an der Stelle nicht weiter eingegangen werden sollen [Koh95][SHK+14].
Bei der Identifikation von Fragepaaren hieße Überanpassung, dass das Modell nicht effektiv auf reale Fragepaare angewandt werden kann, weil es nur für die Fragen des Trainingssets gut funktionieren würde. Da die Test- und Trainingsdaten, wie in 2.3 festgestellt, jedoch der gleichen Menge entsprungen sind, ist das Risiko für Überanpassung als gering einzustufen. Da zudem das primäre Ziel in dieser Arbeit ein gutes Verfahren für das Trainingsset ist, spielt Überanpassung keine große Rolle.
Unteranpassung hingegen beschreibt ein Modell, das zu linear ist, da es zu wenig Parameter besitzt, die nötig sind, um die Datenpunkte des Trainingssets abzubilden. Um dieses Problem zu beheben, müssen lediglich mehr Parameter definiert werden, die zu den Eingabedaten die richtigen Ausgabedaten vorhersagen können. [ES02, S. 440][SF95, S. 130]

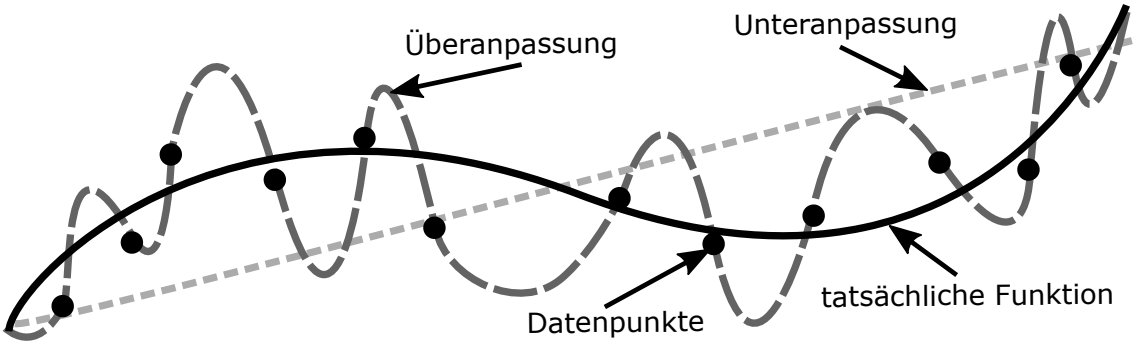


Abbildung 3.1: Diese sinusähnliche Funktion soll mit einem Modell abgebildet werden. Die schematische Darstellung zeigt, dass durch zu wenige Parameter eine Unteranpassung entsteht und keine guten Ergebnisse erzielt werden. Überanpassung tritt auf, wenn das Modell zu viele Parameter besitzt. Das führt dazu, dass auf Daten, die nicht dem Trainingsset entspringen, keine zufriedenstellenden Ergebnisse mehr geliefert werden. In diesem Fall wurde das Modell zu stark an die Trainingsdaten angepasst.

4 Entwicklung eines Verfahrens

Nachdem einige Grundlagen für die Verfahrensentwicklung erläutert wurden, müssen diese sinnvoll kombiniert werden, um ein gutes Verfahren zur Erkennung von Frageduplikaten erstellen zu können. Es werden nachfolgend sehr einfach gehaltene Verfahren beschrieben und direkt evaluiert, da sie später wenig relevant sein werden. Der Fokus dieses Kapitels soll auf maschinellem Lernen liegen. Die Implementierung der Verfahren geschieht in Python 3.6.

4.1 Python Entwicklungsumgebung

Als Programmiersprache zur Implementierung des Verfahrens wurde Python gewählt, da die Sprache gegenüber anderen Programmiersprachen wie R, C#, MATLAB oder Java einige Vorteile bietet. Python ist sehr weit verbreitet und hat eine große aktive Community, vor allem im Bereich Data Science und Datenanalyse. Zudem ist die Programmiersprache einfach zu erlernen und effizienter in der Performanz[1] als beispielsweise MATLAB. Für Python wurden außerdem viele nützliche Data Science-Bibliotheken aus anderen Sprachen portiert. Einige davon sind NumPy[2], SciPy[3], Scikit-Learn[4], Pandas[5] oder matplotlib[6], welches sich hervorragend zur Datenvisualisierung eignet.

4.2 Baseline-Implementierungen

Die Grundlage für alle kommenden Baseline-Implementierungen stellen die Klassen im Diagramm 4.1 auf Seite 27 dar. Sie dienen als ersten Referenzpunkt, wie verschiedene, einfache Ansätze auf Test- und Trainingsset performen. Zur Evalution wird der Log-Loss verwendet (siehe 5.1.5), da es das einzige verfügbare Evaluierungsverfahren für das Testset ist.
In der Basisklasse *ComparingBase* wird das komplette Trainingsset eingelesen. Mit Hilfe der *get_matching_score*-Funktion wird das Trainingsset durchlaufen und der Score errechnet. Als Parameter steht *for_percent_of_data* zur Verfügung, mit welchem angegeben werden kann, welcher Anteil des Trainingssets durchlaufen werden soll.

[1] `http://scipy.github.io/old-wiki/pages/PerformancePython`
[2] `http://numpy.org/`
[3] `http://scipy.org/`
[4] `http://scikit-learn.org/`
[5] `http://pandas.pydata.org`
[6] `http://matplotlib.org`

Da durchaus lange Laufzeiten auftreten können, errechnet ein Durchlauf mit 5% der Daten bereits einen realistischen Score in kurzer Zeit.
Damit eigene Vergleichsoperationen angewandt werden können, steht die abstrakte Funktion *predict_pair* zur Verfügung. Diese hat als Parameter das Fragepaar und den Wert, ob es sich um ein Duplikat handelt. Jede ableitende Klasse, wie zum Beispiel der *ExampleComparer*, muss diese Methode überschreiben, um eigene Vergleichsmethoden anzuwenden.

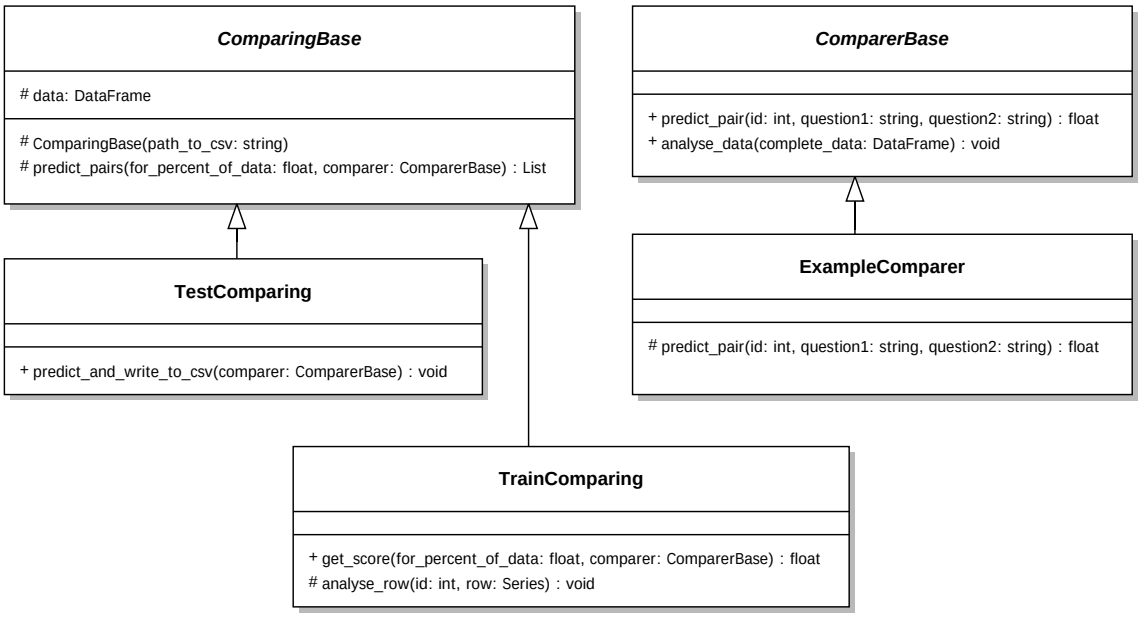


Abbildung 4.1: Die Klassenstruktur der Baseline-Implementierungen ermöglicht ein erstes Ausprobieren von einfachen Ansätzen für Test- und Trainingsset. Je nach Datenset kann zwischen der Basisklasse *TestComparing* und *TrainComparing* ausgewählt werden, welche *ComparingBase* als Elternklasse haben. Die Vergleichsklassen, wie zum Beispiel der *ExampleComparer*, leiten von *CompararBase* ab und können für beide Datensets verwendet werden.

Diese Klassen werden später auf Grund eines komplexeren Verfahrens abgelöst. Für den Anfang soll jedoch zuerst überprüft werden, wie die nachstehenden Ansätze abschneiden.

4.2.1 Equal-Comparer

Der erste Ansatz ist in der Klasse *EqualComparer* implementiert, die jedes Fragepaar als Duplikat ansieht. Das heißt, es wird jedem Paar der Wert 1 zugewiesen, woraus sich folgender Log-Loss ergibt:

- Trainingsset: 21,7876
- Testset: 28,4466

4.2.2 Unequal-Comparer

Die Klasse *UnequalComparer* stellt das Gegenteil des Equal-Comparers dar und sieht jedes Fragepaar als Nicht-Duplikat an, indem es allen Paaren den Wert 0 zuweist. Folgender Log-Loss resultiert:

- Trainingsset: 12,7516
- Testset: 6,0929

Wie bei 2.1 bereits festgestellt wurde, ist nur ungefähr jedes dritte Fragepaar des Trainingssets ein Duplikat. Damit war es zu erwarten, dass der Log-Loss des Unequal-Comparers besser ausfällt als der des Equal-Comparers.

4.2.3 Random-Comparer

Ein weiterer Ansatz ist es, ein Fragepaar zufällig als Duplikat anzusehen. Implementiert ist dies in der *RandomComparer*-Klasse, in welcher jedem Paar zufällig der Wert 0 oder 1 zugewiesen wird. Eine zweite Variante ist es, anstelle von 0 oder 1, eine zufällige Fließkommazahl aus dem zwischen diesen Zahlen liegenden Intervall zuzuweisen. Es ergeben sich die folgende Werte:

- Trainingsset Version 1: 17,3075
- Trainingsset Version 2: 0,9984
- Testset Version 1: 17,2066
- Testset Version 2: 1,0026

Wie zu erkennen ist, existiert ein deutlicher Unterschied zwischen den beiden Verfahren, da falsche Vorhersagen den Score bei Version 1 stärker verschlechtern. Beispielsweise führt der Wert 0,3 bei einem Duplikat zu einem viel besseren Log-Loss als der Wert 0 (siehe erstes Beispiel bei 5.1.5). Die zweiten Version stellt demnach eine erhebliche Verbesserung zu den vorangegangenen Implementierungen dar.

4.2.4 Mean-Comparer

Bis jetzt wurden keinerlei Daten in die Berechnung einbezogen. Ein erster datenbezogener Ansatz ist die Bildung des Durchschnittes der is_duplicate-Felder des Trainingssets, welcher 0,3692 beträgt. Dieser wird dann jedem Fragepaar zuzuweisen. Das Verfahren ist in der Klasse *MeanComparer* implementiert. Es ergibt sich folgender Log-Loss:

- Trainingsset: 0,6585
- Testset: 0,5553

Der Mean-Comparer stellt erneut eine Verbesserung zum Random-Comparer (Version 2) dar. Der Log-Loss für das Testset ist hierbei besser als für das Trainingsset. Es lässt sich vermuten, dass der Anteil an Duplikaten im Testset noch geringer ist.

4.2.5 Zusammenfassung

Die dargestellten Baseline-Implementierungen stellen die simpelsten Herangehensweisen dar und zeigen auf, wie einfach es ist, einen Log-Loss kleiner Eins zu erreichen. Speziell der Mean-Comparer stellt eine obere Schranke für alle nachfolgenden Verfahren dar und dessen Log-Loss sollte durch keinen komplexen Algorithmus unterschritten werden. Folgende Tabelle fasst die vorangegangenen Implementierungen noch einmal zusammen:

Tabelle 4.1: Der Log-Loss für die Baseline-Implementierung zeigt, das der Mean-Comparer bereits einen guten Log-Loss für Test- und Trainingsset liefert. Es ist ebenfalls erkennbar, dass bei allen Implementierungen der jeweilige Log-Loss beider Sets ähnlich ausfällt.

Verfahren	**Rückgabewert**	**Trainingsset**	**Testset**
EqualComparer	1	21,7876	28,4466
RandomComparer V1	0 oder 1	17,3075	17,2066
UnequalComparer	0	12,7516	6,0929
RandomComparer V2	zwischen 0 und 1	0,9984	1,0026
MeanComparer	0,3692	0,6585	0,5553

4.3 Schritte zu einem Modell

Nachdem mit einfachen Verfahren bereits gute Ergebnisse erzielt wurden, besteht die nächste Aufgabe darin, weiterführende Methoden anzuwenden. Dazu werden die Datensets zuerst vorverarbeitet, um sie zu vereinheitlichen. Anschließend werden die Features berechnet und in einer CSV-Datei abgespeichert, damit sie danach für das Modelltraining benutzt werden können.

4.3.1 Vorverarbeitung

Die Vorverarbeitung stellt den ersten Schritt der Entwicklung eines geeigneten Verfahrens zur Duplikaterkennung dar und besteht aus den in Kapitel 3 genannten Herangehensweisen.

Segmentation & Tokenization

Zuerst muss jede Frage in ihre Token segmentiert werden. Dafür gibt es zum einen die Möglichkeit jedes Token durch Leerzeichen zu separieren oder einen Tokenizer zu verwenden. Der Unterschied liegt darin, dass zum Beispiel Fragezeichen in der ersten Methode nur als eigenständiges Token erkannt werden, wenn zwischen letztem Wort und Fragezeichen ein Leerzeichen steht.

