Merkmalsextraktion bei Nachrichtenartikeln zur Themenklassifikation am Beispiel von Identitätsdatendiebstahl

Bachelorarbeit

ausgearbeitet von

Gina Caroline Muuss

3104947

zur Erlangung des akademischen Grades

Bachelor of Science (B.Sc.)

vorgelegt an der

Rheinische Friedrich-Wilhelms-Universität Bonn

Institut für Informatik IV

Arbeitsgruppe für IT-Sicherheit

im Studiengang

Informatik (B.Sc.)

Erstprüfer:	Dr. Felix Boes	Universität Bonn
Zweitprüfer:	Prof. Dr. Peter Martini	Universität Bonn
Betreuer:	Timo Malderle & Dr. Felix Boes	Universität Bonn

Bonn, 15. Juni 2020

Inhaltsverzeichnis

1. Einleitung 1

2. Vorstellung der Grundlagen 3
   2.1. Dokumentklassifizierung . . . 3
      2.1.1. Preprocessing . . . 3
      2.1.2. Feature Selection . . . 4
      2.1.3. Feature Vektoren . . . 4
      2.1.4. Klassifizierung . . . 5
   2.2. Maschinelles Lernen . . . 5
      2.2.1. Support Vector Machines . . . 5
      2.2.2. Fehlerarten . . . 6
      2.2.3. Performance Metriken . . . 7
      2.2.4. Kreuzvalidierung . . . 8
      2.2.5. Gittersuche . . . 8

3. Related Work 9
   3.1. Preprocessing . . . 9
   3.2. Feature Selection . . . 10
   3.3. Dokumentklassifizierung . . . 12
   3.4. Anwendung . . . 13

4. Evaluierte Lösungsansätze 15
   4.1. Preprocessing . . . 15
      4.1.1. Stemming . . . 15
      4.1.2. Lemmatisierung . . . 16
      4.1.3. Entfernung bestimmter Wörter . . . 16
      4.1.4. Satzzeichen-Entfernung und Zahlen-Ersetzung . . . 16
   4.2. Feature Selection . . . 17
      4.2.1. Alle Features . . . 17
      4.2.2. Hauptkomponentenanalyse . . . 17
      4.2.3. Relative Descrimination Criterion . . . 18
      4.2.4. Multivariate Relative Discriminative Criterion . . . 18
      4.2.5. Gewichtungsverfahren . . . 18

4.3. Klassifizierung 19

5. Evaluationsverfahren **21**
5.1. Verwendete Datensätze 22
5.2. Gewählte Parameter 22
5.3. Verwendete Metriken 22

6. Ergebnisse **23**
6.1. Betrachtung der Verfahrensschritte 23
6.2. Betrachtung nach Metrik 28

7. Zusammenfassung **31**

Literatur **35**

Abbildungsverzeichnis **38**

Tabellenverzeichnis **40**

A. Ergebnisse **45**

B. Grafiken **59**

1 Einleitung

In vielen Bereichen der digitalen Welt ist es wichtig, die Identität von Nutzern eindeutig authentifizieren zu können, damit nur berechtigte Personen Zugriff auf vertrauliche Informationen erhalten. Beispiele hierfür sind Online-Banking, soziale Medien, und Kundenkonten. Zu diesem Zweck werden meist vertrauliche Informationen verwendet, die den Nutzer authentifizieren sollen. Der Verlust der Vertraulichkeit dieser Information stellt ein Problem dar, da dann die Eindeutigkeit der Authentifizierung nicht mehr gegeben ist. Ein Beispiel für den Verlust der Vertraulichkeit findet sich im Jahr 2019 beim US-amerikanischen Unternehmen „Facebook“. Nach Berichten der Nachrichtenagentur CBS wurden hier über 540 Millionen Zugangsdaten öffentlich bekannt [Sil].

Im Fall eines solchen Verlustes der Vertraulichkeit der Identitätsdaten, ist es relevant, Nutzer zu informieren, damit sie Maßnahmen zum Schutz ihrer Daten ergreifen können. Diese Benachrichtigungen sind zeitkritisch, da Angreifer eine gewisse Zeit brauchen, um fremde Identitätsdaten auszunutzen [Mal+18]. Werden die Nutzer also mit einem nur geringen Zeitverzug informiert, bietet dies die Möglichkeit für Schutzmaßnahmen und die Wahrscheinlichkeit für einen Missbrauch der Daten wird geringer.

Nun verfügen nicht alle Nutzer über das Expertenwissen oder haben die Zeit, ständig eine Überprüfung der Sicherheit ihrer Daten durchzuführen. Daher wäre es wünschenswert, in der Lage zu sein, Nutzer über einen sie betreffenden Identitätsdiebstahl zeitnah zu benachrichtigen.

Um Nutzer benachrichtigen zu können, müssen zuvor Daten über Identitätsdiebstähle gesammelt werden. Um Identitätsdiebstähle zu finden, sollen Online-Artikel, wie zum Beispiel der oben genannte Artikel von CBS, durchsucht werden. In 2013 hat alleine der *Europe Media Monitor (EMM)* täglich durchschnittlich 175 000 Online-Nachrichtenartikel gesammelt [Ste13]. Diese Informationsdichte macht eine manuelle Detektion, in einem angemessenen zeitlichen Rahmen, unmöglich. Daher wird ein automatisiertes System zur Erkennung benötigt. Diese Nachrichtenartikel sind in natürlicher Sprache geschrieben, was eine maschinelle Klassifikation nicht trivial macht. Des wegen sollen Verfahren des *Natural Language Processing (NLP)* verwendet werden, um die Nachrichtenartikel, die über Identitätsdiebstähle berichten, von anderen zu unterscheiden. Die so identifizierten Artikel werden dann einem Experten präsentiert, der entscheidet, ob es sich wirklich um einen Identitätsdiebstahl handelt und, falls nötig, die dazugehörigen Identitätsdaten recherchiert. Gefundenen Daten sollten dann zur Benachrichtigung der betroffenen Nutzer verwendet werden.

Das Ziel der Arbeit ist, aus vorhandenen Nachrichtenartikeln *Feature Vektoren* zu extrahieren, so dass danach ein Verfahren des maschinellen Lernens die Artikel klassifizieren kann. Hierbei sollen möglichst alle relevanten Artikel gefunden werden. Als zweitrangiges Ziel sollen möglichst alle irrelevanten Artikel aussortiert werden. Die Forschungsfrage lautet somit:

> Welches Verfahren zur Merkmalsextraktion aus Nachrichtenartikeln eignet sich, um die Artikel mithilfe von maschinellem Lernen möglichst genau dahingehend zu klassifizieren, ob sie über einen Identitätsdiebstahl berichten?

Um die Beantwortung der Frage zu ermöglichen, wird sie in mehrere Teilziele aufgeteilt. Zunächst werden Verfahren der aktuellen Forschung betrachtet. Daraufhin werden einzelne Verfahren ausgewählt und anhand eines Beispieldatensatzes evaluiert. Aufgrund dieser Ergebnisse wird dann eine Empfehlung für ein einzusetzendes System gegeben.

Diese Arbeit soll verschiedene Verfahren des *Natural Language Processings* vergleichen und ihre Performance zur oben genannten Fragestellung evaluieren. Hierzu werden zunächst in Kapitel 2 einige Grundlagen des NLP und maschinellen Lernens vorgestellt. Daraufhin wird der aktuelle Stand der Forschung in Kapitel 3 dargestellt und diese Arbeit wird in den wissenschaftlichen Kontext eingeordnet. Danach werden in Kapitel 4 die zur Evaluation ausgewählten Ansätze erläutert. In Kapitel 5 wird das Evaluationsverfahren beschrieben und in Kapitel 6 werden die Ergebnisse diskutiert.

2 Vorstellung der Grundlagen

Das Thema dieser Arbeit liegt im Bereich des Natural Language Processing, mit Spezialisierung auf *Dokumentklassifizierung*. Im Folgenden werden einige für diese Arbeit relevanten Grundlagen des Natural Language Processings und des maschinellen Lernens erläutert. Hierzu gehören die nötigen Metriken zur Auswertung und die Grundlagen der Verarbeitung der Texte.

2.1 Dokumentklassifizierung

Das Problem der *Dokumentklassifizierung* wird meist in mindestens drei Unterprobleme aufgeteilt [Bah+18]. Zunächst wird ein sogenanntes *Preprocessing* angewendet, das die Eingabetexte auf die weitere Verarbeitung vorbereitet [Bah+18]. Das anschließende Teilproblem wird als *Feature Extraction* bezeichnet. Hierbei werden aus diesen Texten Features generiert, die den Text repräsentieren [Bah+18]. Meist handelt es sich um einzelne Wörter oder Wortgruppen. Eine Teilmenge dieser Features wird ausgewählt und an den Algorithmus zur Klassifizierung übergeben [Bah+18]. Dann findet die eigentliche Klassifizierung statt [Bah+18]. Dies ist das letzte Teilgebiet.

2.1.1 Preprocessing

Um Texte verarbeiten zu können, werden diese zu Beginn des Verfahrens vorverarbeitet. Dieser Vorgang wird *Preprocessing* genannt [UG14]. Beispiele für solche Verfahren sind das Konvertieren zu Kleinbuchstaben oder das Entfernen von Satzzeichen. Es werden aber auch komplexere Transformationen, wie die im Folgenden erläuterten, verwendet. Zudem generiert das *Preprocessing* aus der Eingabe eines Textes sogenannte Tokens, also einzelne Wörter oder Wortgruppen [UG14].

Stemming bezeichnet die Praxis, Suffixe von Wörter zu entfernen [SMR08]. Hierfür werden die Wörter ohne ihre Bedeutung betrachtet und es werden aufgrund von konstruierten Regeln Wortteile entfernt [SMR08]. So wird der *Feature Raum* deutlich verkleinert, da zum Beispiel aus den Wörtern „abducting“ und „abduction“ der Präfix „abduct“ rekonstruiert wird. Hierbei soll eine Reduzierung auf Wortstämme erreicht werden, jedoch macht eine Verwendung solcher Regeln Fehler, da das System das Konzept von Wortstämmen nicht kennt. So kann es zum Beispiel passieren, dass das System aus dem Wort „saw“ den Buchstaben „s“ generiert, da die Bedeutung der Suffixe nicht beachtet wird [SMR08].

Lemmatisierung ist ein weiterer Ansatz, der betrachtet wird. Es handelt sich um einen Ansatz, der dem Stemming ähnelt, jedoch mit dem Unterschied, dass die Wörter mit ihrer Bedeutung betrachtet werden [SMR08]. Die Wörter sollen hier nicht auf ihre Wortstämme reduziert werden, sondern auf ihre Lemmas. Das Lemma eines Wortes ist, im lexikographischen Sinn, die Form des Wortes, die als Stichwort in einem Nachschlagewerk zu finden wäre. Für diese Reduzierung werden Vokabulare und Transformationen verwendet, die die Flexionen umkehren sollen [SMR08]. So wird zum Beispiel aus dem Wort „saw“ je nach Kontext im Satz das Wort „see“ oder „saw“ [SMR08].

Stopword-Entfernung Die meisten Texte enthalten Füllwörter, die keine Bedeutung für das Verfahren tragen [SR03]. Hierzu gehören im Englischen Artikel wie „the“ und „a“ oder auch Pronomen wie „I“ und „myself“. Diese Wörter werden Stopwords genannt und entfernt. Für die englische Sprache gibt es domänenunspezifische Listen mit diesen sogenannten *Stopwords*, die im Rahmen anderer Projekte entstanden sind. Die Entfernung von Stopwords aus diesen Listen kann einen positiven Einfluss auf die Klassifizierung von Texten haben [SR03]. Auch domänenspezifische Listen können aus dem jeweiligen Korpus erstellt werden [AY11]. Auch die Anwendung von diesen Listen kann einen Klassifizierung begünstigen [AY11].

2.1.2 Feature Selection

Als *Feature Selection* oder auch *Feature Extraction* wird die Auswahl oder Konstruktion von Features bezeichnet, die dann zur Generierung der *Feature Vektoren* verwendet werden [Den+19]. Hierzu werden zunächst die Tokens oder ihre Kombinationen gewichtet [Den+19]. Daraufhin wird meist nur eine Teilmenge ausgewählt und weiterverarbeitet [Den+19]. Diese Auswahl findet auf Korpusebene und nicht auf Dokumentebene statt [Den+19]. Für die Auswahl der Features existieren mehrere Möglichkeiten. Ansätze, die nur einmal Features filtern und diese dann für den Prozess verwenden, nennt man Filter Ansätze [Den+19]. Dahingegen werden Ansätze, die alle möglichen Teilmengen mithilfe des Lernverfahrens evaluieren, Wrapperansätze genannt [Den+19]. Zudem existieren Hybridmodelle, diese verwenden zunächst einen Filteransatz und daraufhin einen Wrapperansatz [Den+19]. Nach der Gewichtung findet sich dann eine Menge von Features, die für jedes Dokument zu betrachten sind [Den+19].

2.1.3 Feature Vektoren

Als *Feature Vektor* wird im Folgenden die Darstellung eines Dokuments als Vektor von gewichteten Werten der zuvor gewählten Features bezeichnet. Nachdem über eine Methode zur *Feature Selection* Features ausgewählt wurden, wird für jedes Dokument ein Feature Vektor bestimmt [MP18]. Dieser enthält für jedes Feature die Information, ob es im Text vorhanden ist [MP18]. Dies kann eine binäre Markierung der Präsenz des Features im Text, die Anzahl der Vorkommnisse des Tokens oder eine komplexere Metrik sein.

2.1.4 Klassifizierung

Die Feature Vektoren alleine stellen keine Einschätzung dar, welches Label dem Dokument zuzuordnen ist. Hierfür wird ein Verfahren benötigt, dass die Texte anhand der Feature Vektoren einem Label zuordnet. Oft werden hierfür Verfahren des maschinellen Lernens verwendet [RN12a].

2.2 Maschinelles Lernen

Für die Klassifizierung der Feature Vektoren werden Verfahren des maschinellen Lernens verwendet. In dieser Arbeit werden dafür *Support Vector Maschines* verwendet, die im Folgenden erläutert werden. Neben dem Verfahren selbst müssen auch Metriken zur Evaluation und ein Verfahren zu Ermittlung besagter Metriken bekannt sein. Daher werden auch die verwendeten Metriken und zur Bestimmung selbiger Kreuzvalidierung erläutert.

2.2.1 Support Vector Machines

Support Vector Machines (SVMs) sind ein Verfahren zur binären Klassifizierung von Punkten in einem $\mathbb{R}^n$ [Ped+11]. Um diese Klassifizierung durchzuführen, legt das Verfahren eine Hyperebene in den Raum [Ped+11]. Eine SVM führt also eine lineare Klassifizierung des Raumes mit einer Menge von Punkten $\{x : w \cdot x + b = 0\}$ als Separator durch [RN12b]. Dieser Separator soll den größtmöglichen Abstand zu allen Eingabepunkten haben, also wird ein Maximum-Margin-Separator konstruiert [RN12b]. Die Eingabepunkte sind Punkte im Raum, denen ein Label $y = 1$ oder $y = -1$ zugeordnet ist. Daraufhin wird eine optimale Lösung für das Optimierungsproblem

$$\underset{\alpha}{\operatorname{argmax}} \sum_j a_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (\vec{x_j} \cdot \vec{x_k})$$

unter den Nebenbedingungen $\alpha_j \geq 0$ und $\sum_j a_j y_j = 0$ gesucht [RN12b]. Dann ist es möglich, w mit $w = \sum_j \alpha_j x_j$ zu berechnen, und dies definiert die Hyperebene [RN12b]. Die Klassifizierung ergibt sich als

$$h(\vec{x}) = \operatorname{sign}\left(\sum_j \alpha_j y_j (\vec{x} \cdot \vec{x_j}) - b \right)$$

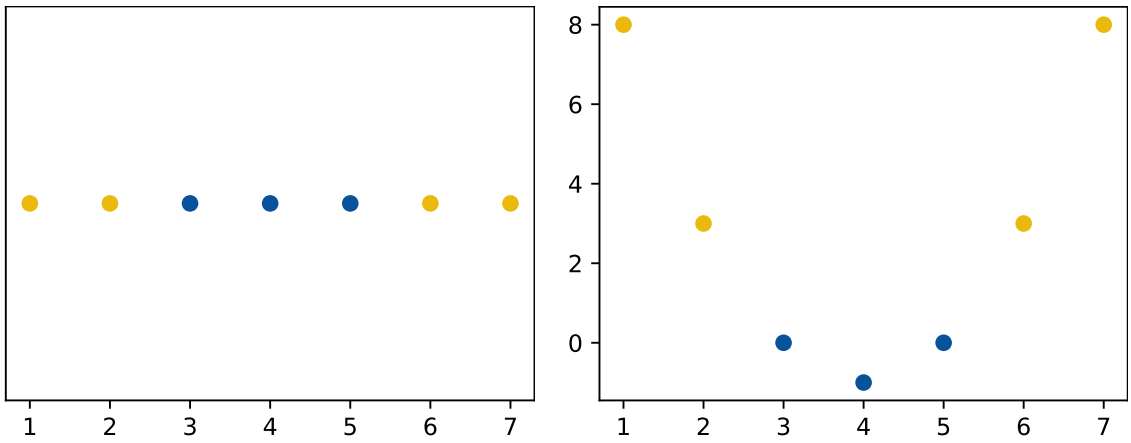
[RN12b]. Die Vektoren, die dem Separator am nächsten liegen, werden *Support-Vektoren* genannt [RN12b]. Nur ihre Gewichte α_j sind dann nicht 0 [RN12b].

Eine SVM stellt eine linearen Klassifizierung dar. Daher wird üblicherweise, wenn nicht lineare Probleme gelöst werden sollen, der sogenannte *Kernel-Trick* verwendet [RN12b]. Hierfür werden die Eingabepunkte in einen höherdimensionalen Raum eingebettet [RN12b]. Wird dieser Raum hochdimensional genug gewählt, ist das Problem dann linear separierbar [RN12b]. Als Beispiel hierfür ist ein Klassifizierungsproblem in $\mathbb{R}$, bei dem die positive Klasse alle Punkte im Intervall

[3, 5] sind. Alle anderen Punkte sind der negativen Klasse zuzuordnen. Abbildung 1a zeigt, dass eine lineare Trennung der Klassen nicht möglich ist. Wird nun der Eingaberaum über die Funktion

$$f : \mathbb{R} \to \mathbb{R}^2 \quad f(x) \mapsto \begin{pmatrix} x \\ (x-3)\cdot(x-5) \end{pmatrix}$$

in den $\mathbb{R}^2$ eingebettet, wie in Abbildung 1b, zeigt sich, dass eine lineare Trennung nun möglich ist.



(a) *Datenpunkte in* $\mathbb{R}^1$ (b) *Datenpunkte in* R^2

Abbildung 1.: *Beispiel zum Kernel-Trick*

2.2.2 Fehlerarten

Jede binäre Entscheidung eines Klassifizierers fällt in eine von 4 Kategorien. Es gibt die korrekten Fälle, also die Eingabepunkte, die richtig als positiv t_p oder richtig als negativ t_n klassifiziert werden.

Zudem kann ein Klassifizierungssystem zwei Arten von Fehler machen. Ein Fehler erster Art ist eine falsch positive Zuordnung f_p, also wird ein als negativ gelabeltes Dokument mit dem positiven Label versehen. Hingegen ist ein Fehler zweiter Art eine falsch negative Zuordnung f_n. Hier wird ein Eingabepunkte fälschlicherweise nicht erkannt. Eine Darstellung dieser Werte in Matrixform wird Konfusionsmatrix genannt [Faw06]. In Abbildung 2 ist die Darstellung zu sehen.

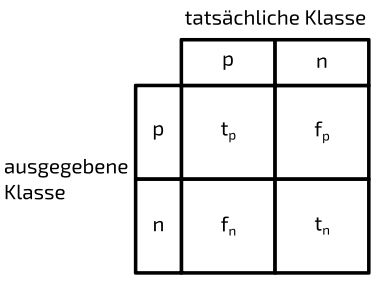


Abbildung 2.: *Konfusionsmatrix*

2.2.3 Performance Metriken

Zur Evaluation der Performance des Systems wird die Ausgabe des finalen Modells als Konfusionsmatrix betrachtet. Hierfür wird eine Teilmenge der in der Wissenschaft üblichen Metriken für die Evaluation von Verfahren des maschinellen Lernens verwendet. Diese Teilmenge wurde ausgewählt, da für die hier betrachtete Anwendung nur ein Teil der Metriken interessant sind. Die Genauigkeit des Systems spielt für die hier betrachtete Anwendung keine große Rolle, da ein falsch positives Ereignis weniger relevant ist als ein falsch negatives Ereignis. Die Metriken werden alle aus der Konfusionsmatrix berechnet und haben einen Wertebereich von $[0,1]$.

Recall

Der Recall wird als das Verhältnis von richtig positiv erkannten Samples zu den tatsächlich als positiv gelabelten Samples angegeben. Er gibt also relativ an, wie viele Fehler zweiter Art das System macht. Wenn das System möglichst viele tatsächlich richtige Sample erkennt ist der Recall höher. Diese Metrik wurde gewählt, da ein Fehler zweiter Art in der hier betrachtete Anwendung einen definitiven Verlust von Information bedeutet. Daher sollte der Recall möglichst hoch sein.

$$REC = \frac{t_p}{t_p + f_n} \quad [\text{RN12a}] \tag{2.1}$$

Precision

Die Precision eines statistischen Verfahrens ist das Verhältnis von richtig positiv erkannten Samples zu überhaupt positiv erkannten Samples. Dieser Wert wird also besonders groß, wenn keine Fehler erster Art auftreten. Dann erreicht der Wert sein Maximum von 1. Die Größe gibt also an, wie gut das System falsche Samples aussortiert.

$$PRE = \frac{t_p}{t_p + f_p} \quad [\text{RN12a}] \tag{2.2}$$

F_β-Mass

Das F_β-Maß ist eine Metrik, die Recall und Precision in einer Größe zusammenfasst. In vielen Echtweltszenarien widersprechen sich eine Optimierung für besonders guten Recall und ein besonders guten Precision Wert. Die Precision wird besonders gut, wenn das System nur anschlägt, wenn es sich sicher ist ein richtiges Sample gefunden zu haben, denn so bleibt die Anzahl an falsch positiven Samples klein. Der Recall hingegen ist dann besonders gut, wenn das System möglichst all richtigen Sample findet, ist also auf jeden Fall 1, wenn das System immer anschlägt. Daher muss hier in der Bewertung der beiden Größen stets ein Abwägungsprozess stattfinden. Hierfür wird oft das F_1-Measure verwendet. Es wird als das harmonische Mittel von Recall und Precision berechnet.

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC} \quad [\text{Chi92}] \tag{2.3}$$

In seiner allgemeinen Formulierung erlaubt das F_β-Measure eine Gewichtung von Recall und Precision über einen Parameter β, wodurch der Recall β mal so viel gewichtet wird, wie die Precision. Formal handelt es sich um das gewichtete harmonische Mittel, die Formel lautet:

$$F_\beta = (1 + \beta^2) \cdot \frac{PRE \cdot REC}{(\beta^2 \cdot PRE) + REC} \text{ [Chi92]} \tag{2.4}$$

2.2.4 Kreuzvalidierung

Um die Performance eines Systems möglichst realitätsnah zu messen, wird Kreuzvalidierung verwendet. Hierbei wird das System auf Daten evaluiert, die für das Training nicht verwendet wurden. So soll Overfitting, also ein Auswendiglernen der Trainingsdaten, verhindert werden [HCL+03]. Hierfür werden die Daten in Trainings- und Validierungsdaten aufgeteilt [RN12c]. Dies hat den Nachteil, dass nicht auf allen Daten gearbeitet wird. Daher wird k-fache Kreuzvalidierung verwendet. Dieses Verfahren unterteilt die gesamten Testdaten in k Blöcke [RN12c]. Daraufhin werden Performancemetriken k mal ausgewertet, wobei jeweils einer der Blöcke als Validierungsdatensatz und $k-1$ Blöcke als Traingsdatensatz fungieren [RN12c]. Die Metriken für das Verfahren werden aus dem arithmetischen Mittel der Ergebnisse der Kreuzvalidierungsläufe bestimmt [RN12c].

2.2.5 Gittersuche

Verfahren des maschinellen Lernens erlauben es meist, das Verhalten des Verfahrens anhand von sogenannten Hyperparametern zu steuern. Die Gittersuche ist eine Möglichkeit, die Hyperparameter für eine Lernverfahren festzusetzen. Hierbei werden verschiedene Kombinationen von Hyperparametern ausprobiert [HCL+03]. Durch eine Kreuzvalidierung wird der beste betrachtete Parametersatz ermittelt und dieser wird dann für das Training verwendet. So werden für jedes evaluierten Verfahren die Hyperparameter verwendet, die – auf der Trainingsmenge – am erfolgversprechendsten wirken [HCL+03].

3 Related Work

Die in der vorliegenden Arbeit evaluierten Techniken liegen im Forschungsgebiet des *Natural Language Processings*, genauer im Teilbereich der Themenklassifizierung. Hingegen ist die betrachtete Anwendung thematisch der IT-Sicherheit zuzuordnen. Im Folgenden werden zunächst verwandten Arbeiten nach den erforschten Teilgebieten der Themenklassifizierung – *Preprocessing*, *Feature Selection* und *Dokumentklassifizierung* – erläutert. Daraufhin wird auf die in den Arbeiten betrachteten Anwendungen eingegangen. Hierbei unterscheiden sich die Verfahren nicht nur in der Zielsetzung, sondern auch in den Eingabedaten. Zum Beispiel unterscheidet sich die Länge der betrachteten Texte, die Sprache der Texte oder der Kontext der Texte erheblich.

3.1 Preprocessing

Das *Preprocessing* ist der erste Schritt der *Dokumentklassifizierung* und besteht meist aus regelbasierten, deterministischen Mutationen der Eingabe. Die Verfahren werden aus Bausteinen von einzelnen Veränderungen zusammengesetzt. Die Auswahl der richtigen Schritte ist stark anwendungsabhängig [UG14]. In der aktuellen Forschung gibt es wenig Veränderung in den gewählten Bausteinen zum *Preprocessing*, jedoch gibt es noch vergleichende Studien, die je einen bestimmten Anwendungsfall betrachten.