Normalization

Anschließend können Aneinandereihungen von Token oder einzelne Token durch Normalization vereinheitlicht werden. Es existieren nur wenige Muster, die mehrere aufeinanderfolgende Token beschreiben. Einige davon sind:

- n't → not
- whats → what is
- the USA → America

Für einzelne Token hingegen existieren zum Beispiel folgende Vereinheitlichungen:

Tabelle 4.2: In der Tabelle sind einige Verfahren zur Normalization von Token mit Beispielen für Ein- und Ausgaben dargestellt. Durch diese Transformationen können die Fragen später besser verglichen werden.

Methode	**Beispiel**	
	Eingabe	**Ausgabe**
Ersetzen von Verb-Kurzformen	's, 'm, 're, 've	is, am, are, have
Entfernen von Zifferngruppierungen	1,000	1000
Korrigieren von Rechtschreibfehlern	intially, imrovement	initially, improvement
Ersetzen von Laufwerksnamen	C:\, D:\, E:\	disk
Vereinheitlichen von Abkürzungen	Rd, lb, U.S.A.	road, pound, America
Ersetzen von Nicht-ASCII Zeichen	®	nonascii
Austauschen von Fließkommazahlen	1.3454	12345
Anwenden von Case-Folding	Quora, DNA	quora, dna

Schätzungsweise kann durch Normalization das Verfahren am Ende nur minimal verbessert werden, da im Vergleich zur Gesamtmenge nur wenige Fragen von obigen Methoden profitieren.

Stemming, Lemmatization und Entfernung der Stoppwörter

Nachdem die Normalization abgeschlossen ist, kann jedes Token durch Stemming oder Lemmatization auf seine Stammform reduziert werden. Es bietet sich gegebenenfalls an, parallel dazu die Stoppwörter zu entfernen. Dadurch ist es jedoch im Nachhinein nicht mehr möglich, die Wörter einer Frage mit ihren POS-Tags zu versehen. Um diesen Nachteil zu umgehen, könnten die im Zuge der Lemmatization sowieso gebildeten POS-Tags nach Stammformbildung separat gespeichert werden. Es empfiehlt sich ohnehin, die POS-Tags zu speichern, da die Bestimmung dieser sehr zeitaufwändig ist.

4.3.2 Feature Extraction

In der Feature Extraction werden alle Features erstellt und in einer CSV-Datei abgespeichert. Die Fragen, aus denen die Features extrahiert wurden, sind anschließend beim Modelltraining nicht mehr nötig. Das Ermitteln der Features stellt den wichtigsten Schritt in der Entwicklung eines guten Verfahrens zur Duplikaterkennung dar. Die gefundene Features wurden in Gruppen aufgeteilt und orientieren sich an Lösungen anderer Competition-Teilnehmer[7] [8] [9] [10] [11] [12], welche zusammengetragen und erweitert wurden.

Wordcount Features

Die Wordcount Features umfassen alle Merkmale, die sich aus Wort- und Zeichenanzahl ableiten lassen. Dabei werden die Anzahl an Zeichen jeweils mit beziehungsweise ohne Leerzeichen und die Anzahl aller Wörter und der verschiedenen Wörter mit beziehungsweise ohne Stoppwörter betrachtet.

Summe der Zeichen, inklusive Leerzeichen

- **len_q1** enthält die Anzahl der Zeichen von Frage 1
- **len_q2** enthält die Anzahl der Zeichen von Frage 2
- **min_len** = min(len_q1, len_q2)
- **max_len** = max(len_q1, len_q2)
- **len_diff** = max_len - min_len
- **len_ratio** = max_len ÷ min_len

Summe der Zeichen, exklusive Leerzeichen

- **len_q1_blank** enthält die Anzahl der Zeichen von Frage 1, ohne Leerzeichen
- **len_q2_blank** enthält die Anzahl der Zeichen von Frage 2, ohne Leerzeichen
- **min_len_blank** = min(len_q1_blank, len_q2_blank)
- **max_len_blank** = max(len_q1_blank, len_q2_blank)
- **len_diff_blank** = max_len_blank - min_len_blank
- **len_ratio_blank** = max_len_blank ÷ min_len_blank

[7] http://github.com/Wrosinski/Kaggle-Quora
[8] http://github.com/aerdem4/kaggle-quora-dup
[9] http://github.com/prischu/quora_question_pairs
[10] http://kaggle.com/c/quora-question-pairs/discussion/34342
[11] http://kaggle.com/sudalairajkumar/simple-leaky-exploration-notebook-quora
[12] http://linkedin.com/pulse/kaggle-quora-question-pairs-mar-2017-may-priscilla-li

Summe der Wörter, inklusive mehrfach vorkommender Wörter

- **wc_q1** enthält die Anzahl der Wörter von Frage 1
- **wc_q2** enthält die Anzahl der Wörter von Frage 2
- **min_wc** = min(wc_q1, wc_q2)
- **max_wc** = max(wc_q1, wc_q2)
- **wc_diff** = max_wc - min_wc
- **wc_ratio** = max_wc ÷ min_wc
- **total_wc** = min_wc + max_wc

Summe der verschiedenen Wörter, inklusive Stoppwörter

- **wc_q1_unique** enthält die Anzahl der Wörter von Frage 1, wobei mehrfach vorkommende Worte einfach gezählt werden
- **wc_q2_unique** enthält die Anzahl der Wörter von Frage 2, wobei mehrfach vorkommende Worte einfach gezählt werden
- **min_wc_unique** = min(wc_q1_unique, wc_q2_unique)
- **max_wc_unique** = max(wc_q1_unique, wc_q2_unique)
- **wc_diff_unique** = max_wc_unique - min_wc_unique
- **wc_ratio_unique** = max_wc_unique ÷ min_wc_unique
- **total_wc_unique** = min_wc_unique + max_wc_unique

Summe der verschiedenen Wörter, exklusive Stoppwörter

- **wc_q1_unique_stops** enthält die Anzahl der Wörter von Frage 1, die keine Stoppwörter sind und wobei mehrfach vorkommende Worte einfach gezählt werden
- **wc_q2_unique_stops** enthält die Anzahl der Wörter von Frage 2, die keine Stoppwörter sind und wobei mehrfach vorkommende Worte einfach gezählt werden
- **min_wc_unique_stops** = min(wc_q1_unique_stops, wc_q2_unique_stop)
- **max_wc_unique_stops** = max(wc_q1_unique_stops, wc_q2_unique_stop)
- **wc_diff_unique_stops** = max_wc_unique_stops - min_wc_unique_stops
- **wc_ratio_unique_stops** = max_wc_unique_stops ÷ min_wc_unique_stops
- **total_wc_unique_stops** = min_wc_unique_stops + max_wc_unique_stops

Bei allen Wordcount Features wurde bewusst das Minimum und Maximum gebildet, um später in der Evaluierung zu sehen, ob diese Features besser funktionieren als die Features in denen die Position der Fragen im Paar beachtet wird.
Weiterhin soll später untersucht werden, ob das Weglassen doppelter Wörter oder Stoppwörter zu einem besseren Verfahren führt. Es könnte sich ebenso herausstellen, dass alle Varianten zusammen in einem guten Ergebnis resultieren.

Levenshtein Feature

Das Levenshtein Feature besteht aus der Berechnung der Levenshtein-Distanz zweier Fragen:

- **levenshtein** enthält die Levenshtein-Distanz von Frage 1 und Frage 2

Shared Words Features

Ein erster komplexerer Ansatz ist es, die Summe gemeinsamer Wörter zu bilden. Dieses Feature ist in den Shared Words Features realisiert, welche aus drei Unterfeatures bestehen. Das erste gibt an, wie viele Wörter beide Fragen gemeinsam haben. Das zweite Feature berechnet die Jaccard–Distanz zwischen beiden Fragen und das letzte Feature benutzt nachstehende Formel:

$$\text{Wert} = \frac{2 \cdot \beta}{\alpha_1 + \alpha_2},$$

wobei

$$\begin{aligned} \alpha_1 &:= \text{Anzahl der Wörter von Frage 1} \\ \alpha_2 &:= \text{Anzahl der Wörter von Frage 2} \\ \beta &:= \text{Anzahl der Wörter, die in beiden Fragen vorkommen.} \end{aligned}$$

Das Ergebnis ist ein Wert zwischen 0 und 1, wobei das Fragepaar umso mehr gemeinsame Wörter besitzt, je näher der Wert an 1 liegt. Eine Variation ist es, die Stoppwörter vorher zu entfernen.
Es resultieren die nachstehenden Features:

- **common_words** enthält β
- **jaccard** enthält die Jaccard-Distanz
- **word_match** enthält den obigen Wert inklusive Stoppwörtern
- **word_match_stops** enthält den obigen Wert exklusive Stoppwörtern

Es ist möglich, dass später die beiden Versionen mit und ohne Stoppwörter in Kombination benötigt werden, um ein Duplikat zuverlässig zu erkennen.

TF-IDF Features

Um die Termfrequenz und die Inverse Dokumentfrequenz in das Verfahren einzubeziehen, wurden die TF-IDF Features implementiert. Diese fügen jedem Wert seine Gewichtung hinzu und für jedes Fragepaar wird, ähnlich wie bei den Shared Words Features, folgender Wert berechnet:

$$\text{Wert} = \frac{2 \cdot \beta}{\alpha_1 + \alpha_2},$$

wobei

$\alpha_1 :=$ Summe Gewichtung der Wörter von Frage 1
$\alpha_2 :=$ Summe Gewichtung der Wörter von Frage 2
$\beta :=$ Summe Gewichtung der Wörter, die in beiden Fragen vorkommen.

Eine Variante ist hierbei ebenfalls, die Stoppwörter nicht einzubeziehen. Es ergeben sich die beiden Features:

- **tfidf_wm** enthält den obigen Wert, inklusive Stoppwörter
- **tfidf_wm_stops** enthält den obigen Wert, exklusive Stoppwörter

Frequency Features

Eine weitere Idee ist es, sich die Nachbarn einer Frage anzusehen. Nachbarn stellen alle Fragen dar, mit denen eine Frage im kompletten Set verglichen wurde. Der zugrundeliegende Gedanke ist, dass Fragen, die öfter im Set verglichen werden, sich mit höherer Wahrscheinlichkeit in einem Duplikat befindet. Zusätzlich kann die Schnittmenge der Nachbarn beider Fragen gebildet werden. Je höher diese ist, desto häufiger sind beide Fragen und desto eher handelt es sich um ein Duplikat. Es ergeben sich diese Features:

- **q1_freq** enthält die Anzahl der Nachbarn von Frage 1
- **q2_freq** enthält die Anzahl der Nachbarn von Frage 2
- **intersect** enthält die Anzahl der gemeinsamen Nachbarn

Zusammenfassung

Insgesamt ergeben sich 43 Features, welche für das Modell Training benutzt werden können. Es bietet sich an, nach weiteren Features zu suchen, doch dieses Vorhaben würde an der Stelle zu weit führen. Darunter fallen zum Beispiel Aspekte wie N-Gramme, Wortvektoren, die Latent Dirichlet Allocation, Fuzzy Features[13] oder die TF-KLD [CT+94, MCCD13, BNJ03, JE13].

[13] `https://github.com/seatgeek/fuzzywuzzy`

4.3.3 Modell Training

Nachdem alle Features der Fragepaare aus dem Trainingsset extrahiert wurden, dienen sie als Grundlage für das Trainieren des Modells. Ein Modell gibt an, wie zu den Eingabewerten die Ausgabewerte vorherbestimmt werden. Die Features des Testsets sind hierfür irrelevant, da diese nicht klassifiziert sind.
Der erste Schritt besteht darin, alle Ein- und Ausgabedaten voneinander zu trennen, indem alle Features auf die Eingangsvariable X und alle is_duplicate-Werte auf die Ausgangsvariable Y zugewiesen werden. Um anschließend das Validierungsset zu erzeugen, werden zufällige 10% des Trainingssets abgespalten. Das verwendete Verfahren, um ein Modell zu erstellen, ist das in 3.4.1 vorgestellte XGBoost, welches neben Trainings- und Validierungsset folgende Parameter benötigt:

Tabelle 4.3: Die Tabelle gibt die Standardwerte der jeweiligen Parameter für XGBoost an. Diese beschreiben beispielsweise, wie mit Überanpassung umgegangen wird (*subsample*, *eta*) oder wie lange der Algorithmus trainieren soll (*num_boost_rounds*, *early_stopping_rounds*).

Parametername	**Wert**	**Beschreibung**
objective	reg:linear	gibt das Lernziel an, zum Beispiel „Lineare Regression“
eval_metric	rmse	gibt das Validierungsverfahren an, das verwendet werden soll, zum Beispiel „Root Mean Square Error“
eta	0,02	gibt die Lernrate an und soll Überanpassung verhindern, indem die Gewichtungen der Features durch den Wert heruntergesenkt werden
max_depth	7	gibt die Tiefe des erzeugten Baumes an; je höher der Wert, desto komplexer das Modell und desto wahrscheinlicher ist Überanpassung
base_score	0,5	gibt den initialen Score aller Instanzen an
scale_pos_weight	1	gibt die Balance für die positiven und negativen Klassen an. Berechnet sich bestenfalls aus Anzahl der negativen Klassen geteilt durch Anzahl der positiven Klassen. Für den Fragevergleich heißt das: Anzahl der Nicht-Duplikate ÷ Anzahl der Duplikate
num_boost_rounds	2500	gibt die Anzahl der Boosting-Runden an
early_stopping_rounds	50	gibt die Anzahl der Runden an, nach denen das Training vorzeitig beendet werden soll, wenn sich der Validierungs-Score innerhalb dieser nur geringfügig verbessert hat

4.3.4 Prediction

In der Prediction werden die Fragepaare des Validierungssets beziehungsweise des Testsets mit Hilfe des trainierten Modells klassifiziert. Das heißt, sie werden ihren Klassen zugewiesen, welche in diesem Fall im Intervall $[0, 1]$ liegen. Dafür sind die aus den Fragepaaren des jeweiligen Sets extrahierten Features erforderlich. Hierbei gibt es zunächst, wie bei dem Trainieren des Models, die Eingangsvariable X, die alle Features enthält. Der Ausgangsvariable Y soll das Ergebnis der Klassifizierung zugewiesen werden. In diesem Schritt stehen die Parameter des Modells bereits fest. Das Ergebnis der Prediction ist ein vollständig klassifiziertes Validierungs-/Testset.