Eine wissenschaftliche Arbeit vergleicht die Performance von Kombinationen der Teilverfahren Tokenisierung, Stopword-Entfernung, Konvertierung zu Kleinbuchstaben und Stemming im Kontext von englischen und türkischen E-Mails und Zeitungsartikeln. Hierbei wird für die E-Mails eine binäre Klassifizierung und für die Zeitungsartikel eine Klassifizierung mit mehreren Klassen durchgeführt. Die Evaluation auf englischen Zeitungsartikeln kommt der hier betrachteten Anwendung am nächsten. Tokenisierung, Entfernung von Stopwords und Konvertierung zu Kleinbuchstaben scheinen eine positive Auswirkung auf die Klassifizierung zu haben. Die Durchführung von Stemming zeigt sich als nicht effektiv. Die Autoren haben in diesem Fall jedoch keine binäre Klassifizierung vorgenommen. Eine binäre Klassifizierung fand nur auf dem E-Maildatensatz statt. Hierbei stellte sich heraus, dass in diesem Fall nur die Konvertierung zu Kleinbuchstaben die Erkennung verbessert hat. Die in der vorliegenden Arbeit betrachtete Anwendung unterscheidet sich von diesen Evaluationen, da hier binäre Klassifizierung auf Zeitungsartikeln durchgeführt wird. [UG14]

Das Ziel, Kurznachrichten der Plattform Twitter in die drei Kategorien neutral, positiv und negativ einzusortieren, hat eine weitere vergleichende Studie. Bei der Verwendung von SVMs stellte sich heraus, dass für diese Anwendung alle betrachteten *Preprocessing*-Techniken förderlich waren. Es wird evaluiert, ob die Entfernung von URLs, Nutzernamen, Hashtags, Satzzeichen und Stopwords, die Ersetzung von Negations-Abkürzungen durch ihre ganze Form, die Normalisierung von Zeichen oder das Stemming von Wörter die Klassifizierung positiv beeinflusst. Der Anwendungsfall unterscheidet sich vom hier betrachteten dahingehend, dass die Texte wesentliche kürzer und informeller sind. Die Struktur der verwendeten Sätze ist einfacher und es kommt zu mehr Wortneuschaffungen und Ausdruck von Gefühlen durch eine Mutation von Wörtern. Zudem werden mehr URLs und Hashtags verwendet, was den Eingaberaum stark verändert. [KH18]

Durch eine weitere wissenschaftliche Arbeit wurde untersucht, wie türkische Zeitungsartikel nach ihrem Autor, dem Genre und dem Geschlecht des Autors klassifiziert werden können. Für diesen Anwendungsfall werden die Techniken Stemming, Satzzeichenentfernung und N-Gram-Modellierung betrachtet. Der Fall der Genre-Klassifizierung lässt sich am ehesten mit dem hier betrachteten Anwendungsfall der Themenklassifizierung vergleichen. Hierbei werden ohne die Verwendung von Stemming und Satzzeichenentfernung und unter Zuhilfenahme von 2-Grams *F1-Werte* von 0.937 erreicht. Der Unterschied zur hier betrachteten Anwendung liegt darin, dass mehr als zwei Genres betrachtet wurden und dass die Eingabedaten in türkischer Sprache vorlagen. [DK17]

Der konkrete Algorithmus *LemmaGen* wird in einer anderen Arbeit beschrieben. Dieser Algorithmus ist ein Lernalgorithmus, der aus einer Menge von Trainingsdaten sogenannte *Ripple down Rules (RDR)* erstellt. Diese RDRs stellen Gesetzmäßigkeiten für die Lemmatisierung von Wörtern dar. Hierbei werden zunächst allgemeine Regeln für die Ersetzung von Wörtern geschaffen, die, sobald ein Widerspruch gefunden wird, durch Ausnahmen erweitert werden. Die Regeln werden durch das Lernverfahren in einem Vorbereitungsschritt generiert. Für die Ausführung zur Laufzeit werden die gelernten Regeln auf den vorliegenden Text angewendet. [Jur+10]

3.2 Feature Selection

Es existieren verschiedenste Ansätze zur *Feature Selection* auf den Daten. Der einfachste Weg, Features auszuwählen, ist, alle generierten Features zu verwenden. Diese Technik zeigt bei den meisten Verfahren deutlich längere Laufzeiten im Training und ist daher praktisch schlecht einzusetzen. Für die Auswahl der Features existieren verschiedene Ansätze: Filter-, Wrapper- und Hybrid-Ansätze. Filteransätze wählen einzelne Features aus und verwenden diese weiter. Dies geschieht meist durch eine Bewertungsfunktion und eine darauf folgende Auswahl auf Basis der Bewertung. Im Gegensatz dazu wählen Wrappermethoden eine Teilmenge der Features mithilfe des Klassifizierers aus. Es wird eine Teilmengen gewählt, und dann wird die Performance des Klassifizierers evaluiert. So werden alle möglichen Teilmengen betrachtet, bis die Performance des Klassifizierers, im Kontext der Anwendung, gut genug ist. Dieser Ansatz ist aufgrund des Einsatzes des Klassifizierers zeitaufwändiger. Hybridmethoden stellen eine Mischung aus Filter-

und Wrappermethoden dar. Es wird zunächst die Grundmenge der Features verkleinert. Techniken wird eine Wrappermethode angewandt. Nachdem die Features ausgewählt wurden, wird für jedes Dokument ein Feature Vektor generiert. [Den+19]

Unter anderem wird eine Abwandlung der χ^2-Bewertung diskutiert. Es handelt sich um eine Filtermethode, die von der klassischen χ^2-Methode abweicht, da die Features nicht nur nach diesem Wert ausgewählt werden. Stattdessen werden die Features werden auch so ausgewählt, dass sie etwa gleichverteilt über den Klassen sind. So soll eine bessere durchschnittliche Performance erreicht werden. Bahassine et. al. evaluieren ihren Ansatz auf arabischen Texten verschiedener Länge und auf diesen Daten wird im Vergleich zu einer klassischen χ^2 Auswahl eine Verbesserung erzielt. [Bah+18]

Das *Relative descrimination criterion* (RDC) stellt eine weitere Filtermethode dar. Es werden eine positive und eine negative Klasse verwendet, wobei für das gerade betrachtete Dokument die positive Klasse immer genau die Dokumente mit dem gleichen Label sind. Die Methode versucht, folgenden Ziele zu erreichen: Terme, die nur in einer Klasse vorkommen, sollten viel gewichtete werden. Kommt ein Term nur in einer Klasse vor, aber selten, so soll er ein geringes Gewicht bekommen. Auch Terme, die in beiden Klassen vorkommen, erhalten geringe Gewichte. Diese Ziele sollen durch das RDC erreicht werden, dass für einen Term t mit der Anzahl an Häufigkeiten tc wie folgt definiert ist:

$$\mathrm{RDC} = \frac{|\mathrm{df}_{\mathrm{pos}} - \mathrm{df}_{\mathrm{neg}}|}{\min(\mathrm{df}_{\mathrm{pos}}, \mathrm{df}_{\mathrm{neg}}) \cdot \mathrm{tc}}$$

Hierbei steht $\mathrm{df}_{\mathrm{Klasse}}$ für die Dokumentfrequenz bezogen auf die Dokumente der Klasse. So entsteht ein Maß, dass vom betrachteten Term und seiner Häufigkeit im Text abhängt. Daraufhin wird für die Gewichtung des Terms selbst die Fläche unter dem Graphen RDC gegen tc betrachtet. Der Ansatz wird in einer Klassifizierung mit mehreren Klassen auf verschiedenen Datensätzen evaluiert. [Reh+15]

Eine auf RDC aufbauende Filtermethode ist das *Multivariate Relative Discrimination Criterion* (MRDC). Es wird zunächst für jeden der Terme das RDC berechnet. Daraufhin wird der Term mit dem höchsten Wert in die Menge der Features S übernommen. Nun sollen die Terme ausgewählt werden, die den höchsten Wert und die kleinste Korrelation zu den bereits ausgewählten Features haben. Für jeden Term wird ein neuer Wert wie folgt bestimmt.

$$\mathrm{MRDC}(t) = \mathrm{RDC}(t) \; - \sum_{t \neq t', t' \in S} (\mathrm{correlation}(\mathrm{t}, \mathrm{t}'))$$

Als Maß für die Korrelation wird der *Pearson Correlation Coefficient* verwendet. Für die Evaluation wurden *Multinomial Naïve Bayes*, *Decision Trees* und *Multilayer Perzeptron* getestet und die Performance wird als in allen Fällen höher als für reines RDC angegeben. [Lab+18]
Die hier betrachtete Anwendung unterscheidet sich insofern, als dass die Performance auf SVMs evaluiert wird.

Eine weitere Filtermethode modelliert die Feature-Teilmengen als *Fuzzy-Rough-Mengen*. Die Features werden *greedy* aus der Menge der möglichen Features ausgewählt. Als Metrik für die

Auswahl wird der Grad der Abhängigkeit der Menge gewählt. Als Featuremenge ist die Menge gewünscht, die möglichst nah am Grad der Abhängigkeit der gesamten Menge ist. Die Anwendung fokussiert sich auf Textnachrichten und daher sind die Eingabedaten verschieden zu den hier betrachteten. [Zuo+18]

Im Gegensatz zu den zuvor diskutierten Methoden ist *LW-Index with Sequence Forward Search* eine Hybridmethode. Der Ansatz verwendet den *LW-Index* um Features auszuwählen. Dieser wird bestimmt, indem für eine Auswahl an Features die Trainingsdaten in den Featureraum eingefügt werden. Daraufhin werden aufgrund der Label Cluster gebildet. Der *LW-Index* gibt dann die durchschnittliche maximale Bewegungsfreiheit der Cluster an, ohne dass sich diese nach der Bewegung überschneiden. Der Algorithmus verwendet *Sequence Forward Search*, um die Teilmenge von Features auszuwählen. Es handelt sich um ein iteratives Auswahlverfahren, bei dem in jeder Iteration das Feature, dass den *LW-Index* am meisten erhöht, hinzugefügt wird. Hierfür wird in jeder Iteration jedes noch nicht verwendete Feature temporär hinzugefügt und der *LW-Index* des Klassifizierungsergebnis berechnet. Es wird so lange iteriert, bis es keine Features mehr gibt, oder die obere Schranke in der Anzahl getroffen wurde. Mit diesem Ansatz werden Ergebnisse nahe einer Wrappermethode erzeugt, jedoch ist die Laufzeit wesentlich geringer. Die geringere Laufzeit wird dadurch erreicht, dass nicht für jede betrachtete Teilmenge das Klassifizierungsverfahren verwendet wird, sondern nur der *LW-Index* berechnet wird. [Liu+17a]

3.3 Dokumentklassifizierung

Das Feld der *Dokumentklassifizierung* ist weitreichend, es werden viele verschiedene Verfahren des maschinellen Lernens für die Klassifizierung verwendet. Dies ist der Fall, da nach dem Schritt der *Feature Selection* die Daten als *Feature Vektoren* vorliegen und daher sehr allgemein in verschiedenste Verfahren gegeben werden können. Diese Verfahren sind nicht mehr unbedingt spezifisch für Probleme des *Natural Language Preprocessing* entworfen.

In vielen Fällen finden *Support Vector Maschines* Anwendung [Liu+17a; DK17; Reh+15; Bah+18]. Sie sind in der Lage, effektiv in großen Eingaberäumen Klassifizierungen durchzuführen [Ped+11].

Ein weiteres klassisches Klassifizierungskonzept ist *Naïve Bayes*. Hierbei handelt es sich um eine statistische Auswertung, die davon ausgeht, dass die Features voneinander unabhängig sind. Zu Beginn des Prozesses hat jede Klasse $c \in N_c$ eine A-priori-Wahrscheinlichkeit Pr. Daraufhin werden alle Features in $\vec{w} = (w_1, w_2, \ldots, w_n)$ betrachtet. Jedes von ihnen verändert die Wahrscheinlichkeitsverteilung. Die Kategorie c, der das Dokument angehört, ist dann:

$$c = \arg\max(p(c|\vec{w})) = \arg\max(Pr(c)p(\vec{w}|c))$$

Es lassen sich verschiedene Modelle für die Berechnung von $p(\vec{w}|c)$ vergleichen. Hierbei zeigt sich, dass sich für *Dokumentklassifizierung* die Betrachtung am besten eignet, die die *tf-idf*-Werte der einzelnen Wörter pro Kategorie als normalverteilt annimmt. [Xu18]

Diese Art der Klassifizierung ist auf mehrere Klassen ausgelegt, die hier betrachtete Anwendung hingegen hat nur zwei Klassen.

Azam et. al. vergleichen die Performance von *Naïve Bayes* zur Performance von *K-Nearest-Neighbour (K-NN)* als Klassifizierer. K-NN wählt zu einem zu klassifizierenden Punkt die K nächsten Punkte im Raum, im Bezug auf den euklidischen Abstand, aus. Die Klassifizierung des Punktes ist dann der Majorant der Klassen der K ausgewählten Punkte. Die Evaluation fand auf einem Datensatz von akademischen Veröffentlichungen statt, welche den jeweiligen Kategorien zugeordnet werden sollten. [Aza+18]
Dies unterscheidet sich von der hier betrachteten Anwendung dahingehend, als die akademischen Veröffentlichungen einen unterschiedlichen Schreibstil und eine andere Länge haben als Zeitungsartikel.

In letzter Zeit sind auch andere Lernverfahren, zum Beispiel Deep Learning mithilfe von *Graph–Convolutional Neural Networks (CNN)* populär geworden. Hierbei werden die Texte zunächst in eine Graphendarstellung konvertiert, wobei sich die Kanten aus der räumlichen Nähe von zwei Wörtern im Text ergeben. Daraufhin werden diesen Graphen mithilfe eines CNN weiterverarbeitet. Die Ausgabe wird durch drei vollständig verbundene Layer erzeugt. Diese vollziehen die eigentliche Klassifizierung. Es werden bei einer nicht binären, sondern hierarchischen Klassifizierung F_1-Werte von bis zu 0.8255 erreicht. [Pen+18]

Ein anderes Beispiel für einen Einsatz von Deep Learning Methoden wird in einer weiteren wissenschaftlichen Arbeit betrachtet. Das Ziel dieser Methode ist eine Multilabel-Klassifizierung. Das heißt, dass jedem Text die Teilmenge von Labeln zugeordnet werden soll, die am relevantesten ist. Die Menge an möglichen Labeln ist in dieser Anwendung sehr groß, anders als in der hier betrachteten Anwendung. Diese ist das andere Extrem, da nur genau ein Label betrachtet wird. Die Texte werden als Sammlung von Tokens repräsentiert. Diese Tokens sind einzelne Wörter oder die Kombinationen von bis zu drei aufeinander folgenden Wörtern, sogenannten N–Grams. [Liu+17b]

3.4 Anwendung

Die Anwendungen von *Dokumentklassifizierung* sind divers. Sie reichen von der Klassifizierung von Texten in Echtzeit, zum Beispiel in der Spamdetektion [Li+19], über die Generierung von personalisierten Vorschlägen [Zhe+18], hin zur Sentimentanalyse [KH18], um die Meinungen der Autoren zu erfassen.

Spamdetektion Der Bereich der Spamdetektion ist aufgrund seiner großen Relevanz für das alltägliche Leben stark erforscht. Auch in der Spamdetektion werden die Eingabedaten binär klassifiziert.

In einer wissenschaftlichen Arbeit wird ein Graph-CNN eingesetzt, um Spam in Bewertungen auf einem großen chinesischen online Marktplatz zu erkennen [Li+19]. Diese Anwendung unterscheidet sich von der hier betrachteten darin, dass die Texte nicht nach ihrem Inhalt klassifiziert werden,

sondern nach ihrer Authentizität. Eine weitere Anwendung findet sich in der Spamdetektion in E-Mails. Es gibt Ansätze mit SVMs, die eine Klassifizierung in Echtzeit vornehmen [SP+18].

Medizin Auch in der Medizin finden sich verschiedenste Anwendungen für *Dokumentklassifizierung*. So existieren Ansätze, Tweets dahingehend zu klassifizieren, ob sie über ein bestimmtes Thema, wie die Grippe, berichten. So lässt sich eine kostengünstige Methode zur Überwachung der öffentlichen Gesundheit konstruieren. [DBM17]
Eine andere Anwendung ist die Detektion von Infektionen, die in Krankenhäusern auftreten. Ehrentraut et. al. klassifizieren Patientenakten dahingehend, ob eine solche Infektion aufgetreten ist [Ehr+18].

Vorschläge Eine weitere Anwendung stellen personalisierte Vorschläge für Nutzer dar. Es existieren Ansätze, Deep Reinforcement Learning zu verwenden um Nutzern personalisierte Vorschläge zu machen. [Zhe+18]

Sentimentanalyse Als Sentimentanalyse wird eine Textanalyse bezeichnet, die nicht als Ziel hat, nach dem Inhalt des Texts zu klassifizieren, sondern nach den Meinungen und Aussagen des Autors. Daher unterscheidet sich die Sentimentanalyse fundamental in ihrem Ziel von den hier betrachteten Ansätzen. Eine solche Kategorisierung hat zudem meist mehr Labels als das hier betrachtete binäre System.

Ein Beispiel für Sentimentanalyse ist die Klassifizierung von Kurznachrichten der Plattform Twitter anhand der Stimmung des Autors. Die sogenannten Tweets sollen in die Kategorien positiv, neutral und negativ einsortiert werden. [KH18]

4 Evaluierte Lösungsansätze

Um ein Verfahren zu finden, welches das gegebene Problem möglichst gut löst, werden im Folgenden systematisch verschiedene Ansätze zur Klassifizierung der Texte evaluiert. Hierfür werden für *Preprocessing* und *Feature Selection* jeweils eine Menge von Verarbeitungsschritten ausgewählt, die evaluiert werden. Jedes der Verfahren verwendet je eine Teilmenge der Verarbeitungsschritte für das *Preprocessing* und eine für die *Feature Selection*. Nicht alle Teilmengen sind inhaltlich sinnvoll, daher werden nicht alle evaluiert. Im Folgenden werden die ausgewählten Verarbeitungsschritte und die gewählten Teilmengen beschrieben.

4.1 Preprocessing

Das *Preprocessing* erhält als Eingabe den ursprünglichen Text und gibt als Ausgabe eine Liste von Tokens aus, die aus den gewählten Verfahren resultieren. Zunächst wird *Tokenisierung* und die *Entfernung von nicht-ASCII Zeichen* angewendet. Diese Transformationen finden in jedem Fall statt. Daraufhin wird evaluiert, welche der folgenden zusätzlichen Verfahren die Performance verbessern. Die Verfahren wurden in der in Abbildung 3 beschriebenen Reihenfolge eingesetzt und alle möglichen Pfade durch das Diagramm wurden evaluiert.

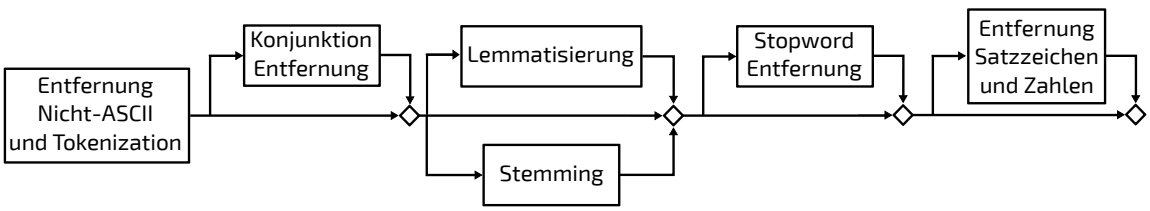


Abbildung 3.: *Schematische Darstellung der Verfahren zum Preprocessing*

4.1.1 Stemming

Als Algorithmus für Stemming wird der *Porter Stemming*-Algorithmus evaluiert. Er bestimmt die Wortstämme für einzelne Wörter auf Basis vordefinierter Regeln [Por80]. Die gewählten Regeln sind spezifisch für die englische Sprache und daher ist dieser Ansatz nur für eine solche Anwendungen geeignet [Por80]. Der Algorithmus betrachtet jedes Wort als Abfolge von Vokalen und Konsonanten [Por80]. Dafür werden folgende Definitionen aus der wissenschaftlichen Arbeit übernommen. Für Folgen von Vokalen wird das Zeichen „*V*“ und analog für Konsonanten „*C*“ verwendet [Por80].

Zudem wird eine m-fache Wiederholung eine Musterfolge „X“ als „X^m“ definiert [Por80]. Optionale Wortteile haben die Form „$[X]$“ [Por80]. Dann hat jedes Wort die Form „$[C](VC)^m[V]$“ [Por80]. Die Metrik des Wortes oder Wortteils wird als m definiert [Por80]. Der Algorithmus definiert dann Regeln in der Form „(Bedingung) S1 → S2“ [Por80]. Die Bedingung kann Anforderungen an m oder das Wort im Allgemeinen stellen [Por80]. Hierbei stellt S1 die Endung vor Anwendung der Regel dar und S2 die Endung nach der Anwendung [Por80]. Ein Beispiel für eine solche Regel ist (m>0) ICATE →IC. Diese Regel wird also nur angewendet, wenn die Metrik mindestens 1 wird, so wird zum Beispiel aus „triplicate“ das Wort „triplic“. Die Regeln sind in fünf Schritte aufgeteilt, die sequentiell abgearbeitet werden [Por80]. Dieser Algorithmus wurde gewählt, da er einer der weitverbreitesten für Stemming ist.

4.1.2 Lemmatisierung

Es ist nicht geklärt, ob Lemmatisierung einen positiven Effekt auf die Klassifizierung hat. Wir verwenden zur Evaluation den *WordNet Lemmatizer* des NLTK Frameworks. Dieser verwendet den „morphy“ Algorithmus des WordNet Frameworks. Dieser Algorithmus nimmt eine Lemmatisierung mithilfe von festen Regeln und Ausnahmenlisten vor [Uni10]. Auch der Ansatz „LemmaGen“ aus [Jur+10] wurde in Erwägung gezogen. Er wurde jedoch verworfen, da die gegebene Implementierung auf modernen GNU/Linux System mit einem modernen C++ Standard nicht mehr lauffähig ist.

4.1.3 Entfernung bestimmter Wörter

Es werden zwei Möglichkeiten, Wörter zu entfernen, evaluiert. Die Erste ist Stopword-Entfernung; hierbei wird eine feste Liste von Wörtern entfernt. Zur Stopword-Entfernung wird die Stopword-Liste des NLTK-Projekts verwendet. Diese beinhaltet 174 Wörter oder Konstruktionen, darunter hauptsächlich Personalpronomen und Konjugationen von Hilfsverben. Die zweite betrachtete Möglichkeit ist die, eine bestimmte Wortart zu entfernen. Hier werden im spezifischen alle Wörter entfernt, die das NLTK-Framework als Konjunktion identifiziert.

4.1.4 Satzzeichen-Entfernung und Zahlen-Ersetzung

Eine weitere Möglichkeit des *Preprocessings* ist das Entfernen von Zeichenketten, die keine Wörter darstellen. Diese Entfernung kann einen Vorteil bieten, wenn die Annahme getroffen wird, dass solche Zeichenketten für die Klassifizierung nicht von Bedeutung ist. Hier wurde dies durch das Ersetzen von Zahlen durch das Literal „NUMBER“ und das Entfernen von Satzzeichen realisiert. Die Annhame die hier getroffen wird ist, dass, wenn die Reihenfolge nicht betrachtet wird, Satzzeichen ihre Bedeutung verlieren und dass der Wert einer Zahl keine Bedeutung hat, sondern nur die Tatsache, dass es eine Zahl gibt.

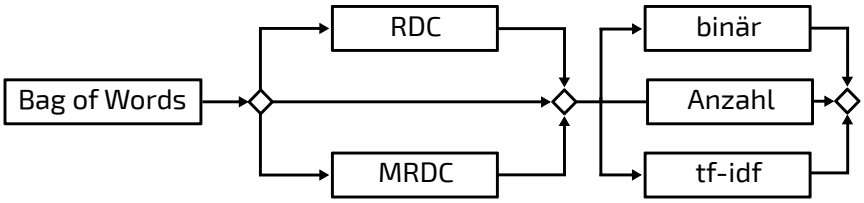


Abbildung 4.: *Schematische Darstellung der Verfahren zur Feature Selection*

4.2 Feature Selection

Um Feature Vektoren aus den vom *Preprocessing* generierten Tokensammlungen zu generieren, bieten sich mehrere Verfahren an. Als erster Schritt der *Feature Selection* wird jedoch in jedem betrachteten Fall die sogenannte *Bag of Words*-Betrachtung angewendet. Hier werden zunächst alle Dokumente auf einer Darstellung gebracht, die pro in diesem Dokument vorkommenden Token die Anzahl an Vorkommnisse darstellt. Hierbei geht die Reihenfolge und Satzzugehörigkeit der Tokens verloren. Daraufhin findet auf dieser Repräsentation die eigentliche Auswahl statt. Nachdem eine Auswahl getroffen wurde, wird dokumentweise jedem Feature ein Wert zugeordnet. Im Folgenden werden vier Verfahren zur *Feature Selection* betrachtet und daraufhin die drei betrachteten Gewichtungsverfahren erläutert. Eine Übersicht über die Verfahren findet sich in Abbildung 4.

4.2.1 Alle Features

Die wohl einfachste Art, Features zu wählen, ist nicht zu wählen. Sondern alle Features zu übernehmen. So geht keine Information verloren. Dieser Ansatz wurde betrachtet; Aufgrund der enormen Anzahl an möglichen Features ist die Berechnung jedoch sehr komplex. Daher wurden der Ansatz verworfen.

4.2.2 Hauptkomponentenanalyse

Eine weitere Möglichkeit, die um die Dimension des Featureraums zu reduzieren, ist die *Hauptkomponentenanalyse*. Hierbei wird der Eingaberaum als Vektor von Zufallsvariablen x dargestellt. Aus den Eingabepunkten lässt sich eine Annäherung für diesen Vektor bestimmen. Die Dimension des Eingaberaums soll reduziert werden, indem passende Linearkombinationen der Einträge des Vektors gesucht werden [Jol86]. Diese wenigen Linearkombinationen sollen möglichst viel der Varianz des ursprünglichen Vektors abbilden [Jol86]. Die Anzahl der gewählten Abbildungen ist ein Hyperparameter. Im i-ten Schritt wird der Vektor von Konstanten a_i ausgewählt, für den die Funktion $a_k x$ zu den bereits gewählten Funktionen unkorreliert ist und die größte Varianz hat [Jol86]. Das a_i ergibt sich als der Eigenvektor zur Kovarianzmatrix aller Features, der dem größten noch nicht betrachteten Eigenwert entspricht [Jol86]. Dieser Ansatz benötigt zur Berechnung die Kovarianzmatrix aller Features, oder eine aus den Samples generierte Annäherung [Jol86]. Diese ist für einen Featureraum, der so groß ist wie der betrachtete, nicht trivial im Hauptspeicher vorzuhalten. Daher wurde dieser Ansatz nicht weiter verfolgt.

4.2.3 Relative Descrimination Criterion

Das *Relative Descrimination Criterion (RDC)* stellt eine Methode zur *Feature Extraction* dar, die versucht, die Features auszuwählen, die die Klassen möglichst gut unterscheiden, wie in Abschnitt 3.2 beschrieben. Für diese Selektionsmethode werden jeweils Fälle evaluiert, in denen 100 oder 10 000 Features ausgewählt wurden.