4.3.5 Nachverarbeitung

Nachdem die Fragepaare klassifiziert wurden, kann es hilfreich sein, die Ausgabewerte anzupassen. Dabei muss für verschiedene Validierungsverfahren unterschiedlich vorgegangen werden.

Bei binärer Validierung

Da die Klassifizierung eines Fragepaars bei einer Validierung, die nur binäre Werte verarbeiten kann, 0 oder 1 betragen muss, ist es notwendig, Werte im Intervall $(0, 1)$ auf eine ganze Zahl zu bringen. Dafür muss eine Rundungsgrenze **rounding_boundary** definiert werden, die angibt, ab welchem Wert auf- und abgerundet werden soll. Im Normalfall beträgt dieser Wert 0,5, kann aber je nach Modell variieren.

Bei nicht-binärer Validierung

Bei einer nicht-binären Validierung, wie beispielsweise dem Log-Loss Verfahren, sind auch Vorhersagen zwischen 0 und 1 gültig. Hierbei kann es hilfreich sein, die Rundungsgrenze in zwei einzelne Grenzen aufzuteilen: Die Grenze *rounding_boundary_0* gibt an, bis zu welchem Wert *rounding_boundary_0* auf 0 abgerundet wird; der zweite Schwellwert *rounding_boundary_1* hingegen wird benutzt, um Werte die größer gleich $1 - rounding\_boundary\_1$ sind, auf 1 aufzurunden. Das hat den Vorteil, dass Predictions für Fragepaare, die mit hoher Wahrscheinlichkeit Duplikate oder Nicht-Duplikate sind, präziser in das Verfahren der nicht-binären Validierung eingehen.

5 Evaluation

Um einen guten Algorithmus zur Frageduplikaterkennung zu entwickeln, müssen die besten Parameter und Kombinationen der Features ermittelt werden. Dies geschieht anhand von Evaluationsmetriken. Die folgenden Kapitel beschäftigen sich mit diesen Metriken und präsentieren die Ergebnisse des maschinellen Lernens.

5.1 Evaluationsmetriken

Um festzustellen, wie gut oder schlecht ein Klassifizierungsverfahren ist, können bestimmte Evaluationsmetriken verwendet werden. Die wichtigsten sind Accuracy, Precision, Recall, F1 und Log-Loss. In den nächsten Unterkapiteln wird jede Metrik kurz erläutert. Mithilfe dieser Verfahren wird später entschieden, welche Parameter am besten zur Duplikaterkennung geeignet sind. Die Ergebnisse der Klassifizierung können im Fragevergleich grundsätzlich in vier Basiswerte eingeteilt werden [Pow11]:

- True Positives (TP): Anzahl an richtig erkannten Duplikaten
- True Negatives (TN): Anzahl an richtig erkannten Nicht-Duplikaten
- False Positives (FP): Anzahl an Nicht-Duplikaten, die als Duplikat erkannt wurden
- False Negatives (FN): Anzahl an Duplikaten, die als Nicht-Duplikat erkannt wurden

Folgende Beispielrechnung soll der Veranschaulichung dienen:

Beispiel

Gegeben seien 10 Fragepaare, unter denen 6 Duplikate sind, von welchen 5 korrekt als Duplikat erkannt wurden. Außerdem erkannte der Klassifizierer 2 der Nicht-Duplikate.
Es resultieren folgende Basiswerte:

- True Positives: 5
- True Negatives: 2
- False Positives: 4-2=2
- False Negatives: 6-5=1

5.1.1 Accuracy

Die Accuracy gibt das Verhältnis von richtigen und falschen Klassifikationen an:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N},$$

wobei N die Anzahl an Fragepaaren ist:

$$N = \text{TP} + \text{TN} + \text{FP} + \text{FN}$$

Die Accuracy ist leicht verständlich und wird deswegen sehr häufig verwendet. Ziel dieser Arbeit soll es sein, eine Accuracy von mindestens 80% zu erreichen. Es sollte allerdings beachtet werden, dass die Accuracy nicht immer ein gutes Maß für die Evaluation eines Klassifizierers ist, besonders wenn das Validierungsset sehr unausgeglichen ist. Bei den Fragepaaren im Trainings-/Validierungsset handelt es sich nur bei ungefähr jedem dritten Paar um ein Duplikat, weswegen die Accuracy des Unequal-Comparers bereits über 60% erreicht (dazu später mehr in 5.4.1). Im Best-Case beträgt sie sogar 100%, falls im gesamten Trainingsset nur Nicht-Duplikate vorkommen. [Pow11]
Für das obige Beispiel ergibt sich eine Accuracy von:

$$\frac{5+2}{10} \mathrel{\hat{=}} 70\%$$

5.1.2 Precision

Um den Algorithmus besser bewerten zu können, bietet es sich an, neben der Accuracy ebenfalls die Precision zu bestimmen. Diese gibt an, wie relevant die gefundenen Duplikate sind und berechnet sich aus:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Das heißt, je größer die Precision ist, desto mehr unter den als Duplikate klassifizierten Fragepaaren sind tatsächlich Duplikate und desto weniger False Positives existieren. [BYR01, S. 75][VR79, Pow11]
Das Muster-Beispiel hat die folgende Precision:

$$\frac{5}{5+2} = \frac{5}{7} \mathrel{\hat{\approx}} 71\%$$

5.1.3 Recall

Wird die Precision berechnet, so ist es sinnvoll, ebenfalls den Recall zu bestimmen, da diese beiden Basiswerte zusammen aussagekräftiger sind. Der Recall gibt das Verhältnis zwischen erkannten Duplikaten und allen Duplikaten an:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Das heißt, je größer der Recall ist, desto mehr Duplikate wurden gefunden und desto weniger False Negatives existieren. [BYR01, S. 75][VR79, Pow11]
Das Beispiel erreicht den folgenden Recall:

$$\frac{5}{5+1} = \frac{5}{6} \overset{\wedge}{\approx} 83\%$$

Der Recall bewertet, wie ernst die Precision zu nehmen ist, denn, wenn bei dem genannten Beispiel nur ein Frageduplikat erkannt werden würde (False Positives = 0), ergäbe sich eine Precision von 100%. Die anderen 4 Duplikate wären nicht gefunden worden, was zu einem schlechten Recall von $\approx 17\%$ geführt hätte.

5.1.4 F1

Der F1 kombiniert die beiden Werte Precision und Recall, indem er die harmonische Mitte dieser Werte bildet [Pow11]. Es resultiert ein Wert zwischen 0 und 1, welcher aussagekräftiger als die Accuracy sein soll, da nicht nur True Positives und True Negatives in dessen Berechnung einbezogen werden:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Je näher der F1 an 1 liegt, desto besser, denn im Fall F1 = 100% müssen Precision und Recall ebenfalls 100% betragen.
Im genannten Beispiel beträgt der F1:

$$2 \cdot \frac{0{,}71 \cdot 0{,}83}{0{,}71 + 0{,}83} \overset{\wedge}{\approx} 77\%$$

Der F1 kann verwendet werden, wenn die Accuracy bei dem Vergleich zweier Verfahren nicht aussagekräftig genug ist, oder wenn das Validierungsset sehr unbalanciert ist.

5.1.5 Log-Loss

Die Evaluierung der Testsets erfolgt in der Quora Competition auf Kaggle per Log-Loss. Sie liefert für eine Klassifikation bestenfalls den Wert 0, wobei nach oben keine Grenze gesetzt ist. Das liegt vor allem daran, dass falsche Predictions gegenüber richtigen stärker in den Log-Loss einberechnet werden.
Um die Funktion zu berechnen, muss durch einen Klassifizierer jeder Klasse ein Wert zwischen 0 und 1 zugeordnet werden. Somit ist es möglich, im Gegensatz zu den vorangegangenen Metriken, Klassifizierer zu bewerten, die keine Binärwerte liefern. Im Vergleich dazu kann die Accuracy keine Kommazahlen verwerten, weswegen die Ausgaben vor der Bewertung gerundet werden müssen, wobei allerdings Informationen verloren gehen (zum Beispiel wie exakt eine Klasse bestimmt wurde).

Folgende Formel beschreibt den Log-Loss:

$$\text{Log-Loss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \ln p_{ij}$$

Dabei ist N ist die Anzahl der Samples, was der Anzahl an Fragepaaren entspricht und M ist die Anzahl an möglichen Klassen, also Zwei (0 und 1). Die Variable y_{ij} enthält die korrekte Klasse j für das Fragepaar i und p_{ij} enthält die von dem Modell vorhergesagte Wahrscheinlichkeit für die Klasse j des Fragepaars i. Dadurch, dass M Zwei beträgt, lässt sich die Formel durch Auflösen der zweiten Summe etwas vereinfachen:

$$-\frac{1}{N}\sum_{i=1}^{N}[y_i \ln\, p_i + (1-y_i)\ln\,(1-p_i)]$$

Zur Veranschaulichung soll folgendes Beispiel dienen, welches den Log-Loss für ein Frageduplikat, also $N = 1$ und $y_1 = 1$, berechnet:

$$-[1 \ln\, p + (1-1)\ln\,(1-p)] = -\ln\, p$$

- für $p = 0{,}1$ ergibt sich: 2,3026
- für $p = 0{,}9$ ergibt sich: 0,1054

Die Ergebnisse zeigen, wie stark falsche Vorhersagen negativ ins Gewicht fallen. Wie im nachstehenden Diagramm zu sehen ist, muss ein Klassifizierer umso stärker verbessert werden, je kleiner der Log-Loss sein soll.

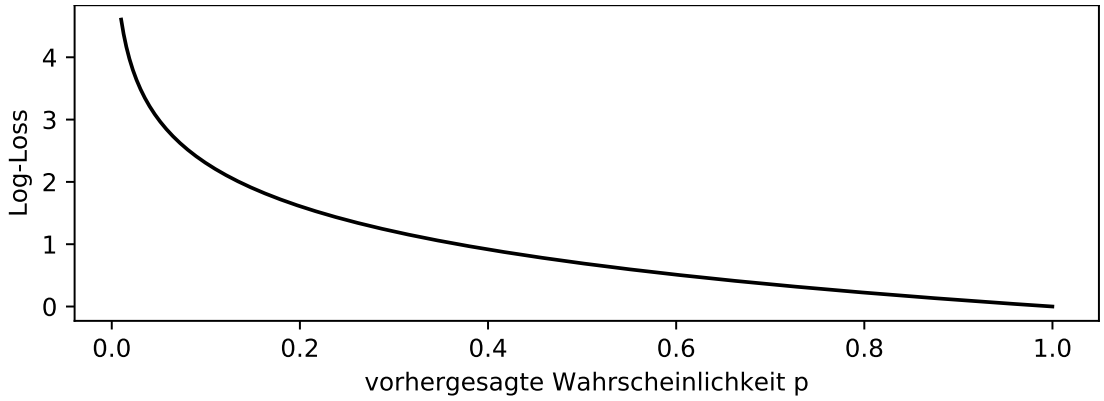


Abbildung 5.1: Die Log-Loss Funktion für ein Fragepaar zeigt deutlich: Ein Klassifikator muss am Ende viel stärker angepasst werden, um den gleichen Fortschritt wie am Anfang zu erzielen.

Der Log-Loss lässt sich für das zu Beginn definierte Beispiel anhand der Vorhersagen berechnen:

- für $p = 1 \wedge y = 1$: 5x und für $p = 1 \wedge y = 0$: 1x
- für $p = 0 \wedge y = 1$: 2x und für $p = 0 \wedge y = 0$: 2x

Damit ergibt sich ein sehr schlechter Log-Loss von 6,9078.

5.2 Hyperparameter Optimization

Bei Hyperparametern handelt es sich um Parameter, die selbst wiederum Parameter steuern [RN04, S. 877f]. Das heißt, im Fall der Identifikation gleicher Fragepaare stellt ein Hyperparameter beispielsweise die für das Training ausgewählten Features dar, denn jedes Feature ist selbst ein Parameter, der die Ähnlichkeit zweier Fragen angibt. Weitere Hyperparameter sind die Rundungsstelle oder die Anzahl der Iterationen für das Trainieren des Modells. In der Hyperparameter Optimization wird versucht, die optimalen Hyperparameter für ein Modell beispielsweise durch Manual Search, Grid Search, Random Search, Bayesian Optimization, oder Gradient-based Optimization zu finden [BBBK11, BB12, SLA12, MDA15, HCL+03]. Diese Untersektion soll Grid Search und Random Search näher erläutern.

5.2.1 Grid Search

Die Grid Search ist neben der Manual Search, bei der eigenhändig nach den besten Hyperparametern gesucht wird, das am häufigsten verwendete Verfahren zur Optimierung von Hyperparametern und stellt den de-facto Standard dar [BB12, S. 1][HLST15, S. 330ff]. Dabei werden alle möglichen Kombinationen für einen Hyperparameter ausprobiert und derjenige, der am Ende das beste Modell liefert, ist der optimale Hyperparameter.
Der Vorteil an Grid Search ist, dass dieses Verfahren sehr gut parallelisiert werden kann [HCL+03, S. 5ff]. Angewandt auf die Auswahl an Features bedeutet das, dass alle möglichen Kombinationen der Features ausprobiert werden müssten. Es wird der Zeitaufwand als der größte Nachteil von Grid Search deutlich. Die Anzahl an Kombinationen für n Features berechnet sich, wenn jedes Mal mindestens ein Feature benutzt wird, aus:

$$\sum_{k=1}^{n} \binom{n}{k}$$

Für die 43 Features resultieren 8.796.093.022.207, also rund 8,8 Billionen, mögliche Kombinationen. Bei einer durchschnittlichen Modelltrainingslaufzeit von 2 Minuten ergeben sich damit 3,3 Millionen Jahre. Deswegen bevorzugen Data Scientists in der Praxis die Random Search als Optimierungsverfahren [BB12].