4.2.4 Multivariate Relative Discriminative Criterion

Multivariate Relative Discriminative Criterion (MRDC) ist einer Erweiterung des RDC und soll auch die Korrelation zwischen den Features betrachten [Lab+18]. (Siehe Kapitel 3.2) Anders als in der ursprünglichen Abhandlung wird jedoch eine alternative Berechnungsmethode für den *Pearson Correlation Coefficient* verwendet.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad \text{[Lab+18]}$$

$$\Leftrightarrow r_{xy} = \frac{n\sum_{i=1}^{n}(x_i y_i) - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n\sum_{i=1}^{n}(x_i^2) - (\sum_{i=1}^{n}(x_i)^2}\sqrt{n\sum_{i=1}^{n}(y_i^2) - (\sum_{i=1}^{n}(y_i)^2}$$

$$\Leftrightarrow r_{xy} = \frac{\sum_{i=1}^{n}(x_i y_i) - n\bar{x}\bar{y}}{\sqrt{\sum_{i=1}^{n}(x_i^2) - n\bar{x}^2}\sqrt{\sum_{i=1}^{n}(y_i^2) - n\bar{y}^2}}$$

Diese hat den Vorteil, dass für alle Features nicht nur die Terme im Nenner vorberechnet werden können, sondern auch für die linke Hälfte des Zählers die Dokumente, die nicht beide das Feature beinhalten, vernachlässigt werden können. So lässt sich die Berechnungszeit für eine Korrelationsberechnung im betrachteten Anwendungsfall stark beschleunigen, da die Feature Vektoren dünnbesetzt sind. Auch dieses Verfahren wird für eine Auswahl von 100 und 10 000 Features betrachtet.

4.2.5 Gewichtungsverfahren

Um die ausgewählten Features zu gewichten, bieten sich verschiedene Methodiken an. Zunächst lässt sich die Präsenz eines Features binär darstellen. Eine weitere Möglichkeit ist die Anzahl der Vorkommnisse des Features im Dokument als Gewicht zu nehmen. Die letzte betrachtete Gewichtungsmethode ist *Term Frequency – Inverse Document Frequency (tf-idf)*. Hierbei wird die Häufigkeit eines Features im Text im Verhältnis zum Gesamtvorkommen des Features betrachtet. Es ergibt sich also *tf* als die Frequenz des Features im aktuellen Text und *idf* als $idf = \log \frac{N}{n}$, wobei N die Anzahl der Dokumente im Korpus ist und n die Anzahl an Dokumenten, in denen das Feature vorkommt [LL99]. Die Metrik ergibt sich dann als $tf\text{-}idf = tf \cdot idf$ [LL99]. Diese drei Gewichtungsmethoden wurden evaluiert.

4.3 Klassifizierung

Die in der hier betrachteten Anwendung benötigte Klassifizierung stellt insofern einen Spezialfall der klassischen Klassifizierung dar, als dass nur zwei unterschiedliche Klassen betrachtet werden: Die Klasse der Texte, die über Identitätsdiebstähle berichtet und die, die dies nicht tun. Diese Arbeit beschränkt sich auf die Betrachtung eines Verfahrens zur Klassifizierung. Das gewählte Verfahren sind *Support Vector Machines (SVMs)*. Das Verfahren wurde gewählt, da es eines der klassischen Klassifizierungsverfahren ist. Zudem wird es in anderer Literatur häufig verwendet und so wird die hier angestellte Betrachtung vergleichbarer mit anderen Arbeiten. Um die Hyperparameter für die SVM festzulegen, wird Gittersuche als Teil des Verfahrens verwendet.

Da das hier betrachtete Problem einen unbalancierten Trainingsdatensatz beinhaltet, werden die Samples verschieden gewichtet, mit der Gewichtungsfunktion

$$w_c = \frac{\text{Anzahl Samples}}{\text{Anzahl Klassen} \cdot \text{Anzahl Samples in Klasse c}} \tag{4.1}$$

Also wird für einen Eingabepunkt der Klasse c ein Fehler in der Bewertungsfunktion der SVM mit w_c multipliziert. So soll sichergestellt werden, dass beide Klassen den gleichen Einfluss auf die Lösung des Optimierungsproblems haben.

Zudem werden die Feature Vektoren vor der Verarbeitung durch die SVM auf die Länge 1 normalisiert. Dies geschieht, indem der Vektor x mit den Skalar $\frac{1}{|x|}$ multipliziert wird. Hierbei ist $|x|$ der Euklidische Abstand zum Ursprung.

Es werden zwei verschiedene Optimierungsprobleme verwendet, um die Methoden zum *Preprocessing* zu evaluieren. Im Folgenden seien $\boldsymbol{x}_i \in \mathbb{R}^n, i = 1,\dots,l$ die Trainingsvektoren und $\boldsymbol{y} \in \mathbb{R}^l$ die Labels mit

$$\forall i \in \mathbb{N}, 0 < i < l+1 : y_i \in \{-1, 1\}$$

Das Verfahren generiert dann einen Gewichtsvektor $\boldsymbol{\omega}$ und die Klassifizierungsfunktion ist $\text{sgn}(\boldsymbol{\omega}^T \boldsymbol{x})$. Für die Verfahren, die 100 Features auswählen, wird eine klassische SVM-Implementierung mit einem *Radial-Basis-Function (RBF)*-Kernel verwendet. Der Kernel ist definiert als

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) = exp(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2), \gamma > 0 \quad [\text{CL11}].$$

Hierbei ist γ ein Hyperparameter. Genauer wird folgendes Optimierungsproblem mithilfe von *libSVM* [CL11] gelöst:

$$\min_{\boldsymbol{\omega}, b, \xi} \frac{1}{2} \boldsymbol{\omega}^T \boldsymbol{\omega} + C \sum_{i=1}^{l} \xi_i \quad [\text{CL11}]$$

$$\text{unter der Bedingung } y_i(\boldsymbol{\omega}^T \phi(\boldsymbol{x}_i) + b) \geq 1 - \xi_i \quad [\text{CL11}]$$

$$\xi_i \geq 0 \quad [\text{CL11}]$$

Hierbei ist C ein Hyperparameter, der steuert, wie sehr Fehler bestraft werden. Wird C groß, werden Klassifizierungsfehler schwerer gewichtet. Die Verfahren, die mehr Features auswählen, verwenden

einen linearen Kernel, da hier der Eingaberaum so groß ist, dass eine lineare Klassifizierung schon in diesem Raum möglich scheint. Hier wird dieses Optimierungsproblem mithilfe von *libLINEAR* [Fan+08] gelöst:

$$\min_{\omega} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}(\max(0, 1 - y_i\omega^T\boldsymbol{x}_i))^2 \tag{4.2}$$

Zudem wird so die Laufzeit deutlich verringert, da es nur einen Hyperparameter C gibt, im Gegensatz zum RBF-Kernel, der zwei benötigt. Dies verkleinert den Raum, der mit Gittersuche durchsucht werden muss, deutlich.

5 Evaluationsverfahren

Um die oben beschriebenen Verfahren zu evaluieren, wird das in Abbildung 5 schematisch dargestellte Evaluationsverfahren angewendet. Zunächst lädt das Evaluationsverfahren die benötigten Validierungs- und Trainingsdaten. Daraufhin werden erst *Preprocessing* und *Feature Extraction* durchgeführt. Auf den so generierten Features werden mithilfe einer inneren Kreuzvalidierung die Hyperparameter für das Training der SVM bestimmt. Mit diesen Parametern wird dann die SVM trainiert. Nachdem die Modellgenerierung auf den Trainingsdaten abgeschlossen ist, beginnt die Validierung. Hierfür werden die gewählten Features und das trainierte Modell der SVM in den Validierungsprozess übertragen. Daraufhin werden *Preprocessing* und *Feature Extraction* durchgeführt. Die so entstandenen Feature Vektoren werden dann mithilfe des zuvor trainierten Modells klassifiziert und die Ergebnisse werden zur Generierung der betrachteten Metriken verwendet.

Das Evaluationsverfahren führt zudem 10-fache Kreuzvalidierung durch. Wie in Abbildung 5 zu sehen, umschließt die Kreuzvalidierung das komplette Verfahren, insbesondere auch die innere Kreuzvalidierung, die für die Generierung der Hyperparameter während des Training verwendet wird. Zudem werden die Trainings- und Validierungsdaten streng getrennt. Es kommt nur zu einem Datenfluss vom Trainingsverfahren in das Validierungsverfahren, indem die gewählten Features und trainierten Modelle übertragen werden. Ein Datenfluss in die andere Richtung ist ausgeschlossen.

Im Folgenden wird weiter auf den verwendeten Datensatz die gewählten Parameter und die verwendeten Metriken eingegangen.

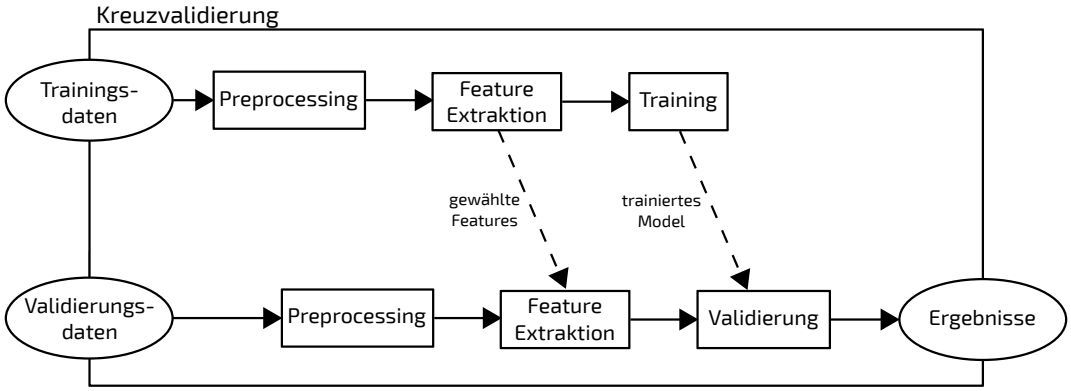


Abbildung 5.: *Schematische Darstellung des Evaluationsverfahren*

5.1 Verwendete Datensätze

Zum Test der Verfahren wird ein Datensatz von englischsprachigen Online-Nachrichtenartikeln verwendet. Diese sind im Zeitraum von 2007 bis 2019 publiziert worden. Der Datensatz enthält 15 211 Artikel. Diese haben im eine Mindestlänge von 200 Zeichen (inklusive Leer- und Satzzeichen), eine Durchschnittslänge von 2 966 Zeichen und eine Medianlänge von 2 659 Zeichen. Von den Artikeln sind 1 997 als „berichtet über einen Identitätsdiebstahl“ markiert. Dementsprechend sind 13 214 nicht so markiert. Der Datensatz ist also nicht balanciert, was durch die Gewichtung der SVM ausgeglichen wird.

5.2 Gewählte Parameter

Die Verfahren haben nur an zwei Stellen Parameter, die es zu wählen gilt. Die erste Stelle ist die Anzahl der Features, die in der *Feature Selection* ausgewählt werden. Hierzu werden die Verfahren zur *Feature Selection* jeweils mit einer Anzahl an Features von 100 und 10 000 evaluiert. Die zweite Stelle, an der Parameter zu wählen sind, sind die Parameter für die SVM. Eine Erläuterung zu den Parametern findet sich in Abschnitt 4.3. Hier werden für die Gittersuche im Fall von 100 Features γ mit $\{0.01, 0.1, 1, 10, 100\}$ und C mit $\{2^{-10}, 2^{-8}, 2^{-6}, 2^{-4}, 2^{-2}, 1, 2^2, 2^4, 2^6, 2^8\}$ evaluiert. Werden 10 000 Features ausgewählt muss aufgrund des linearen Kernels der SVM nur C gefunden werden. Hier werden Werte zwischen 2^{-15} bis 2^{10} durch *libLINEAR* ausgewählt.

5.3 Verwendete Metriken

Mit Precision, Recall und F_1-Metrik werden die in der Wissenschaft üblichen Metriken verwendet. Zudem werden wir F_β mit $\beta = 19$ betrachten, da für die hier betrachtete Anwendung der Recall einen höheren Stellenwert hat, als die Precision. Dies ist der Fall, da es einem Experten trivial möglich ist, einen irrelevanten Artikel auszusortieren. Einen nicht markierten Artikel zu finden, sieht das System jedoch nicht vor.

Jede der zehn Instanzen einer Kreuzvalidierung erzeugt eine Konfusionsmatrix. Daher werden auch zehn verschiedene Werte für jede der Metriken berechnet. Der Gesamtwert, der für das Verfahren verwendet wird, wird als das arithmetische Mittel bestimmt.

6 Ergebnisse

Im Folgenden werden die Ergebnisse aus verschiedenen Perspektiven beleuchtet. Zunächst wird die Performance der verschiedenen betrachteten Verarbeitungsschritte einzeln evaluiert. Daraufhin werden die besten Gesamtverfahren für die einzelnen Metriken betrachtet. Die vollständigen Ergebnisse für alle Verfahren finden sich im Anhang A. Die Grafiken auf Seitenbreite finden sich auch im Anhang B.

6.1 Betrachtung der Verfahrensschritte

Für die Betrachtung der einzelnen Verfahrensschritte werden drei Abbildungsformen herangezogen. Zunächst eine Darstellung der Precision gegen den Recall, wie in Abbildung 7 zu sehen. Diese Darstellung ermöglicht einen einfache Evaluation der Schritte, wenn die Verfahren nach bestimmten Kriterien farblich markiert werden. Hierbei gilt sowohl für den Recall, als auch die Precision, dass ein größerer Wert besser ist. Jedoch ist zu beachten, dass eine weitere Entfernung vom Ursprung nicht unbedingt einen besseren Klassifizierer beschreibt. Die zweite Darstellung ist eine, welche die Veränderung der Performance durch Einsatz eines Verfahrensschrittes betont. Dies wird dadurch erreicht, dass jedes Verfahren mit dem Verfahren, das bis auf den aktuell betrachteten Schritt äquivalent ist, durch einen Pfeil verbunden wird. Der Pfeil zeigt stets vom Verfahren ohne den Schritt auf das Verfahren mit dem Schritt. Ein Beispiel findet sich in Abbildung 9. Durch diese Darstellung soll für Schritte, für die die Trennung der Gruppen nicht direkt graphisch klar ist, eine Tendenz hervorgehoben werden. Die dritte Darstellung ist ein Histogramm der erreichten F_β Werte. Hierfür werden die Werte durch Abrunden auf das nächste Vielfache von 0.2 gruppiert. Diese Darstellung ermöglicht einen einfachen Vergleich der Performance auf Basis der F_β Metriken, auch wenn die Ergebnisse gestreut liegen. Zudem wird in dieser Darstellung das arithmetische Mittel der Metriken angegeben.

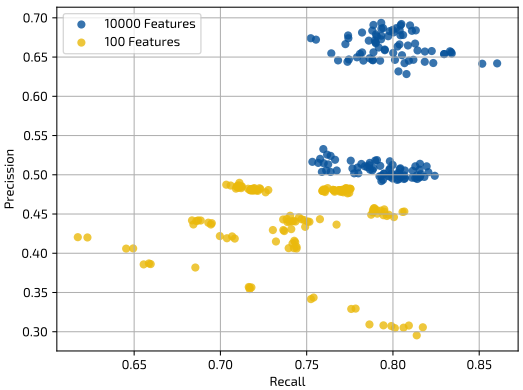


Abbildung 6.: *Recall und Precision nach Anzahl betrachteter Features*

Anzahl ausgewählter Features In Abbildung 6 ist der Vergleich der Verfahren mit verschiedenen Anzahlen ausgewählter Features zu sehen. Es stellt sich heraus, dass die Verfahren, bei denen nur 100 Features ausgewählt wurden alle deutlich schlechtere Performance zeigen, als die Verfahren mit mehr Features. Daher wird der Rest der Analyse sich auf die Verfahren mit mehr Features beschränken.

Feature Extraction Ein Vergleich der beiden getesteten Verfahren zur *Feature Extraction* findet sich in Abbildung 7. Auch hier zeigt sich eine klare Trennung der beiden Punktwolken. Der Recall der Methoden befindet sich, bis auf einige Ausreißer, in der gleichen Größenordnung. Es ist eine leichte Tendenz zu einem besseren Recall für das RDC-Verfahren zu erkennen. Zudem ist die Precision für das RDC-Verfahren in allen Fällen deutlich besser. Daher liefert das RDC-Verfahren hier deutlich bessere Ergebnisse.

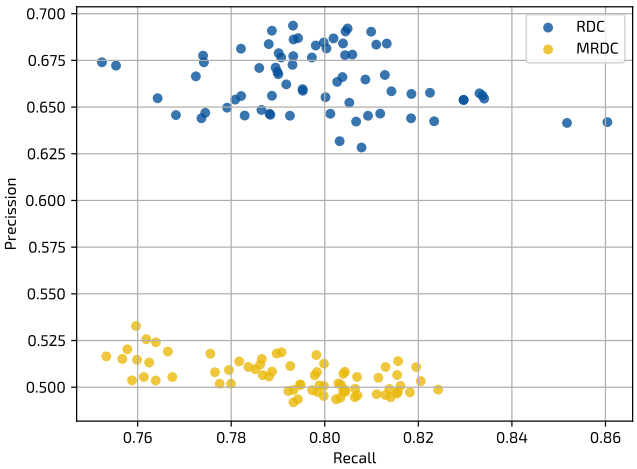
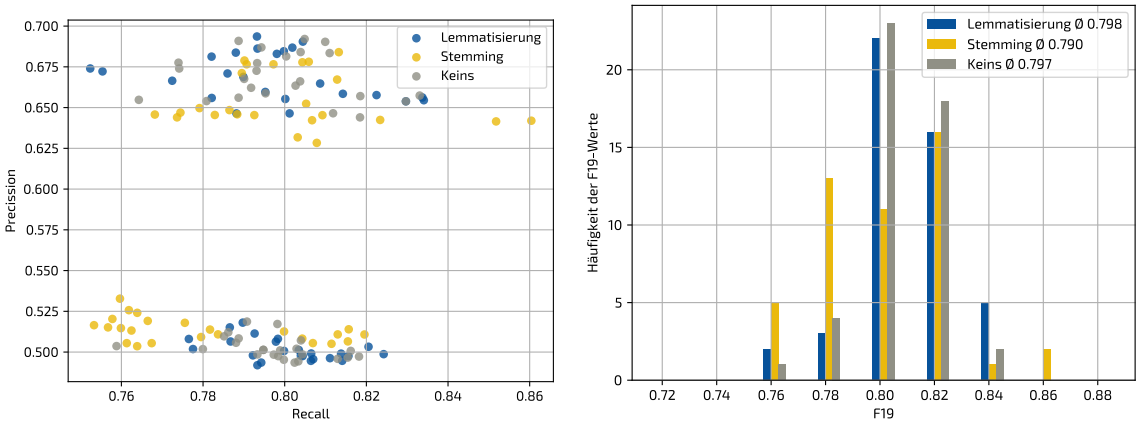


Abbildung 7.: *Recall und Precision nach Feature Extractions-Verfahren*

Stemming und Lemmatisierung Die Betrachtung von Abbildung 8a zeigt, dass sich über die Sinnhaftigkeit des Einsatzes von Lemmatisierung und Stemming keine klare Aussage treffen lässt. So führt Stemming zu vielen der niedrigsten Recall-Werte, aber auch zu den höchsten. Wird lediglich das arithmetische Mittel von F_1 und F_{19} als $\overline{F_\beta}$-Wert betrachtet ergibt sich das Bild, dass Stemming mit $\overline{F_1} = 0.669$ und $\overline{F_{19}} = 0.79$ schlechter ist, als keine der Techniken anzuwenden mit $\overline{F_1} = 0.672$ und $\overline{F_{19}} = 0.797$. Dies ist jedoch der großen Varianz der Werte geschuldet. In Abbildung 8b wird dies nochmals verdeutlicht. Die F_{19} Werte sind für Stemming weiter gestreut, als für die anderen beiden Ansätze. Das erlaubt besonders schlechte und besonders gute Ergebnisse. Daher hängt die Wahl hier stark von den anderen gewählten Verarbeitungsschritten ab.



(a) *Precision gegen Recall*

(b) *Verteilung F_{19}*

Abbildung 8.: *Ergebnisse zu Stemming und Lemmatisierung*

Konjunktionen Um zu beurteilen, ob das Entfernen von Konjunktionen dem Verfahren einen Vorteil beschafft, lässt sich Abbildung 9 betrachten. In der Grafik gut zu sehen ist, dass es keinen klaren Trend gibt, ob die Entfernung von Konjunktionen den Klassifizierer verbessert. Zudem lässt

sich beobachten, dass die beiden Klassifizierer, welche die besten Recall Werte erzielen, sich darin unterscheiden, ob Konjunktionen entfernt wurden. Von diesen beiden schneidet der Klassifizierer, der keine Konjunktionen entfernt, deutlich besser ab. Im Allgemeinen lässt sich jedoch keine Aussage über die Nützlichkeit des Entfernens von Konjunktionen treffen.

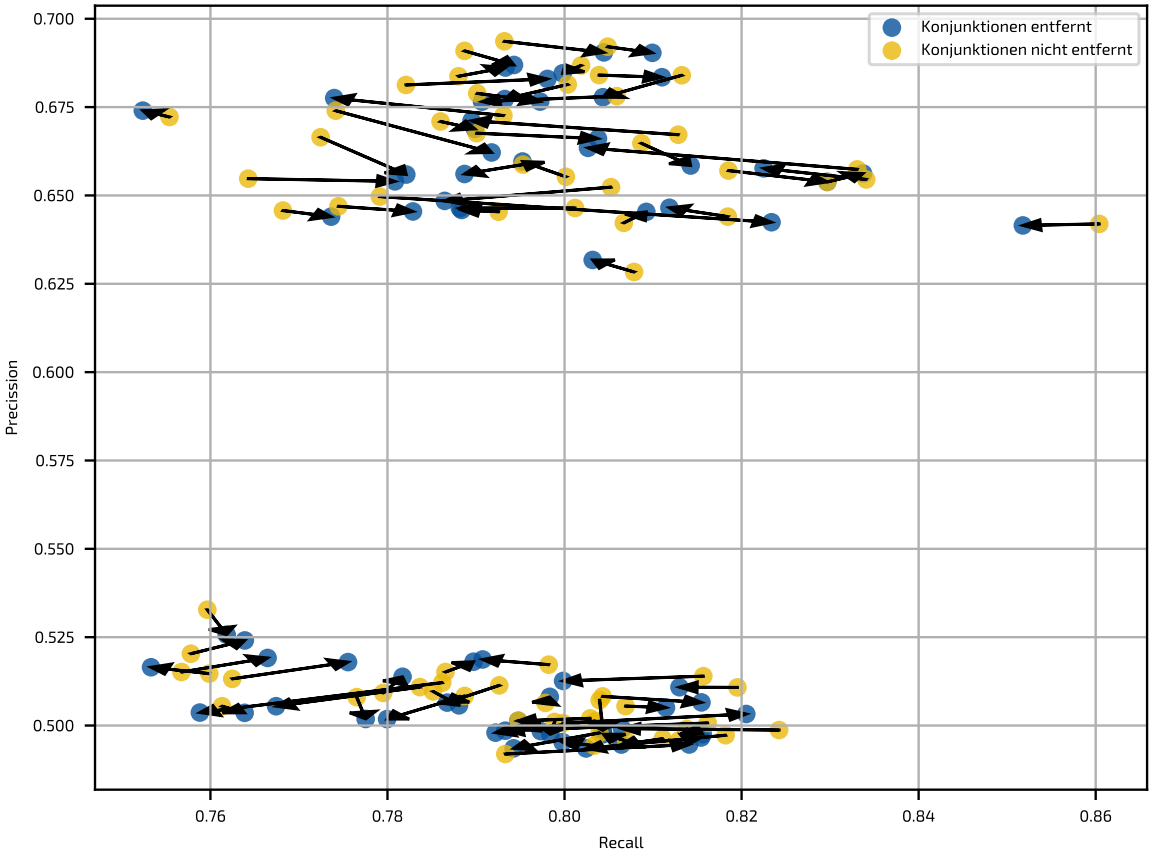


Abbildung 9.: *Veränderung Recall und Precision bei Verwendung von Entfernung von Konjunktionen*

Feature Gewichtungsverfahren Für die Gewichtung der Features wurden drei Varianten betrachtet. Eine binäre Darstellung, die Anzahl der Vorkommnisse und *tf-idf*. Ein Vergleich der Performance findet sich in Abbildung 10. Hier zeigt sich, dass die Verfahren nicht trivial zu ordnen sind. Dennoch ist zu sehen, dass in Kombination mit anderen Verfahren die Gewichtung nach Anzahl einen besseren Recall erreicht, als die anderen Techniken. Auch zeigt sich, dass eine Gewichtung mit *tf-idf* eine bessere Precision ermöglicht.

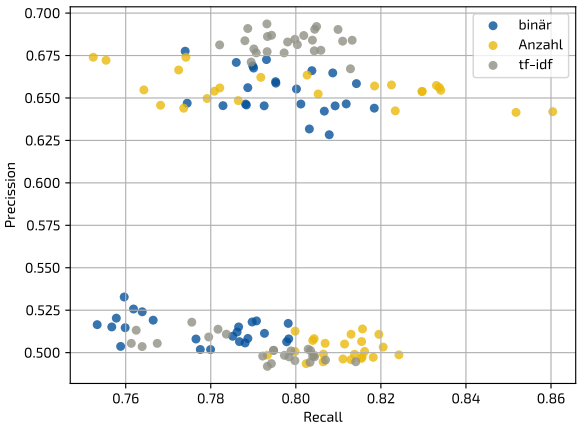


Abbildung 10.: *Recall und Precision nach Feature Gewichtungsverfahren*

Stopword Entfernung Für die Entfernung von Stopwords lässt sich kein allgemeiner Trend verzeichnen. Jedoch lässt sich sagen, dass, wie in Abbildung 11 zu sehen ist, der durchschnittliche F_{19}-Wert kleiner ist, wenn Stopwords entfernt werden. Gleiches gilt für den F_1-Wert, der ohne die Entfernung im Durchschnitt 0.673 erreicht. Wird die Entfernung von Stopwords als Verarbeitungsschritt eingesetzt fällt der Wert auf 0.668. Interessant ist auch, dass für die beiden Verfahren, die den höchsten Recall erreichen, keine Stopwords entfernt wurden. Daher ist eine Entfernung von Stopwords aus der NLTK Stopword-Liste nicht förderlich für das Verfahren.