5.2.2 Random Search

Im Gegensatz zur Grid Search werden bei der Random Search zufällige Kombinationen für einen Hyperparameter gewählt. Dadurch lassen sich in kürzerer Zeit bessere oder ähnlich gute Modelle wie bei der Grid Search finden. Der Grund dafür ist, dass für ein Datenset meistens nur einige wenige Hyperparameter relevant sind. Die Manual Search liefert im Vergleich dazu zwar etwas bessere Ergebnisse, jedoch in deutlich längerer Zeit, weswegen eine Kombination beider Verfahren sinnvoll sein kann. [BB12, S. 1][BBBK11, S. 5]

5.3 Durchführung

In dieser Sektion werden die Durchgänge beschrieben, die nötig waren, um ein Verfahren zur Duplikaterkennung anhand der in 4.3.2 definierten Features zu finden. Dabei stellte die Accuracy den Hauptbewertungsmaßstab dar, da diese intuitiv am besten eingeschätzt werden kann. Für den Fall, dass die Accuracy zu wenig Aussagekraft besaß, wurde der F1 zusätzlich herangezogen.

5.3.1 Feature-Selection

Jeder Versuch bestand aus 170 Modelltrainings-Durchgängen unter Verwendung des Random Search Verfahrens (siehe 5.2.2). Die Wahl fiel auf die Random Search, da es, wie verdeutlicht, eine bessere Alternative zur Grid Search darstellt. Dabei bestand die Auswahl immer aus mindestens einem und maximal 43 Features. Das Ziel war es, herauszufinden, welche Features am relevantesten sind und ob weniger relevante Features in Kombination mit relevanten Features zu einem noch besseren Verfahren führen oder dieses verschlechtern.

5.3.2 Erstellen des Validierungssets

Das Trainingsset wurde wie in 1.2 beschrieben aufgeteilt, sodass ein Validierungsset aus 40.429 Fragepaaren entstand, von denen 14.994 Paare ein Duplikat waren. Somit war die Anzahl an Nicht-Duplikaten und Duplikaten relativ unausgewogen, wobei die Accuracy in Verbindung mit dem F1 trotzdem genügend Aussagekraft besaß.

5.3.3 Definieren der Scopes

Interessant war es zu wissen, wie ein Verfahren nicht nur auf das gesamte Validierungsset abschneidet, sondern auch für spezielle Fragepaargruppen. Um die Ergebnisse besser einschätzen zu können, wurden folgende vier **Scopes** definiert:

- **all** enthielt alle Fragepaare des Validierungsset → entsprach 40.429 Paare (davon 14.994 Duplikate)
- **0_30** enthielt alle Fragepaare des Validierungssets, bei denen beide Fragen aus maximal 30 Zeichen bestanden → entsprach 1.701 Paare (davon 634 Duplikate)
- **30_70** enthielt alle Fragepaaren des Validierungssets, bei denen beide Fragen zwischen 30 und 70 Zeichen lang waren → entsprach 19.390 Paare (davon 7.335 Duplikate)
- **70_150** enthielt alle Fragepaare des Validierungssets, bei denen beide Fragen aus mindestens 70 und maximal aus 150 Zeichen bestanden → entsprach 5.338 Paare (davon 1.236 Duplikate)

Besonders wichtig ist der Scope 30_70, da in diesem die Fragenlängen vertreten waren, die am häufigsten auftreten (siehe 2.3).

5.3.4 Ablauf

Das folgende Diagramm zeigt den generellen Ablauf der Evaluation. Wie zu entnehmen ist, mussten die Hyperparameter zuerst je nach Versuch eingestellt werden, indem beispielsweise Normalization oder Lemmatization angewandt wurde. Anschließend erfolgte die Extraktion aller Features in eine CSV-Datei, um sie in den folgenden Iterationen nicht erneut berechnen zu müssen. Jede Iteration bekam eine bestimmte Anzahl und Auswahl an Features für das Modelltraining, wobei keine Kombination doppelt auftrat. Nachdem das Training abgeschlossen war, wurden alle anfangs definierten Metriken angewandt. Die Accuracy und der F1 stellten die wichtigsten Metriken dar, aber der Log-Loss, die Precision und der Recall sollten ebenfalls berechnet werden. Die Ergebnisse aller vier Scopes wurden am Ende der Iteration in einer CSV-Datei angehängt und die nächste Iteration begann. Die Anzahl an Iterationen betrug 170, da diese Menge ein ausreichend repräsentatives Ergebnis lieferte, um einschätzen zu können, wie gut die zu testenden Hyperparameter performten.

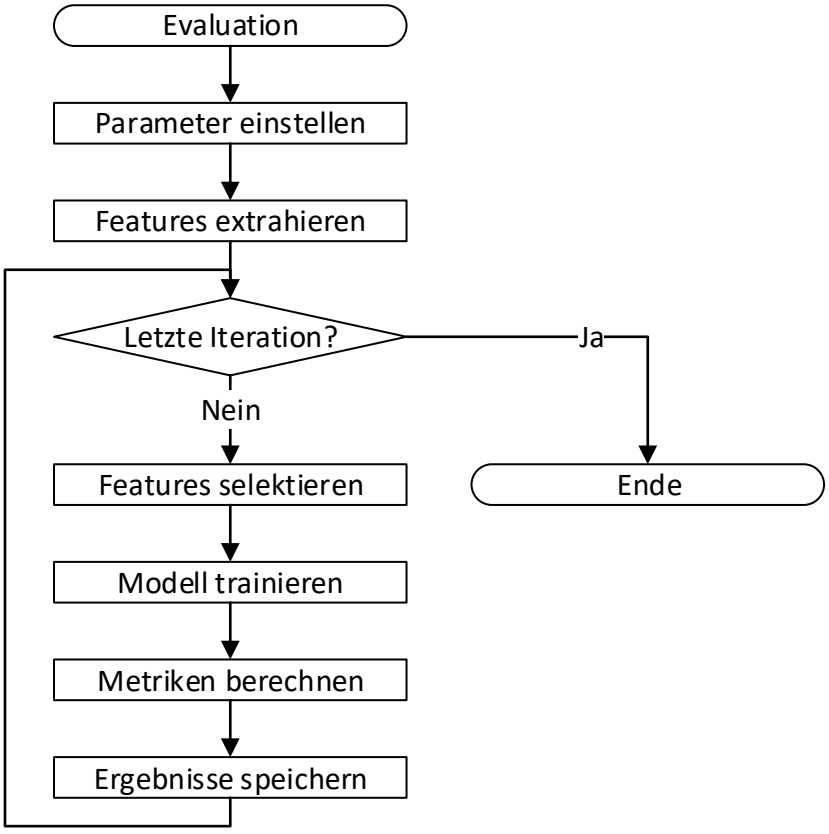


Abbildung 5.2: In der Evaluation wurden zuerst die Hyperparameter eingestellt und alle Features extrahiert, um anschließend für jede Iteration verschiedene Features selektieren zu können. Nachdem das Modell mit diesen Features trainiert und dessen Ergebnisse gespeichert wurden, begann die nächste Iteration.

5.4 Ergebnisse

In dieser Sektion folgen die Ergebnisse aller Versuche, einschließlich des finalen Versuchs, wobei bei ersteren das Hauptaugenmerk auf den gesamten Scope gelegt wurde, da für diesen die Ergebnisse die höchste Aussagekraft besaßen. Zu Beginn werden die zwei Baseline-Versuche, Equal-Comparer (4.2.1) und Unequal-Comparer (4.2.2), aus den Baseline-Implementierungen beschrieben. Anschließend folgt der Basisversuch, auf dem die Versuche zur Parameter Optimization und der finale Versuch basieren.

5.4.1 Baseline-Versuche

Damit die anderen Versuche besser eingeschätzt werden konnten, wurden die Baseline-Verfahren durchgeführt, bei denen alle Fragepaare entweder nur als Duplikat oder nur als Nicht-Duplikat klassifiziert wurden.

Tabelle 5.1: Die Ergebnisse der Baseline-Versuche zeigen, wie unausgewogen das Validierungsset ist und lassen schlussfolgern, dass für die Bewertung eines Versuches neben der Accuracy das F1 Maß hinzugezogen werden sollte.

Versuch	**is_duplicate**	**Accuracy**	**Recall**	**Precision**	**F1**	**Log-Loss**
1	0	62,91%	0%	0%	0%	8,5397
2	1	37,09%	100%	37,09%	54,11%	14,4862

Im ersten Versuch erhielt jedes Fragepaare die Markierung als Nicht-Duplikat, woraus sich eine Accuracy von 63% für die komplette Set ergab. Der Recall, die Precision und der F1 betrugen erwartungsgemäß 0%, da die Anzahl an True Positives Null war. Logischerweise betrug die Anzahl an False Positives ebenfalls Null. Für die Scopes 0_30 und 30_70 belief sich die Accuracy auf 55% und für den Scope 70_150 auf 72%. Die hohe Accuracy für den Scope mit den längeren Fragen ist daher begründet, dass in diesem nur 1.470 der 5.338 Fragepaare ein Duplikat waren. Damit war der Scope sehr unbalanciert. Dies musste bei den folgenden Versuchen beachtet werden, da dort mitunter viel bessere Accuracies im Vergleich zu den anderen Scopes herauskommen konnten. Die anderen Scopes waren, wie sich gezeigt hat, hingegen sehr gut ausbalanciert.
Bei dem zweiten Versuch, in welchem jedes Fragepaar als Duplikat gezählt wurde, entsprach die Precision der Accuracy, da die Anzahl an True Positives summiert mit der Anzahl an False Positives offensichtlich gleich der Anzahl an Fragepaaren war. Außerdem lag der Recall bei 100%, da alle Duplikate gefunden wurden und kein Duplikat als Nicht-Duplikat erkannt wurde. Für die anderen Scopes berechnete sich die Accuracy analog. Der F1 für den gesamten Scope ließ auf ein sehr schlechtes Verfahren schließen. Der Log-Loss beider Versuche war ebenfalls nicht zufriedenstellend.

5.4.2 Basisversuch

Die Grundlage für alle folgenden Versuche bildete der Basisversuch. Er sollte ein relativ gutes erstes Resultat liefern und im weiteren Verlauf optimiert werden. Dazu wurde Parameter Optimization auf die Parameter *max_depth*, *num_boost_rounds* und *rounding_boundary* und auf die Auswahl der Features angewandt.

Einstellen der Hyperparameter

In diesem Basisversuch waren die Parameter für das Modelltraining wie folgt definiert: Die Standardparameter (siehe 4.3.3) wurden vor dem Modelltraining leicht angepasst. Die Evaluation sollte per Log-Loss erfolgen, weswegen *eval_metric* den Wert *logloss* bekam und *objective* auf *binary:logistic* (logistische Regression) gesetzt werden musste. Der Basis-Score sollte 0,2 betragen, da dieser Log-Loss am Ende angestrebt wurde. Für die *early_stopping_rounds* wurden 50 angesetzt, da minimale Verbesserungen keine Rolle spielten. Ingesamt sollten 2.000 Trainings-Runden (*num_boost_rounds*) ausgeführt werden und die Duplikate wurden nicht künstlich vermehrt, um ein ausgeglicheneres Verhältnis im Trainingsset zu erhalten (*scale_pos_weight* = 1). Zudem sollten die Token der Frage per Leerzeichen separiert und in Kleinbuchstaben umgewandelt werden.

Tabelle 5.2: Die Hyperparameter des Basisversuchs sollten ein erstes Modell mit einer guten Accuracy liefern. Die Parameter *max_depth*, *num_boost_rounds* und *rounding_boundary* ließen noch einiges an Optimierungspotential offen. Die Evaluation erfolgte per Log-Loss und es wurden 2.000 Trainingsiterationen ausgeführt. Um minimale Verbesserungen außer Acht zu lassen, erhielt der Parameter *early_stopping_rounds* den Wert 50.

Hyperparameter	**Wert**	**zur Optimierung**
objective	binary:logistic	nein
eval_metric	logloss	nein
eta	0,02	nein
max_depth	7	ja
base_score	0,2	nein
scale_pos_weight	1	nein
num_boost_rounds	2.000	ja
early_stopping_rounds	50	nein
rounding_boundary	0,5	ja
Tokenization	per Leerzeichen	ja
Lemmatization	keine	ja
Case-Folding (Normalization)	Kleinbuchstaben	ja
Anzahl an Features	[1, 43]	ja

Ergebnisse

In der nachstehenden Tabelle sind die Iterationen dargestellt, die für die jeweilige Metrik den besten Wert lieferten:

Tabelle 5.3: Die Tabelle zeigt die **besten** Werte für jede Evaluationsmetrik des ersten Versuchs; dieser lieferte bereits gute Ergebnisse, die im Folgenden verbessert werden sollten. Die Accuracy war bereits schon zu Beginn höher als die angestrebten 80%.