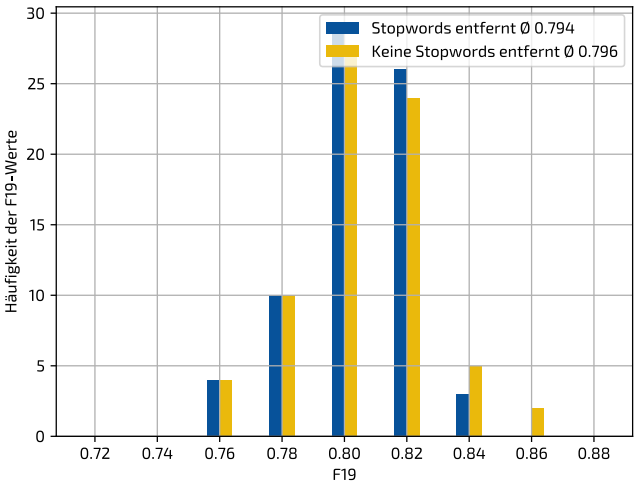
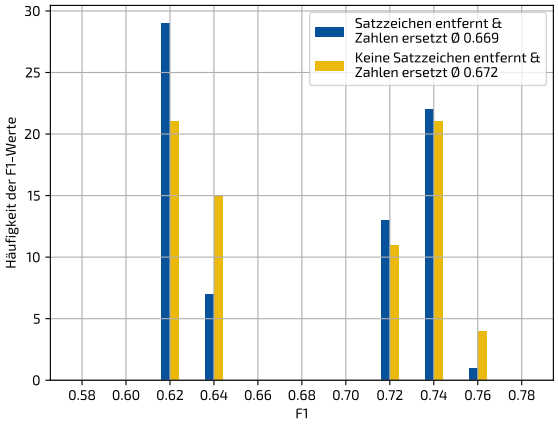
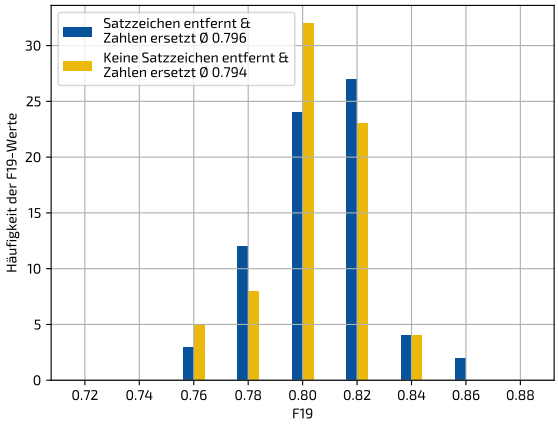


Abbildung 11.: *Recall und Precision getrennt nach Stopword Entfernung*

Satzzeichen-Entfernung und Zahlen-Ersetzung Die Entfernung von Satzzeichen und das Ersetzen von Zahlen durch die Zeichenkette „NUMBER“ hat einen interessanten Effekt auf die Ergebnisse des Klassifizierers. Dieser Verarbeitungsschritt verbessert die Performance, wenn für die Betrachtung die F_{19}-Metrik herangezogen wird. Dies zeigt sich in Abbildung 12b dadurch, dass die Verteilung leicht in Richtung eines höheren Wertes verlagert ist. Zieht man jedoch die F_1-Metrik heran, wie in Abbildung 12a so ist das Ergebnis ohne diesen Verarbeitungsschritt besser. Daher scheint der Schritt einen besseren Recall, gegenüber einer besseren Precision zu erzeugen. Dies zeigt sich auch in Abbildung 13. Hier ist in den Pfeilen im Mittel ein Trend in Richtung einer geringeren Precision und eines höheren Recalls zu verzeichnen.



(a) *Verteilung F_1*



(b) *Verteilung F_{19}*

Abbildung 12.: *Satzzeichen-Entfernung und Zahlen-Ersetzung*

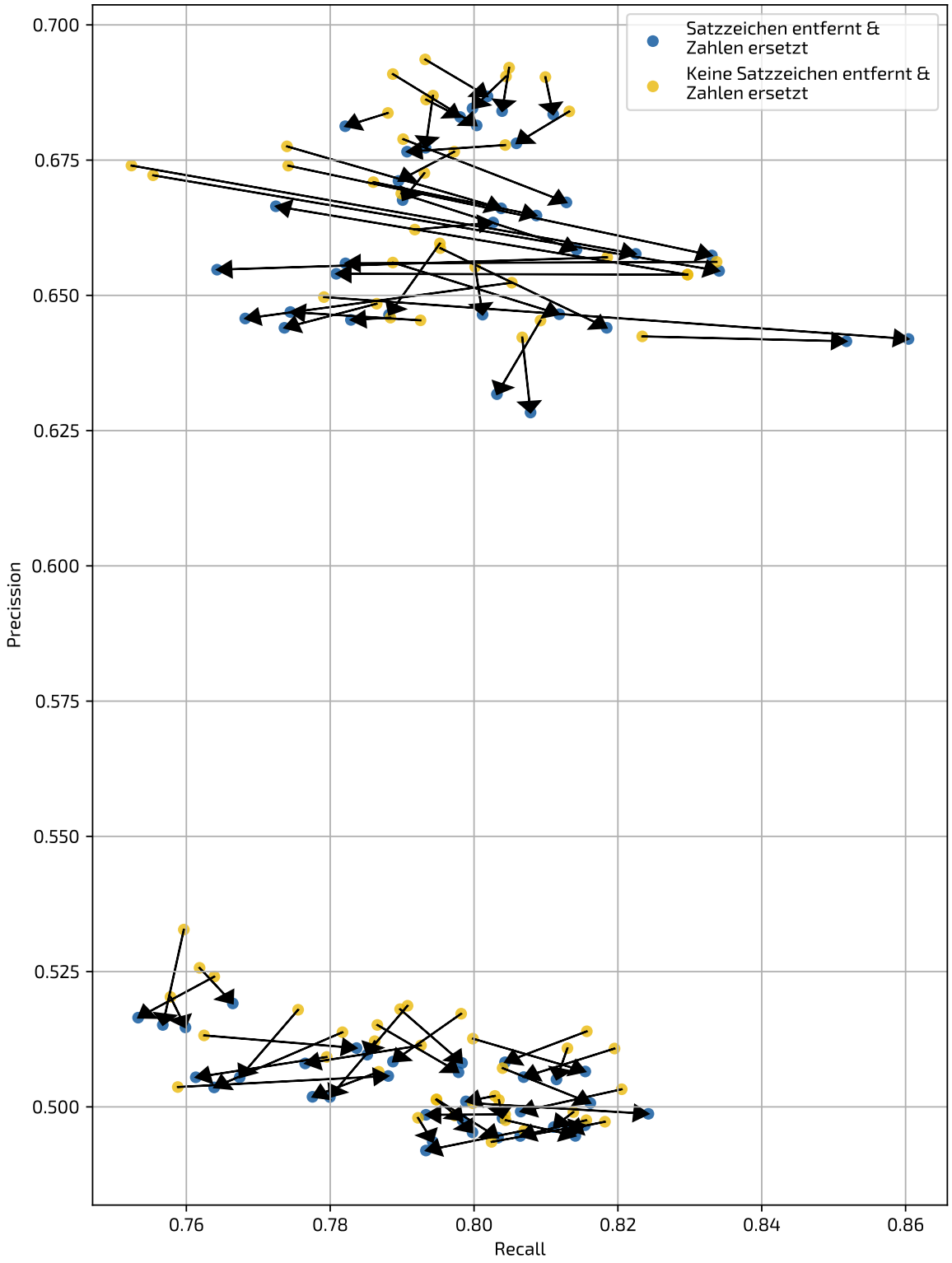


Abbildung 13.: *Veränderung Recall und Precision bei Verwendung von Satzzeichen-Entfernung und Zahlen-Ersetzung*

6.2 Betrachtung nach Metrik

Da die Verfahrensschritte in der Anwendung nicht einzeln, sondern als Ganzes eingesetzt werden, werden im Folgenden die besten Verfahren für die betrachteten Metriken beleuchtet. Diese Einschätzung erlaubt die Auswahl eines Verfahrens unter Berücksichtigung etwaiger Wechselwirkungen zwischen den Schritten. Es werden alle betrachteten Verfahren in Erwägung gezogen, interessant ist jedoch, dass alle Verfahren, die für eine Metrik das beste Ergebnisse erzeugen, 10 000 Features ausgewählt haben. Eine Übersicht über die besten Verfahren nach Metrik findet sich in Tabelle 1.

Tabelle 1.: *Übersicht der besten Klassifizierer nach Metrik*

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Recall	Precision	F_1	F_{19}
Nein	Stemming	Nein	Ja	RDC	Anzahl	**0.860**	0.642	0.735	**0.860**
Nein	Lemma-tisierung	Nein	Nein	RDC	tf-idf	0.793	**0.694**	0.739	0.793
Ja	Keine	Nein	Nein	RDC	tf-idf	0.810	0.690	**0.745**	0.810

Recall Das betrachtete Verfahren, das den höchsten Wert für den Recall erzeugt, führt die folgenden Schritte durch:

1. Entfernen von Nicht-ASCII Zeichen
2. Tokenisierung
3. Stemming mit dem Porter-Stemmer
4. Entfernen von Satzzeichen und Ersetzen von Zahlen durch die Zeichenkette „NUMBER“
5. Auswahl von 10 000 Features mit dem *Relative Discrimination Criterion*
6. Gewichtung der Features über die Häufigkeit eines Features im Text
7. Normalisierung der Feature Vektoren auf Länge 1
8. Training einer *Support Vector Machine* mit einem linearen Kernel

Es wird ein Recall von 0.860 erreicht, die Precision erreicht 0.642. Interessant zu sehen ist, dass die Precision unter der Auswahl über den Recall nicht zu stark leidet. Es handelt sich um den 69ten Klassifizierer, wenn nach Precision sortiert wird. Damit befindet sich der Klassifizierer auch hier in den oberen 10%. Es scheint sich also um einen guten Klassifizierer zu handeln, obwohl eine Auswahl nach dem Recall alleine im Allgemeinen nicht sinnvoll ist, da sich trivial ein Klassifizierer mit einem Recall von 1 erzeugen lässt.

Precision Das Verfahren, das hingegen die höchste Precision erreicht, führt die folgenden Schritte durch:

1. Entfernen von Nicht-ASCII Zeichen
2. Tokenisierung
3. Lemmatisierung mit dem NLTK-Framework
4. Auswahl von 10 000 Features mit den *Relative Discrimination Criterion*
5. Gewichtung der Features über *tf-idf*
6. Normalisierung der Feature Vektoren auf Länge 1
7. Training einer *Support Vector Machine* mit einem linearen Kernel

Es werden eine Precision von 0.694 und ein Recall von 0.793 erreicht. Dieses Verfahren gleicht dem Verfahren mit dem besten Recall nur darin, dass keine Stopwords entfernt wurden und RDC zur *Feature Selection* verwendet wurde. Daher scheint keines der anderen Verfahren einen grundsätzlich negativen Einfluss auf die Klassifizierung zu haben.

F_1 Um einen möglichst hohen F_1-Wert zu erreichen werden folgende Schritte durchgeführt:

1. Entfernen von Nicht-ASCII Zeichen
2. Tokenisierung
3. Entfernen von Konjunktionen
4. Auswahl von 10 000 Features mit den *Relative Discrimination Criterion*
5. Gewichtung der Features über *tf-idf*
6. Normalisierung der Feature Vektoren auf Länge 1
7. Training einer *Support Vector Machine* mit einem linearen Kernel

Das Verfahren erreicht einen Recall von 0.810, einen Precision von 0.690 und damit einen F_1-Wert von 0.745. Es unterscheidet sich zum zweitbesten Verfahren darin, dass im zweitbesten keine Konjunktionen entfernt wurden. Dieses hat einen um 0.005 kleineren Recall und einen um 0.002 kleineren F_1-Wert, dafür aber eine um 0.002 höhere Precision. Hier scheint das Entfernen von Konjunktionen also einen minimalen Vorteil für die Precision auf Kosten des Recalls zu bringen. Jedoch sind die Unterschiede sehr klein und auf anderen Eingabedaten kann sich die Reihenfolge verändern.

F_{19} Die F_{19}-Metrik wurde speziell für die hier betrachtete Anwendung angesetzt. Es zeigt sich, dass es sich um das Verfahren handelt, das dem mit den höchsten Recall gleicht. Das ist wenig überraschend, da die F_{19}-Metrik den Recall stark bevorzugt. Jedoch ist auch die Precision annehmbar, wie oben bereits festgestellt.

7 Zusammenfassung

Die Frage, welches Verfahren sich am besten eignet, um Nachrichtenartikel bezüglich ihres Inhalts zu klassifizieren, lässt sich nicht trivial beantworten. Viele der betrachteten Verfahren liefern akzeptable Ergebnisse und unterscheiden sich nur in der Ausprägung der Fehlerklassen. Allgemein lässt sich lediglich sagen, dass in jedem Fall die Verwendung von 10 000 Features der von 100 vorgezogen werden sollte, da die Ergebnisse für 100 gewählte Features deutlich schlechter sind. Für die anderen betrachteten Verfahrensschritte lassen sich solche allgemeinen Aussagen nicht treffen.

Betrachtet man einen allgemeinen Anwendungsfall, in dem Fehler erster und zweiter Art gleich schwer ins Gewicht fallen, wird der beste Klassifizierer durch den besten F_1 Wert beschrieben. In dieser Betrachtung ist dies das Verfahren, dass die folgenden Schritte durchführt:

1. Entfernen von Nicht-ASCII Zeichen
2. Tokenisierung
3. Entfernen von Konjunktionen
4. Auswahl von 10 000 Features mit den *Relative Discrimination Criterion*
5. Gewichtung der Features über *tf-idf*
6. Normalisierung der Feature Vektoren auf Länge 1
7. Training einer *Support Vector Machine* mit einem linearen Kernel

Dieses Verfahren erreicht auf den Testdaten in 10-facher Kreuzvaliderung eine Recall von 0.81, eine Precision von 0.69, F_1 von 0.74 und F_{19} von 0.81. Damit erzeugt es annehmbare Ergebnisse für den allgemeinen Fall einer binären Themenkategorisierung.