Iteration	Accuracy	Recall	Precision	F1	Log-Loss	Features
31	76,66%	**83,97%**	64,16%	72,74%	0,4154	13
61	85,43%	66,66%	**91,82%**	77,24%	0,3092	10
12	**87,94%**	76,90%	89,10%	**82,55%**	0,2490	38
131	87,90%	76,90%	88,99%	82,51%	**0,2489**	41

Der erste richtige Versuch lieferte bereits ein gutes Ergebnis. Es konnten Kombinationen an Features gefunden werden, die eine Accuracy von 87,94% und einen Log-Loss von 0,2489 hatten. Die anderen Metriken lieferten ebenfalls gute Ergebnisse. Insgesamt betrug die Accuracy in 77 Durchgängen mindestens 87%. Der hohe Recall ging, wie zu erwarten war, mit einer schlechten Precision einher. Dieser Effekt war auch umgekehrt zu beobachten: Eine gute Precision trat nur in Verbindung mit einem schlechten Recall auf. Die Ursachen sind die in 5.1.2 und in 5.1.3 gewonnenen Erkenntnisse zu den Metriken Precision und Recall. Die Anzahl der Parameter für Iteration 31 und 61 zeigte, dass durch Unteranpassung (zu wenige Features) kein guter F1 beziehungsweise Log-Loss erreicht werden konnte.
Die höchste Accuracy der anderen Scopes war ebenfalls gut, wie in der nächsten Tabelle zu sehen ist. Bei den längeren Fragepaaren betrug die Accuracy sogar über 91%, wobei sich für die hauptsächlich mittellangen und die kurzen Fragepaare etwas schlechtere Accuracies mit 85,36% und 84,66% ergaben:

Tabelle 5.4: Die besten Durchgänge aller Scopes zeigen, dass für den Scope 70_150 bereits im Basisversuch eine Accuracy von über 91% erreicht wurde. Für die kurzen Fragepaare (0_30) wurde die schlechteste Accuracy unter allen Scopes erreicht. Es fiel außerdem auf, dass für den Scope 30_70 die meisten Features nötig waren, um eine gute Accuracy zu erreichen.

Scope	Accuracy	Recall	Precision	F1	Log-Loss	Features
all	87,94%	76,90%	89,10%	82,55%	0,2490	38
0_30	84,66%	78,13%	86,58%	82,14%	0,3358	39
30_70	85,36%	76,86%	89,38%	82,65%	0,3004	41
70_150	91,63%	81,09%	87,58%	84,21%	0,1711	38

Es fiel auf, dass für den Scope 30_70 am meisten Features für eine hohe Accuracy benötigt wurden. Die 20 besten Iterationen enthielten die folgenden sechs Features jedes Mal:

- levenshtein
- intersect
- q1_freq
- q2_freq
- min_unique_nostops_word_count
- ratio_unique_word_count

Dies lässt erkennen, dass zum Beispiel das Entfernen von Stoppwörtern genauso sinnvolle Ergebnisse lieferte wie das Einbeziehen dieser. Außerdem tauchte jedes Intersection Feature auf. Daraus kann geschlussfolgert werden, dass es neben dem Inhalt auch eine wichtige Rolle spielt, welche Nachbarn eine Frage hat. Weiterhin war das Levenshtein Feature bei den 45 besten Iterationen das wichtigste Feature. Im Folgenden sollte durch Hyperparameter Optimization ein noch besseres Modell gefunden werden.

5.4.3 Optimieren des Parameters num_boost_rounds

Bei den num_boost_rounds-Versuchen sollte ähnlich wie bei den max_depth-Versuchen untersucht werden, ob die Erhöhung um 500 Boosting-Rounds ein besseres Ergebnis lieferte. Die Versuche zeigten eine sehr leichte Verbesserung der Accuracy um 0,02%. Die geringe Verbesserung könnte daher kommen, dass early_stopping_rounds bei viele Fragepaaren schon bei 1.500 Iterationen das weitere Training abbricht, und so keine 2.500 Iterationen erreicht werden. Das heißt, es ist nicht zu erwarten, dass mit einer noch höheren Anzahl an Boosting Rounds bessere Ergebnisse erzielt werden. Somit stellen 2.500 Iterationen den optimalen Wert für der Parameter num_boost_rounds dar.

Tabelle 5.5: Die Varianten der num_boost_rounds-Versuche zeigten, dass durch mehr Boosting-Rounds die Accuracy leicht verbessert werden konnte, da das Modell mehr an das Trainingsset angepasst wurde.

Versuch	num_boost_rounds	∅ Accuracy Top 10	Accuracy Top 1
1	2.000 (Basisversuch)	87,90%	87,94%
2	2.500	87,90%	87,96%

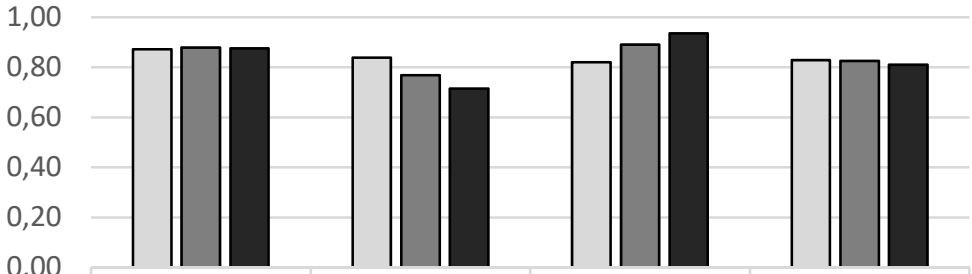
5.4.4 Optimieren des Parameters rounding_boundary

Die rounding_boundary-Versuche zielten auf das Finden der Rundungsstelle ab, welche das beste Modell ergeben würde. Allerdings lieferte die Betrachtung der Accuracies keine eindeutige Tendenz, welche die beste Rundungsstelle ist.

Tabelle 5.6: Die Varianten der rounding_boundary-Versuche unterschieden sich kaum in ihren jeweils höchsten Accuracies. Am besten schnitt der Basisversuch ab, bei dem die Rundungsstelle 0,5 betrug.

Versuch	rounding_boundary	∅ Accuracy Top 10	Accuracy Top 1
1	0,4	87,20%	87,23%
2	0,5 (Basisversuch)	87,90%	87,94%
3	0,6	87,62%	87,65%

Die drei Versuche unterschieden sich nur gering in der jeweils besten Accuracy. Die höchste Accuracy lieferte der Basisversuch mit einer Rundungsstelle von 0,5. Die Ergebnisse der anderen Metriken der rounding_boundary-Versuche waren hingegen sehr aufschlussreich um festzustellen, ab welcher Nachkommastelle für welchen Anwendungszweck am besten aufgerundet werden sollte.
Wie im nachfolgenden Diagramm zu sehen ist, stieg die Precision um über 4%, wenn bei 0,6 aufgerundet wurde. Das heißt, von den markierten Duplikaten handelte es sich bei mehr Paaren tatsächlich um Duplikate. Jedoch verringerte sich der Recall dabei um mehr als 5%, sodass demnach weniger Duplikate gefunden wurden. Das Runden bei 0,4 bewirkte genau das Gegenteil. Im Speziellen stieg der Recall um über 7%, während die Precision um mehr als 7% sank. In diesem Fall waren Precision und Recall fast gleich auf, was in einem etwas besseren F1 gegenüber dem des Basisversuchs resultierte. Sollte der Algorithmus benutzt werden, um zu einer Frage ähnliche Fragen aufzulisten, so ist es sinnvoller bei 0,6 zu runden, da es dabei nicht auf die Anzahl, sondern auf die Genauigkeit der Fragepaare ankommt.



	Accuracy	Recall	Precision	F1
round 4	87,20%	83,85%	82,04%	82,93%
round 5	87,90%	76,77%	89,08%	82,47%
round 6	87,62%	71,56%	93,53%	81,08%

Abbildung 5.3: Es stieg die Precision, wenn bei 0,6 aufgerundet wurde, jedoch verringerte sich dabei der Recall. Das Runden bei 0,4 bewirkte genau das Gegenteil.

Die Accuracy beim Runden ab 0,4 war etwas niedriger als bei 0,5; der F1 war wegen des besseren Verhältnisses von Recall und Precision dafür höher. Daher kann es später sinnvoll sein bei 0,4 anstatt bei 0,5 zu runden.

5.4.5 Optimieren der Tokenization

In diesem Abschnitt wird untersucht, welche Tokenization am besten abschneidet. Dazu kam zum einen die einfache Tokenization anhand von Leerzeichen und zum anderen der TokTokTokenizer[1] zum Einsatz. Der StanfordTokenizer wurde nicht verwendet, da er sich als zu langsam herausstellte, selbst die Verwendung eines Caches konnte die Zeit nicht drastisch verringern. Der ToktokTokenizer lieferte eine ähnliche Zuverlässigkeit in wesentlich kürzerer Zeit.
Weiterhin sollte untersucht werden, ob Lemmatization das Ergebnis weiter verbessern kann. Dies geschah unter Einsatz des WordNetLemmatizer, der jedoch keine POS-Tags als Input erhielt, da die Bestimmung dieser zu viel Zeit in Anspruch genommen hätte.
Die Features, in denen Tokenization verwendet wurde und bei denen eine Optimierung gefunden werden sollte, waren:

- einige Wordcount Features
- Shared Words Features
- TF-IDF Features

Die Versuche erreichten folgende Accuracies:

Tabelle 5.7: Bei den Versuchen zur Tokenization wurden kaum Verbesserungen festgestellt. Der Versuch, in dem Lemmatization eingesetzt wurde, erreichte die höchste Accuracy.

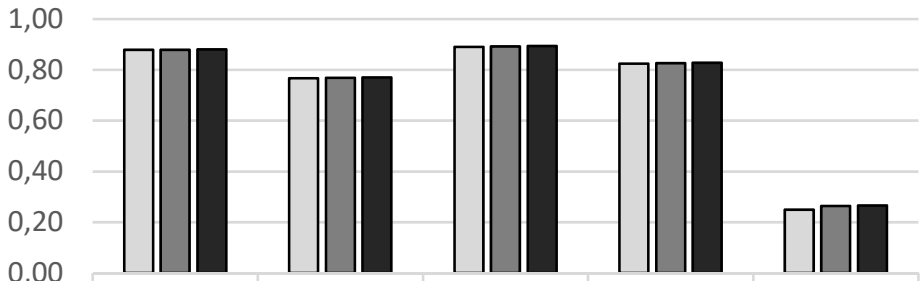
Versuch	**Optimierung**	**∅ Accuracy Top 10**	**Accuracy Top 1**
1	Leerzeichen (Basisversuch)	87,90%	87,94%
2	Tokenizer	87,98%	88,02%
3	Lemmatizer	88,09%	88,13%

In nachstehender Auflistung ist die Schnittmenge der Features der 15 besten Lemmatizer-Versuche zu sehen:

- levenshtein
- intersection
- q1_freq
- q2_freq
- tfidf_wm
- word_match_stops
- min_unique_nostops_word_count

[1] `http://nltk.org/_modules/nltk/tokenize/toktok.html`

Wie in Diagram 5.4 zu erkennen, liegen alle Werte sehr dicht beieinander, sodass es keine eindeutigen Tendenzen gibt. Ursache für die geringen Unterschiede könnte sein, dass sich die Anzahl der Wörter durch die andere Tokenzerlegung erhöht hat und somit unter anderem Fragezeichen separiert verarbeitet werden. Damit liefert die Features *tfidf_wm*, *word_match_stops* und *min_unique_nostops_word_count* etwas andere, jedoch nicht zwangsläufig bessere Werte. Ein weiterer Grund für die ähnlichen Ergebnisse der Metriken könnte sein, dass die Intersection Features (*intersect*, *q1_freq*, *q2_freq*) ebenfalls sehr maßgebend für die Accuracy waren, aber nicht von der Tokenization oder Lemmatization beeinflusst wurden.



	Accuracy	Recall	Precision	F1	Log-Loss
Leerzeichen	87,90%	76,77%	89,08%	82,47%	0,2495
Tokenize	87,98%	76,89%	89,21%	82,59%	0,2635
Lemmatize	88,09%	76,96%	89,44%	82,73%	0,2651

Abbildung 5.4: Es ist zu erkennen, dass alle Werte sehr nah beieinander liegen und es keine eindeutige Tendenz gibt.

Ein großer Unterschied des Log-Losses war ebenfalls nicht erkennbar. Es lässt sich demnach keine großartige Optimierung durch die Verwendung des TokTokTokenizers oder des WordNetLemmatizers finden.

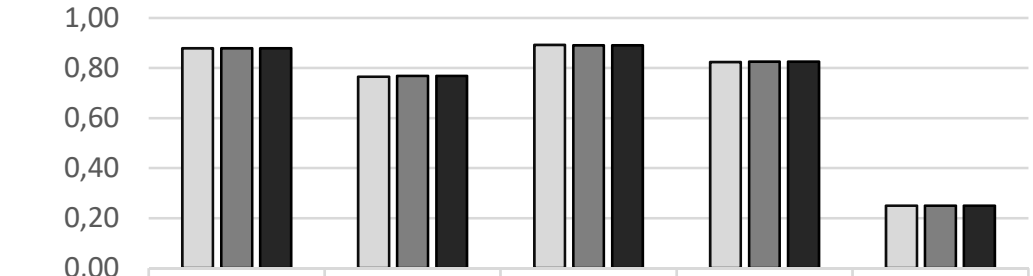
5.4.6 Optimieren des Parameters max_depth

Um festzustellen, ob das Modell sich mit wachsender Tiefe verbessert, wurde der Parameter max_depth in den Einstellungen 6,7 und 8 getestet. Zu erwarten war, dass die Accuracy mit der Baumtiefe zunimmt, da das Modell sich besser an das Validierungsset anpasst.

Tabelle 5.8: Bei den Varianten der max_depth-Versuche wurde ersichtlich, dass ein tieferer Baum zu einer höheren Accuracy führte.

Versuch	**max_depth**	**∅ Accuracy Top 10**	**Accuracy Top 1**
1	6	87,87%	87,90%
2	7 (Basisversuch)	87,90%	87,94%
3	8	87,91%	88,02%

In den drei Versuchen hat sich herausgestellt, dass je tiefer der Baum gebildet wurde, desto besser der resultierende Score war. Es wurde der Durchschnitt der besten 10 Iterationen gebildet, wodurch sich zeigte, dass sich die Accuracy von Baumtiefe 6 auf 7 um 0,03% und von 7 auf 8 nochmals um 0,01% verbesserte. Die Verbesserungen lagen damit aber unter 0,1% und sind daher als minimal einzustufen. Eine großartige Verbesserung ist mit einer Tiefe größer 8 nicht zu erwarten. Die Unterschiede wären bei mehr Iterationen wahrscheinlich deutlicher ausgefallen.
Es ergaben sich kaum Verbesserungen hinsichtlich der Bewertungsmetriken mit Zunahme der Baumtiefe:



	Accuracy	Recall	Precision	F1	Log-Loss
depth 6	87,87%	76,55%	89,21%	82,40%	0,2500
depth 7	87,90%	76,77%	89,08%	82,47%	0,2495
depth 8	87,91%	76,90%	89,02%	82,51%	0,2496

Abbildung 5.5: Die Metriken der max_depth-Versuche veränderten sich kaum durch Zunahme der Baumtiefe.

Es lässt sich festhalten, dass der optimale Wert des Parameters max_depth 8 ist, da dieser unter allen Versuchen die höchste Accuracy und den besten F1 hatte.

5.4.7 Optimieren durch Normalization

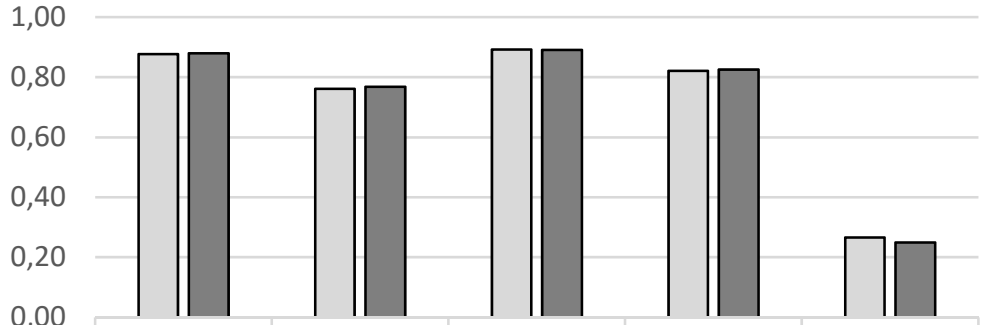
Um festzustellen, inwieweit die Groß- und Kleinschreibung der Fragen eine Rolle bei der Vorhersagung von Fragepaaren spielte, wurden die Fragen in ihrer normalen Schreibweise und als Kleinbuchstaben verarbeitet:

Tabelle 5.9: Die Versuche zur Normalization zeigten, dass eine leichte Verbesserung erfolgte, wenn die Fragen vor der Feature-Extraktion durch Case-Folding in Kleinbuchstaben umgewandelt wurden.

Versuch	**Orthografie**	**∅ Accuracy Top 10**	**Accuracy Top 1**
1	klein (Basisversuch)	87,90%	87,94%
2	normal	87,73%	87,76%

Die Versuche zur Normalization haben gezeigt, dass ein Modell eine etwas bessere Accuracy erzielt, wenn alle Fragen vor der Feature-Extraktion in Kleinbuchstaben

umgewandelt werden. Wie nachfolgend zu entnehmen ist, war die Precision bei der normalen Schreibweise höher. Der F1 und der Recall waren hingegen geringer. Die bessere Precision könnte daher kommen, dass neben anderen Features die Shared Words Features und TF-IDF-Features durch die veränderte Schreibweise anders ausfallen und im Modelltraining zu differenzierteren Baumstrukturen führen.



	Accuracy	Recall	Precision	F1	Log-Loss
Normal	87,73%	76,18%	89,17%	82,16%	0,2662
Klein	87,90%	76,77%	89,08%	82,47%	0,2495

Abbildung 5.6: Die entscheidenden Metriken (Accuracy, F1 und Log-Loss) verbesserten sich durch Anwenden von Case-Folding im geringen Maß.

Diese Versuche haben gezeigt, dass es eine leichte Optimierung war, jede Frage durch Case-Folding in Kleinbuchstaben umzuwandeln, da dadurch die Accuracy und der F1 gering anstiegen.

5.4.8 Finales Modell

Nachdem genügend Versuche durchgeführt wurden, konnten die Erkenntnisse aus diesen in dem finalen Modell vereinigt werden.

Anwenden der Optimierungen

Im letzten Versuch sollten alle gefundene Optimierungen der vorangegangenen Versuche angewendet werden, um ein möglichst erfolgreiches Modell zu erhalten. Die vorherigen Versuche haben gezeigt, dass sich das Basis-Modell leicht verbessern ließ. Es ergaben sich folgende Optimierungen:

- Case-Folding: Kleinbuchstaben
- Tokenization : ToktokTokenizer
- Lemmatization: WordNetLemmatizer
- Rundungsstelle: bei 0,4
- Tiefe des Modells: 8
- Anzahl an Boosting Runden: 2.500

Die Versuche zur Optimierung wurden gezielt nicht nacheinander angewandt, da jeder Parameter durch andere Parameter direkt beeinflusst wird und folglich keine klaren Aussagen über Verbesserungen geschlussfolgert werden hätten können.
Die Tokenizer und Lemmatizer wurden schlussendlich in das finale Modell aufgenommen, da sie dieses zumindest nicht verschlechterten und eventuell noch bessere Accuracies herbeiführten. Insgesamt konnte in allen Versuchen nicht viel am Basisversuch verbessert werden, da die größten Fortschritte durch die Auswahl an Features erzielt werden.

Auswahl an Features

Im finalen Durchgang sollten in der Feature-Selection Features, die sich als nützlich erwiesen hatten, in jeder Iteration ausgewählt werden. Ziel war es, eine noch bessere Kombination der Features zu finden.
Die Top Features, die aus den besten Versuchen der obigen Optimierungen herausgesucht wurden, waren:

- levenshtein
- intersection
- q1_freq
- q2_freq
- word_match_stops
- tfidf_wm_stops

Diese sechs Features waren in allen Iterationen Bestandteil der Feature-Selection. Außerdem wurde in jeder Iteration zufällig aus den restlichen Features ausgewählt.

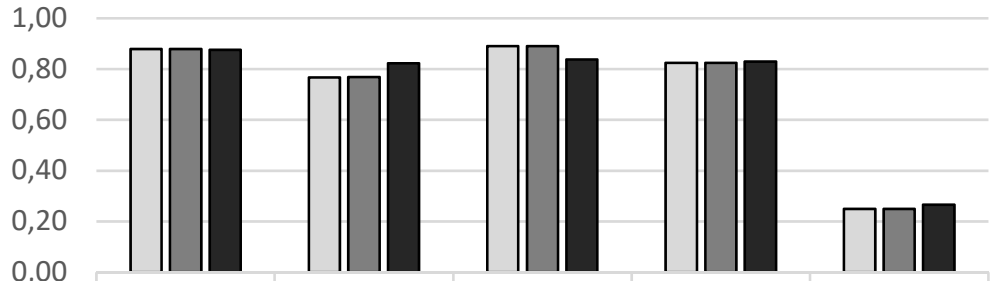
Ergebnis

In den 170 Versuchen stellte sich heraus, dass die 111. Iteration die beste Accuracy lieferte. Dabei wurden 31 Features miteinander kombiniert, sodass sich die restlichen 12 Features als wenig relevant herausstellten.

Tabelle 5.10: Die Iteration 111 erreichte unter allen Versuchen sowohl die beste Accuracy (87,51%), als auch den besten F1 (83,09%). Auffällig ist außerdem, dass ein Durchlauf mit nur 19 Features, aber ähnlich hoher Accuracy (87,51%) existierte.

Iteration	Accuracy	Recall	Precision	F1	Log-Loss	Features
111	**87,57%**	82,39%	83,81%	**83,09%**	0,2658	31
84	87,49%	**82,61%**	83,49%	83,04%	0,2648	31
163	87,51%	82,01%	**83,94%**	82,96%	0,2687	19
80	87,48%	82,35%	83,64%	82,99%	**0,2642**	34

Im Vergleich dazu hatte der beste max_depth-Versuch zwar am Ende eine etwas höhere Accuracy, benötigte aber dafür 11 Features mehr.
Wird der finale Durchlauf mit dem Basisversuch verglichen, so fällt auf, dass sich der Recall und der F1 verbessert haben. Lediglich die Accuracy und die Precision haben abgenommen, doch wie in den Round-Versuchen bereits festgestellt, ist die Rundungsstelle die Ursache dieses verhältnismäßig großen Unterschieds.



	Accuracy	Recall	Precision	F1	Log-Loss
Basis	87,90%	76,77%	89,08%	82,47%	0,2495
Tiefe 8	87,91%	76,90%	89,02%	82,51%	0,2496
Final	87,52%	82,35%	83,74%	83,04%	0,2655

Abbildung 5.7: Wie im Diagramm erkennbar ist, konnte keine bessere Accuracy, dafür aber ein besserer F1 als im Basisversuch erzielt werden. Der Versuch mit einer Modell-Tiefe von 8 überbot ebenfalls die Accuracy des finalen Versuchs, lieferte aber einen schlechteren F1.

Aus der Accuracy lässt sich ableiten, dass mindestens 8 von 10 Fragepaaren im Validierungsset erfolgreich als Duplikat beziehungsweise Nicht-Duplikat erkannt werden konnten. Von allen markierten Duplikaten sind zirka 82% tatsächlich Duplikate und die beste Iteration identifizierte über 83% der Duplikate.

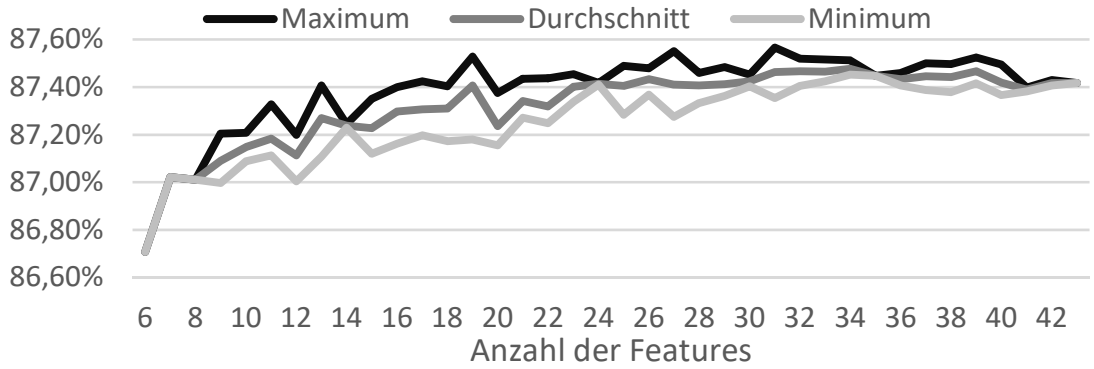


Abbildung 5.8: Die Extrema der Accuracy für die verschiedenen Anzahlen an Features zeigen, dass eine hohe Anzahl an Features nicht unbedingt zu einem bessere Ergebnis führt. Mit 40 Features war die Accuracy im Worst-Case fast so hoch wie im Best-Case mit 11 Features. Tendenziell stieg jedoch die Accuracy mit der Anzahl an Features.

Wie im vorangegangen Diagramm zu sehen war, bedeutet eine hohe Feature-Anzahl nicht automatisch, dass ein Verfahren gut ist. Beispielsweise konnte mit 11 Features im Best-Case fast die gleiche Accuracy erreicht werden, wie im Worst-Case mit 40 Features. Weiterhin erzielten 13 Features fast die gleiche Accuracy wie 41 Features. Der Unterschied zwischen den oben genannten sechs Hauptfeatures zu dem besten Versuch mit 31 Features beträgt 0,86%. Das heißt, unter Zuhilfenahme der restlichen 25 Features konnte die Accuracy nicht einmal um 1% erhöht werden.
Werden die anderen Scopes genauer betrachtet, ist erkennbar, dass der Scope 70_150 die besten Ergebnisse lieferte. Der Grund hierfür ist, dass dieser Scope mit über 70% Nicht-Duplikaten sehr unbalanciert ist. Bei längeren Fragepaaren ist die Levenshtein-Distanz zu groß, weswegen dort die Wahrscheinlichkeit höher ist, dass der Algorithmus Nicht-Duplikate als solche erkennt.

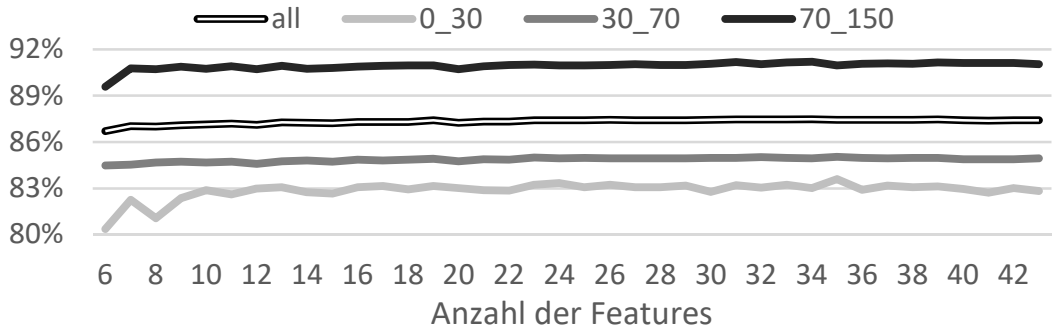


Abbildung 5.9: Die Accuracy für den Scope 70_150 war deutlich höher als die anderen Scores, was in dem unausgeglichenen Verhältnis von Duplikaten und Nicht-Duplikaten begründet ist. Für die Scopes 0_30 und 30_70 wurde eine niedrigere Accuracy erreicht als für das gesamte Validierungsset. Bei 28 Features wurde die beste Accuracy aller Scopes erzielt.

Die beste Iteration wurde ebenfalls auf das Testset angewandt. Der Algorithmus erreichte dabei einen Log-Loss von 0,5528. Dieser Wert lag jedoch weit unter dem ersten Platz der Competition (0,1158) und dem Competition-Durchschnitt (0,3410), wie im Diagramm zu sehen ist. Die Ursachen könnten sein, dass die Features am Ende zu wenige oder zu unaussagekräftig waren, oder dass in dieser Arbeit vorrangig die Accuracy anstelle des Log-Losses als Bewertungsmaßstab verwendet wurde.

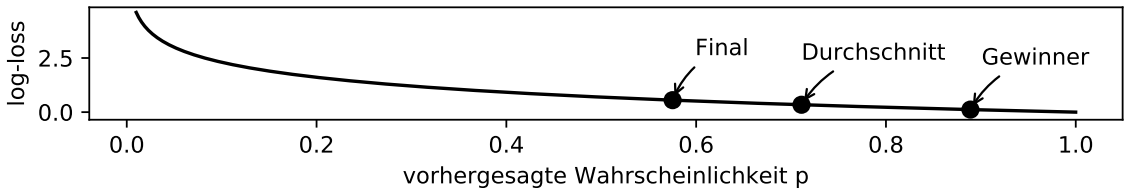


Abbildung 5.10: Der Log-Loss für die beste finale Iteration liegt deutlich unter dem Competition-Durchschnitt und dem Log-Loss der Gewinner.

Das nachfolgende Diagramm zeigt, wie die einzelnen Features bei der besten Iteration im Modell priorisiert wurden. Es ist deutlich erkennbar, dass das *Levenshtein Feature* das wichtigste Feature im ganzen Modell darstellt. Weiterhin ist ersichtlich, dass die *Word Count Features* weiter unten priorisiert sind. Es kann beobachtet werden, dass die *TF-IDF Feature*s sowohl mit, als auch ohne Stoppwörter benutzt wurden. Die *Wordmatch Features* waren jedoch nur ohne das Einbeziehen von Stoppwörtern sinnvoll. Die Jaccard-Distanz aus den *Shared Words Features* fand ebenfalls keine Verwendung.

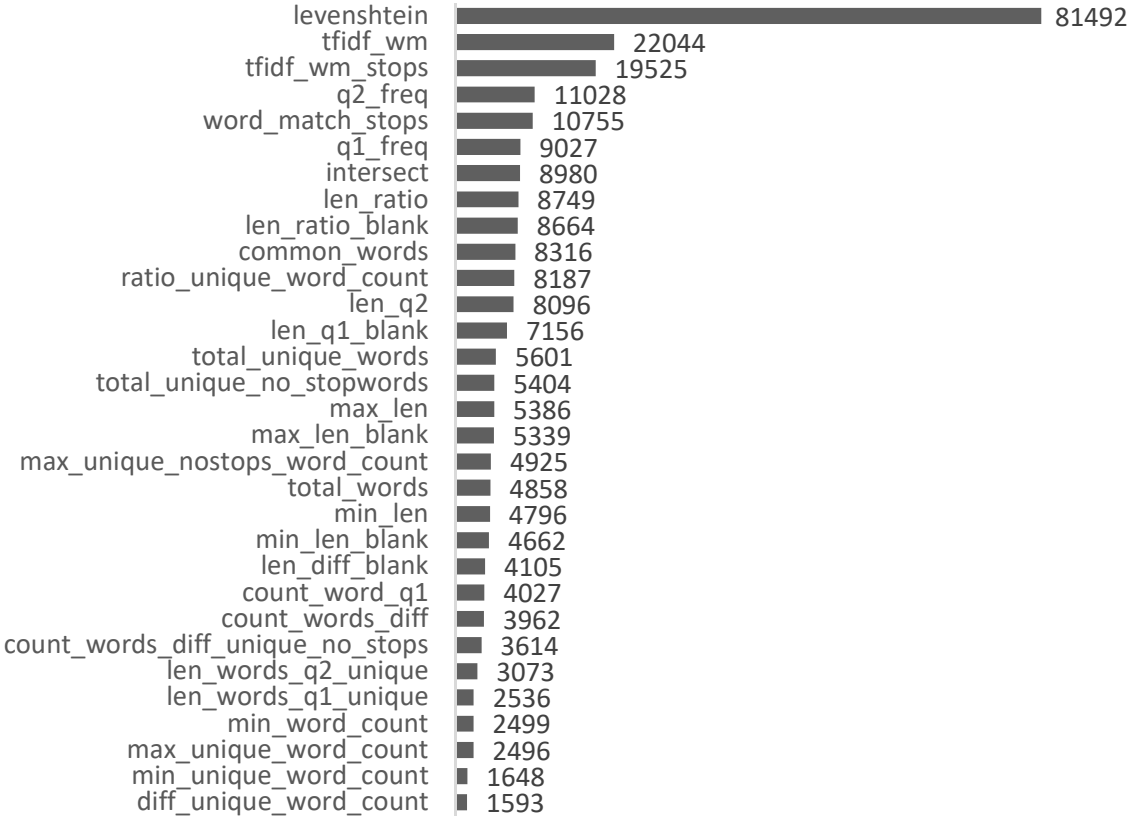


Abbildung 5.11: Das Diagramm zeigt, wie die einzelnen Features bei der besten Iteration priorisiert wurden. Die ersten drei Plätze, belegt von den Features *levenshtein*, *tfidf_wm* und *tfidf_stops* stechen deutlich aus der Masse heraus. Es ist ebenfalls erkennbar, dass viele Features mit *min* und *max* vertreten sind. Insgesamt waren alle *Intersection Features*, *TF-IDF Features* und das *Levenshtein Feature* enthalten.

Die *TF-IDF Features*, das *Levenshtein Feature* und die *Intersection Features* waren im finalen Modell vollständig vertreten. Außerdem fanden von den *Word Count Features* deutlich mehr Features mit *min/max* im besten Lauf Verwendung, als Features mit *q1/q2*. Damit hat sich gezeigt, dass es von Vorteil ist, die Minima und Maxima zu bilden. Ebenfalls waren die Features *len_ratio* und *len_ratio_blank* dabei, woraus erkennbar ist, dass es sinnvoll ist, Features inklusive und exklusive Leerzeichen zu erstellen.

5.5 Auszug aus den klassifizierten Validierungsdaten

Die in 2.1 vorgestellten Fragepaare des Trainingssets waren nicht im Validierungsset enthalten, sodass am Ende nicht untersucht werden konnte, wie das Verfahren bei diesen Fragepaaren funktioniert hätte. Deswegen werden in dieser Sektion einige Fragepaare des Validierungssets herausgenommen, um die Performanz des finalen Modells detaillierter einzuschätzen zu können. Dazu werden zuerst richtig klassifizierte und anschließend falsch klassifizierte Paare analysiert. Zum Schluss folgen Fragepaare, die von Quora falsch, aber vom finalen Modell richtig klassifiziert wurden.

5.5.1 Richtig vorhergesagte Fragepaare

Es folgen einige Fragepaare des Validierungssets, die das finale Modell korrekt klassifizierte.

Nicht-Duplikat 134965

- Frage 1: What are some things new employees should know going into their first day at Carter's?
- Frage 2: What are some things new employees should know going into their first day at AT&T?

Dieses Nicht-Duplikat wurde korrekt klassifiziert, trotz der hohen Anzahl an gemeinsamen Worten. Da die Word Count Features niedriger priorisiert waren als die TF-IDF Features, erkannte der Algorithmus, dass „Carter's" und „AT&T" nicht gleich sind und es sich deswegen um kein Duplikat handelt.

Duplikat 95513

- Frage 1: As I begin to take in that Donald Trump has been elected President, do you think he may actually be a good president?
- Frage 2: Would Donald Trump be a good president?

Das Duplikat mit der Id 95513 wurde richtig erkannt. Dies könnte daran liegen, dass „Donald Trump" ein Term mit einem hohen TF-IDF Maß ist, welcher in beiden Fragen vorkommt. Außerdem sind bis auf ein Wort von Frage 2 alle Wörter ebenfalls in Frage 1 vorhanden. Dies resultiert in einem hohen Wert im **common_words**-Feature.

5.5.2 Falsch vorhergesagte Fragepaare

Nachfolgend werden zwei Fragepaare näher betrachtet, die durch das finalen Modell nicht richtig klassifiziert wurden.

Nicht-Duplikat 52213

- Frage 1: What are the funniest moments in 2016 Rio Olympics?
- Frage 2: What heartbreaking moments from Rio Olympics 2016 will be remembered forever?

Dadurch, dass die TF-IDF-Features unter den drei am höchsten priorisierten Features waren, konnten Fragepaare wie dieses nicht richtig erkannt werden. Die Wörter „Rio“, „Olympics“ und „2016“ besaßen ein hohes Maß im Vergleich zu den anderen Worten und das Verfahren wies fälschlicherweise die Klasse {Duplikat} zu, da in beiden Fragen diese drei Worte vorkamen.

Duplikat 294377

- Frage 1: What are the ways to protect my smartphone from viruses?
- Frage 2: Can a smartphone get viruses? If so, how do you prevent or remove them?

Dieses Duplikat wurde ebenfalls nicht richtig von dem Verfahren erkannt. Trotz vieler gemeinsamer Wörter mit hohem TF-IDF Wert, wie „viruses“ oder „smartphone“, überwog die Anzahl an verschiedenen Worten und das Verfahren klassifizierte das Paar als Nicht-Duplikat.

5.5.3 Falsch klassifizierte Fragepaare

Weiterhin gab es Fragepaare, die von Quora falsch klassifiziert, aber von dem Verfahren korrekt erkannt wurden.

Nicht-Duplikat 355888

- Frage 1: Why is lightning often accompanied by thunder?
- Frage 2: How does lightning happen?

Der Fokus der ersten Frage liegt auf dem Donner als Begleiterscheinung von Blitzen, die zweite Frage beschäftigt sich dagegen mit der Ursache dieser. Demnach handelt es sich nicht um ein Duplikat.

Duplikat 156864

- Frage 1: Laptop Recommendation: Which Laptop should I go for?
- Frage 2: Laptop Recommendation: Which laptop should go for?

Die Anzahl und Reihenfolge an gleichen Worten stimmt bei diesem Paar überein, weswegen dieses ein Duplikat ist.
Ein möglicher Ansatz, um das Training zu verbessern, könnte sein, die falsch klassifizierten Fragepaare mit Hilfe des Verfahrens zu korrigieren, damit eine bessere Evaluation stattfinden kann.

5.6 Fazit

Es konnte gezeigt werden, dass mit Hilfe von maschinellem Lernen ein Fragepaar mit einer Accuracy von über 87% als Duplikat oder Nicht-Duplikat erkannt werden konnte. Dazu war es nicht nötig, komplexe neuronale Netze oder eine Unzahl an Features zu verwenden; dieses gute Ergebnis konnte bereits durch einfache Grundlagen des Textvergleichs erzielt werden. Die Levenshtein-Distanz stellte das mit Abstand wichtigste Feature des Fragevergleichs dar. Die Termfrequenz und die Inverse Dokumentfrequenz spielten ebenso eine wichtige Rolle. Insgesamt wurden im besten finalen Versuch Features aus allen fünf Feature-Gruppen verwendet. Es zeigte sich, dass es sinnvoll ist, die Stoppwörter zu entfernen und einen Tokenizer und Lemmatizer in Kombination zu benutzen. Auch das Anwenden von Case-Folding wies sich als hilfreich heraus. Für die wichtigsten Fragepaare, welche im Scope 30_70 evaluiert wurden, konnte eine Accuracy von über 85% erzielt werden.
Die Benutzung von Random Search war von Vorteil, da tatsächlich nur einige wenige Features die meiste Arbeit leisteten und so viel besser nach den besten Feature-Kombinationen gesucht werden konnte.
Leider erreichte das finale Modell keinen guten Log-Loss bezüglich des Testsets, aber der Fokus dieser Bachelorarbeit lag hauptsächlich auf einem guten Verfahren für das Trainingsset. Die Evaluation könnte für das Testset im gleichen Maße wiederholt werden, nur dass zur Einschätzung der besten Features und des besten Verfahrens der Log-Loss als Hauptmetrik benutzt wird.

5.6.1 Weitere Möglichkeiten

Neben den genannten Möglichkeiten zur Verfahrensentwicklung hätten noch weitere Methoden angewandt werden können. Darunter fallen zum Beispiel das Einbeziehen von POS-Tags in die Lemmatization oder das Normalisieren der Fragen durch Textersetzungen (4.3.1). Anstatt des Lemmatizers könnten auch Stemmer wie der PorterStemmer verwendet werden. Es hätten auch die Köpfe der Fragen miteinander verglichen und die Datenanalyse ausgedehnt werden können. Um den Log-Loss zu verbessern könnte das in 4.3.5 vorgestellte Rundungsverfahren getestet werden.
Das Diagramm zu den Feature-Prioritäten im finalen Modell (5.11) zeigte auf, dass es sich als vorteilhaft auswirken könnte, alle Features nur unabhängig ihrer Position im Fragepaar zu vergleichen.
Es hätten noch weitere Parameter von XGBoost geändert werden können. Dazu gehören zum Beispiel *scale_pos_weight*, um die Anzahl der Duplikate an die der Nicht-Duplikate anzugleichen oder *eval_metrik*, um während des Trainings mit der Accuracy zu evaluieren. Ebenfalls könnten mehr als 170 Iterationen durchgeführt werden. Anstelle von XGBoost hätte auch ein anderes System des maschinellen Lernens verwendet werden können, wie zum Beispiel LightGBM[2]. Mehrere Modelle hätten auch durch Pooling kombiniert werden können [WFHP16, S. 444].

[2] `https://github.com/Microsoft/LightGBM`

6 Ausblick

Die computergestützte Textverarbeitung ist noch nicht vollständig erforscht, weswegen vor allem die Optimierung neuronaler Netze heutzutage immer mehr im Vordergrund steht. Die drei Gewinner[1][2][3] der Competition von Quora verwendeten ebenfalls neuronale Netze. Zudem existieren weitere Forschungsbereiche in der Sprachverarbeitung, wie zum Beispiel das automatische Beantworten von Fragen, maschinelles Übersetzen von Texten oder die Erkennung von Spam in E-Mails [KKKS17].
Der Fokus soll künftig weg von der Informationsextraktion und hin zu einer verbesserten Kommunikation mit dem Computer gelenkt werden. Dazu zählt ein computerseitiges Textverständnis durch Anwendung von domänenspezifischem Wissen beziehungsweise Allgemeinwissen. Außerdem sollen zukünftig mehr Informationen aus Online-Ressourcen bezogen werden, um es Computern zu ermöglichen, Features selbstständig zu definieren. Dies soll im gleichen Maße geschehen, wie es heutzutage eigenhändig von Data Scientists betrieben wird. Die Motivation dahinter ist, dass für jedes neue Problem neue Features gefunden werden müssen, welche oft zu kompliziert oder unvollständig sind und deren Suche viel Zeit in Anspruch nimmt. [FRC13, S. 766][SM13, KSSZ16]
Da es sich bei den Datensets von Quora nur um englische Fragen handelt, stellt sich die Frage, inwieweit der Algorithmus auf andere Sprachen anwendbar ist. Beispielsweise müssen Wörter im Chinesischen oder Japanischen anders segmentiert werden, da Leerzeichen und Interpunktion gänzlich fehlen [X+03]. Für Sprachen, die von rechts nach links gelesen werden, wie Arabisch, muss das Verfahren, im Speziellen das POS-Tagging, ebenfalls angepasst werden [AR15].
Besonders auf Facebook wird gezielt in der computergestützten Sprachverarbeitung geforscht, um zum Beispiel unerwünschte Inhalte unter der Vielzahl an Sprachen besser zu identifizieren[4]. Ein Ansatz, der auch beim Vergleich von Fragen sinnvoll wäre, ist die Verwendung von Word Embeddings, die in tiefgehenden neuronalen Netzen genutzt werden [MSC+13].
Der ursprünglich von Quora genutzte Algorithmus konnte schlussendlich verbessert werden. Vielleicht ist es in näherer Zukunft nicht mehr nötig, Kaggle-Competitions zu starten, wenn der Computer den Großteil der Arbeit erledigen kann, die zurzeit Data Scientists und Machine Learning Engineers beschäftigt.

[1] https://kaggle.com/c/quora-question-pairs/discussion/34355
[2] https://kaggle.com/c/quora-question-pairs/discussion/34310
[3] https://kaggle.com/c/quora-question-pairs/discussion/34288
[4] https://code.facebook.com/posts/181565595577955/introducing-deeptext-facebook-s-text-understanding-engine/

Literaturverzeichnis

[All95] James Allen: *Natural language understanding*, Benjamin/Cummings Publ., Redwood City, Calif. [u.a.], 2. ed. Aufl., 1995, ISBN 0805303340.

[AR15] Fahad Albogamy und Allan Ramsay: *POS Tagging for Arabic Tweets.*, in *RANLP*, S. 1–8, 2015.

[Bak95] Carl Lee Baker: *English syntax*, MIT-Pr., Cambridge, Mass. [u.a.], 2. ed. Aufl., 1995, ISBN 0262023857.

[BB12] James S. Bergstra und Yoshua Bengio: *Random search for hyper-parameter optimization*, *Journal of Machine Learning Research*, Bd. 13(Feb):S. 281–305, 2012.

[BBBK11] James S. Bergstra, Rémi Bardenet, Yoshua Bengio und Balázs Kégl: *Algorithms for hyper-parameter optimization*, in *Advances in Neural Information Processing Systems*, S. 2546–2554, 2011.

[Bir06] Steven Bird: *NLTK: The Natural Language Toolkit*, in *Proceedings of the COLING/ACL on Interactive Presentation Sessions*, COLING-ACL '06, S. 69–72, Association for Computational Linguistics, Stroudsburg, PA, USA, 2006.

[BL+07] James Bennett, Stan Lanning *et al.*: *The netflix prize*, in *Proceedings of KDD cup and workshop*, Bd. 2007, S. 35, New York, NY, USA, 2007.

[BNJ03] David M. Blei, Andrew Y. Ng und Michael I. Jordan: *Latent dirichlet allocation*, *Journal of machine Learning research*, Bd. 3(Jan):S. 993–1022, 2003.

[Bre72] Joan W. Bresnan: *Theory of complementation in English syntax.*, Dissertation, Massachusetts Institute of Technology, 1972.

[Bri00] Eric Brill: *Part-of-speech tagging*, *Handbook of natural language processing*, S. 403–414, 2000.

[BYR01] Ricardo Baeza-Yates und Berthier de Araújo Neto Ribeiro: *Modern information retrieval .*, ACM Press, New York, NY, [repr.] Aufl., 2001, ISBN 020139829X.

[CEE+10] Kai-Uwe Carstensen, Christian Ebert, Cornelia Ebert, Susanne Jekat, Hagen Langer und Ralf Klabunde: *Computerlinguistik und Sprachtechnologie: Eine Einführung*, Springer-Verlag, 2010.

[CG16] Tianqi Chen und Carlos Guestrin: *Xgboost: A scalable tree boosting system*, in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, S. 785–794, ACM, 2016.

[Cho02] Noam Chomsky: *Syntactic structures*, Walter de Gruyter, 2002.

[CT+94] William B. Cavnar, John M. Trenkle *et al.*: *N-gram-based text categorization*, *Ann Arbor MI*, Bd. 48113(2):S. 161–175, 1994.

[Emo76] Joseph E. Emonds: *A transformational approach to English syntax: Root, structure-preserving, and local transformations*, Academic Press Inc, 1976.

[ES02] Brian Everitt und Anders Skrondal: *The Cambridge dictionary of statistics*, Bd. 106, Cambridge University Press Cambridge, 2002.

[FRC13] Carol Friedman, Thomas C. Rindflesch und Milton Corn: *Natural language processing: State of the art and prospects for significant progress, a workshop sponsored by the National Library of Medicine*, *Journal of Biomedical Informatics*, Bd. 46(5):S. 765 – 773, 2013, ISSN 1532-0464.

[HCL+03] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin *et al.*: *A practical guide to support vector classification*, 2003.

[HLST15] Frank Hutter, Jörg Lücke und Lars Schmidt-Thieme: *Beyond manual tuning of hyperparameters*, *KI-Künstliche Intelligenz*, Bd. 29(4):S. 329–337, 2015.

[HTF09] Trevor Hastie, Robert Tibshirani und Jerome Friedman: *Overview of supervised learning*, in *The elements of statistical learning*, S. 9–41, Springer, 2009.

[ID10] Nitin Indurkhya und Fred J Damerau: *Handbook of natural language processing*, Bd. 2, CRC Press, 2010.

[Jac01] Paul Jaccard: *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*, *Bull Soc Vaudoise Sci Nat*, Bd. 37:S. 547–579, 1901.

[JE13] Yangfeng Ji und Jacob Eisenstein: *Discriminative Improvements to Distributional Sentence Similarity.*, in *EMNLP*, S. 891–896, 2013.

[JM00] Dan Jurafsky und James H. Martin: *Speech and language processing : An introduction to natural language processing, computational linguistics, and speech recognition*, Prentice Hall, Upper Saddle River, NJ, intern. ed. Aufl., 2000, ISBN 013122798X.

[KKKS17] Diksha Khurana, Aditya Koli, Kiran Khatter und Sukhdev Singh: *Natural Language Processing: State of The Art, Current Trends and Challenges*, *arXiv preprint arXiv:1708.05148*, 2017.

[Koh95] Ron Kohavi: *A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection*, S. 1137–1143, Morgan Kaufmann, 1995.

[Kro93] Robert Krovetz: *Viewing morphology as an inference process*, in *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, S. 191–202, ACM, 1993.

[KSSZ16] Mirjana Kocaleva, Done Stojanov, Igor Stojanovic und Zoran Zdravev: *Pattern Recognition and Natural Language Processing: State of the Art*, *TEM Journal*, Bd. 5(2):S. 236–240, 2016.

[Lan96] Pat Langley: *Elements of machine learning*, Morgan Kaufmann Publ., San Francisco, Calif., 1996, ISBN 1558603018.

[Lev66] Vladimir I. Levenshtein: *Binary codes capable of correcting deletions, insertions, and reversals*, in *Soviet physics doklady*, Bd. 10, S. 707–710, 1966.

[Man08] Christopher Manning: *Introduction to information retrieval*, Cambridge University Press, New York, 2008, ISBN 0521865719.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean: *Efficient estimation of word representations in vector space*, *arXiv preprint arXiv:1301.3781*, 2013.

[MDA15] Dougal Maclaurin, David Duvenaud und Ryan Adams: *Gradient-based hyperparameter optimization through reversible learning*, in *International Conference on Machine Learning*, S. 2113–2122, 2015.

[Mit04] Ruslan Mitkov: *The Oxford handbook of computational linguistics* ., Oxford Univ. Press, Oxford [u.a.], 1. publ. in paperback Aufl., 2004, ISBN 9780199276349.

[MSC+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado und Jeff Dean: *Distributed representations of words and phrases and their compositionality*, in *Advances in neural information processing systems*, S. 3111–3119, 2013.

[Mus16] Tauno F. Mustanoja: *A Middle English Syntax: Parts of Speech*, John Benjamins Publishing Company, 2016.

[Por80] Martin F. Porter: *An algorithm for suffix stripping*, *Program*, Bd. 14(3):S. 130–137, 1980.

[Pow11] David Martin Powers: *Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation*, 2011.

[RN04] Stuart J. Russell und Peter Norvig: *Künstliche Intelligenz : Ein moderner Ansatz*, Pearson Studium, München [u.a.], 2. aufl. Aufl., 2004, ISBN 9783827370891, dt. Übers. der 2. engl. Ausg.

[RWY11] G. Raskutti, M. J. Wainwright und B. Yu: *Early stopping for non-parametric regression: An optimal data-dependent stopping rule*, in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, S. 1318–1325, Sept 2011.

[SBC+01] Richard Sproat, Alan W. Black, Stanley Chen, Shankar Kumar, Mari Ostendorf und Christopher Richards: *Normalization of non-standard words*, *Computer Speech & Language*, Bd. 15(3):S. 287 – 333, 2001, ISSN 0885-2308.

[SF95] K. Srinivasan und D. Fisher: *Machine learning approaches to estimating software development effort*, *IEEE Transactions on Software Engineering*, Bd. 21(2):S. 126–137, Feb 1995, ISSN 0098-5589.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever und Ruslan Salakhutdinov: *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*, *Journal of Machine Learning Research*, Bd. 15:S. 1929–1958, 2014.

[SLA12] Jasper Snoek, Hugo Larochelle und Ryan P. Adams: *Practical bayesian optimization of machine learning algorithms*, in *Advances in neural information processing systems*, S. 2951–2959, 2012.

[SM13] R. Socher und C. Manning: *Deep learning for natural language processing (without magic)*, in *Keynote at the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL 2013)*, 2013.

[TKMS03] Kristina Toutanova, Dan Klein, Christopher D. Manning und Yoram Singer: *Feature-rich part-of-speech tagging with a cyclic dependency network*, in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, S. 173–180, Association for Computational Linguistics, 2003.

[VR79] CJ Van Rijsbergen: *Information retrieval. dept. of computer science, university of glasgow*, *URL: citeseer. ist. psu. edu/vanrijsbergen79information. html*, Bd. 14, 1979.

[WFHP16] Ian H. Witten, Eibe Frank, Mark A. Hall und Christopher J. Pal: *Data Mining: Practical machine learning tools and techniques*, Morgan Kaufmann, 2016.

[WS92] W. John Wilbur und Karl Sirotkin: *The automatic identification of stop words*, *Journal of information science*, Bd. 18(1):S. 45–55, 1992.

[X+03] Nianwen Xue *et al.*: *Chinese word segmentation as character tagging*, *Computational Linguistics and Chinese Language Processing*, Bd. 8(1):S. 29–48, 2003.

[YRC07] Yuan Yao, Lorenzo Rosasco und Andrea Caponnetto: *On Early Stopping in Gradient Descent Learning*, *Constructive Approximation*, Bd. 26(2):S. 289–315, Aug 2007, ISSN 1432-0940.

[ZY05] Tong Zhang und Bin Yu: *Boosting with early stopping: Convergence and consistency*, *Ann. Statist.*, Bd. 33(4):S. 1538–1579, 08 2005.

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Studentenservice – Zentrales Prüfungsamt

Selbstständigkeitserklärung

Name: Taubert Vorname: Stefan geb. am: 30.07.1994 Matr.-Nr.: 369897	**Bitte beachten:** 1. Bitte binden Sie dieses Blatt am Ende Ihrer Arbeit ein.

Selbstständigkeitserklärung*

Ich erkläre gegenüber der Technischen Universität Chemnitz, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch nicht als Prüfungsleistung eingereicht und ist auch noch nicht veröffentlicht.

Datum: 21.12.2017 Unterschrift: ……

* Statement of Authorship

I hereby certify to the Technische Universität Chemnitz that this thesis is all my own work and uses no external material other than that acknowledged in the text.

This work contains no plagiarism and all sentences or passages directly quoted from other people's work or including content derived from such work have been specifically credited to the authors and sources.

This paper has neither been submitted in the same or a similar form to any other examiner nor for the award of any other degree, nor has it previously been published.