Wird jedoch die hier beschriebene Anwendung beachtet, sind genauere Aussagen zur Beantwortung der Forschungsfrage möglich. Die Anwendung hat die Eigenschaft, dass Fehler erster Art in einem späteren Verarbeitungsschritt durch einen Experten korrigiert werden können. Eine Korrektur von Fehlern zweiter Art ist jedoch nicht vorgesehen, daher wiegen diese in der Bewertung schwerer. Hieraus folgt, dass der Recall wichtiger ist, als die Precision. Jedoch sollte die Precision nicht beliebig klein werden, da dann das System den Experten nur noch wenig Klassifizierungsarbeit abnimmt.

Unter diesen Gesichtspunkten wird für ein einzusetzendes Verfahren eines mit den folgenden Schritten vorgeschlagen:

1. Entfernen von Nicht-ASCII Zeichen
2. Tokenisierung
3. Stemming mit den Porter-Stemmer
4. Entfernen von Satzzeichen und Ersetzen von Zahlen durch die Zeichenkette „NUMBER“
5. Auswahl von 10 000 Features mit den *Relative Discrimination Criterion*
6. Gewichtung der Features über die Häufigkeit eines Features im Text
7. Normalisierung der Feature Vektoren auf Länge 1
8. Training einer *Support Vector Machine* mit einem linearen Kernel

Diese Verfahren erreicht auf den Testdaten in 10-facher Kreuzvaliderung eine Recall von 0.86, eine Precision von 0.64, F_1 von 0.74 und F_{19} von 0.86. Dies sind sowohl für den Recall als auch für F_{19} die höchsten erreichten Werte.

Im Vergleich zu anderen Arbeiten sind die hier erreichten Ergebnisse plausibel. In einer wissenschaftlichen Abhandlung wurden in einem Mehrklassen-Klassifizierungsproblem auf Zeitungsartikeln mithilfe des RDC F_1-Werte von 0.7438 erreicht [Reh+15]. Die hier erzielten Ergebnisse sind leicht besser, mit einem maximalen F_1-Wert von 0.7448. Jedoch ist ein Vergleich in der dritten Nachkommastelle nicht sehr aussagekräfig, da die Testdaten und das Klassifizierungsproblem unterschiedliche sind.

In einer anderen Arbeit wurde berichtet, dass das MRDC in einem mehrklassen Klassifizierungsproblem auf Zeitungsartikeln mithilfe von *Decision Trees* F_1-Werte von bis zu 0.69 erreicht [Lab+18]. Zudem wurden in der Abhandlung für RDC nur F_1-Werte von 0.50 erreicht [Lab+18]. Eine solche Verbesserung der Werte gegenüber RDC ließ sich für die hier betrachtete Anwendung und unter Verwendung von SVMs als Klassifizierer nicht beobachten.

Des Weiteren wurde berichtet, dass Stopword-Entfernung einen positiven Einfluss auf eine Klassifizierung englischer Zeitungsartikel hat [UG14]. Dies konnte hier nicht nachgewiesen werden, da in der hier betrachteten Anwendung die Entfernung von Stopwords in den meisten Verfahren einen negativen Einfluss auf die Klassifizierung hatte. Dies könnte in der ausgewählten Stopword-Liste oder den unterschieden in der Anwendung begründet sein.

Im Rahmen dieser Arbeit konnten aus Zeit- und Ressourcengründen nicht alle sinnvoll scheinenden Ansätze verfolgt werden. Im Folgenden werden einige Ansätze erläutert, deren Einsatz die Ergebnisse verbessern könnten.

Im Rahmen des *Preprocessings* lassen sich noch weitere Verfahren testen. Unter anderem wäre es interessant zu evaluieren, ob ein Schritt, der alle Wörter entfernt, die keine Eigennamen, Adjektive, Verben oder Nomen sind, das Ergebnis verbessert. Dieser Schritt könnte das Verfahren verbessern, da zu vermuten ist, dass inhaltliche Bedeutung größtenteils durch diese Wörter erzeugt wird. Es werden alle Wörter entfernt, die diese Wörter verknüpfen und so sinnvolle Sätze erzeugen. Da

in dem hier betrachteten Ansatz die Wörter jedoch ohne ihre Reihenfolge oder Nähe zueinander betrachtet werden, geht ihre Bedeutung unter Umständen verloren.

Ein weiterer Schritt im *Preprocessing* könnte *Part of Speech Tagging* sein. Hierbei wird den Wörtern ihre Wortart zugeordnet. Diese Information könnte dann beim Training des maschinellen Lernens mitverwendet werden. So werden aus dem Wort „duck“ im Satz „I duck behind the rubber duck“ zwei verschiedene Features, eins für das Nomen und eins für das Verb. Im allen aktuellen Verfahren werden die beiden Wörter zu einem Feature zusammengefasst. Diese beiden Wörter haben jedoch inhaltlich eine andere Bedeutung, also könnte eine Unterscheidung solcher Wörter den Klassifizierer verbessern.

Auch im Bereich der *Feature Selection* sind Verbesserungen denkbar. So könnte eine speichereffizientere Implementierung und die Betrachtung von weniger Verfahren eine Evaluation von Verfahren mit mehr Features ermöglichen. Die einzelnen Kreuzvalidierungssätze haben nach dem *Preprocessing* im Mittel 160 000 Features. Es geht also ein signifikanter Teil der Information verloren, wenn 10 000 oder gar 100 Features ausgewählt werden. Die große Diskrepanz zwischen den Ergebnissen für 100 gewählte Features zu 10 000 gewählten Features lässt vermuten, dass die Auswahl von mehr Features die Performance weiter verbessern könnte. Dies erhöht jedoch die benötigten Ressourcen. Hier könnte eine Möglichkeit sein, nur die Verfahren, welche mit 10 000 Features die besten Ergebnisse geliefert haben, erneut zu testen. So könnten auch Tests mit Übernahme aller Features realisiert werden.

Zudem könnte eine Evaluation von Hauptkomponentenanalyse für die *Feature Extraction* durch Ausnutzung der Dünnbesetztheit der Feature Vektoren möglich sein. Die Feature Vektoren sind dünnbesetzt, da in jedem Text nur ein kleiner Bruchteil der Wörter vorkommt. Diese Eigenschaft lässt sich nutzen um die Hauptkomponentenanalyse speichereffizienter zu gestalten und so eine Berechnung zu ermöglichen. Zudem lässt sich auch hier der Aufwand dadurch reduzieren, dass diese Methode lediglich für die *Preprocessing*-Schritte evaluiert wird, welche gute Ergebnisse erzeugen.

Eine weitere Verbesserungsmöglichkeit stellt das Klassifizierungsverfahren dar. Hier wurde sich auf die Betrachtung eines Verfahrens beschränkt. Weitere Verfahren könnten bessere Ergebnisse liefern. Besonders spannend scheint hier der Bereich des Deep Learnings. Es gibt Ansätze, diese Techniken für die Empfehlung von Zeitungsartikeln zu nutzen [Zhe+18]. Hier wäre interessant, ob ein solcher Ansatz auch für die Klassifizierung gute Ergebnisse liefert.

Abschließend lässt sich sagen, dass aufgrund der erzielten Ergebnisse der Ansatz vielversprechend ist. Es wurden alle Kombinationen der betrachteten Verfahrensschritte evaluiert und die für die Anwendung am besten geeigneten Verfahren beleuchtet. Jedoch sollten noch weitere Versuche mit weiteren Verfahrensschritten unternommen werden, um die Ergebnisse zu verbessern.

Literatur

[AY11] Hakan Ayral und Sirma Yavuz. „An automated domain specific stop word generation method for natural language text classification“. In: *2011 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE. 2011, S. 500–503.

[Aza+18] Muhammad Azam u. a. „Feature Extraction based Text Classification using K-Nearest Neighbor Algorithm“. In: *IJCSNS Int. J. Comput. Sci. Netw. Secur* 18 (2018), S. 95–101.

[Bah+18] Said Bahassine u. a. „Feature selection using an improved Chi-square for Arabic text classification“. In: *Journal of King Saud University-Computer and Information Sciences* (2018).

[Chi92] Nancy Chinchor. „MUC-4 evaluation metrics“. In: *Proceedings of the 4th conference on Message understanding*. Association for Computational Linguistics. 1992, S. 22–29.

[CL11] Chih-Chung Chang und Chih-Jen Lin. „LIBSVM: A library for Support Vector Machines“. In: *ACM transactions on intelligent systems and technology (TIST)* 2.3 (2011), S. 1–27.

[DBM17] Xiangfeng Dai, Marwan Bikdash und Bradley Meyer. „From social media to public health surveillance: Word embedding based clustering method for twitter classification“. In: *SoutheastCon 2017*. IEEE. 2017, S. 1–7.

[Den+19] Xuelian Deng u. a. „Feature selection for text classification: A review“. In: *Multimedia Tools and Applications* 78.3 (2019), S. 3797–3816.

[DK17] Ayça Deniz und Hakan Ezgi Kiziloz. „Effects of various *Preprocessing* techniques to Turkish text categorization using n-gram features“. In: *2017 International Conference on Computer Science and Engineering (UBMK)*. IEEE. 2017, S. 655–660.

[Ehr+18] Claudia Ehrentraut u. a. „Detecting hospital-acquired infections: a document classification approach using *Support Vector Machine*s and gradient tree boosting“. In: *Health informatics journal* 24.1 (2018), S. 24–42.

[Fan+08] Rong-En Fan u. a. „LIBLINEAR: A library for large linear classification“. In: *Journal of machine learning research* 9.Aug (2008), S. 1871–1874.

[Faw06] Tom Fawcett. „An introduction to ROC analysis“. In: *Pattern recognition letters* 27.8 (2006), S. 861–874.

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

[Jol86] Ian T Jolliffe. „Principal components in regression analysis“. In: *Principal component analysis*. Springer, 1986, S. 129–155.

[Jur+10] Matjaz Juršic u. a. „Lemmagen: Multilingual lemmatisation with induced ripple-down rules“. In: *Journal of Universal Computer Science* 16.9 (2010), S. 1190–1214.

[KH18] HM Keerthi Kumar und BS Harish. „Classification of short text using various *Preprocessing* techniques: An empirical evaluation“. In: *Recent Findings in Intelligent Computing Techniques*. Springer, 2018, S. 19–30.

[Lab+18] Mahdieh Labani u. a. „A novel multivariate filter method for feature selection in text classification problems“. In: *Engineering Applications of Artificial Intelligence* 70 (2018), S. 25–37.

[Li+19] Ao Li u. a. „Spam Review Detection with Graph Convolutional Networks“. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, S. 2703–2711.

[Liu+17a] Chuan Liu u. a. „A new feature selection method based on a validity index of feature subset“. In: *Pattern Recognition Letters* 92 (2017), S. 1–8.

[Liu+17b] Jingzhou Liu u. a. „Deep learning for extreme multi-label text classification“. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2017, S. 115–124.

[LL99] Savio LY Lam und Dik Lun Lee. „Feature reduction for neural network based text categorization“. In: *Proceedings. 6th international conference on advanced systems for advanced applications*. IEEE. 1999, S. 195–202.

[Mal+18] Timo Malderle u. a. „Warning of Affected Users About an Identity Leak“. In: *International Conference on Soft Computing and Pattern Recognition*. Springer. 2018, S. 278–287.

[MP18] Marcin Michał Mirończuk und Jarosław Protasiewicz. „A recent overview of the state-of-the-art elements of text classification“. In: *Expert Systems with Applications* 106 (2018), S. 36–54.

[Ped+11] F. Pedregosa u. a. „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830.

[Pen+18] Hao Peng u. a. „Large-scale hierarchical text classification with recursively regularized deep graph-cnn“. In: *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee. 2018, S. 1063–1072.

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

[Reh+15] Abdur Rehman u. a. „Relative discrimination criterion–A novel feature ranking method for text data“. In: *Expert Systems with Applications* 42.7 (2015), S. 3670–3681.

[RN12a] Stuart Russell und Peter Norvig. *Künstliche Intelligenz*. Pearson Deutschland GmbH, 2012, S. 1006.

[RN12b] Stuart Russell und Peter Norvig. *Künstliche Intelligenz*. Pearson Deutschland GmbH, 2012, S. 863–868.

[RN12c] Stuart Russell und Peter Norvig. *Künstliche Intelligenz*. Pearson Deutschland GmbH, 2012, S. 825.

[Sil] Jason Silverstein. *Facebook data breach: Hundreds of millions of records exposed on Amazon server, according to UpGuard cybersecurity research firm - CBS News*. 2019 CBS Interactive Inc. https://www.cbsnews.com/news/millions-facebook-user-records-exposed-amazon-cloud-server/. Accessed: 2020-03-05.

[SMR08] Hinrich Schütze, Christopher D Manning und Prabhakar Raghavan. „Introduction to information retrieval“. In: *Proceedings of the international communication of association for computing machinery conference*. Bd. 4. 2008.

[SP+18] Manmohan Singh, Rajendra Pamula u. a. „Email Spam Classification by *Support Vector Machine*“. In: *2018 International Conference on Computing, Power and Communication Technologies (GUCON)*. IEEE. 2018, S. 878–882.

[SR03] Catarina Silva und Bernardete Ribeiro. „The importance of stop word removal on recall values in text categorization“. In: *Proceedings of the International Joint Conference on Neural Networks, 2003.* Bd. 3. IEEE. 2003, S. 1661–1666.

[Ste13] Ralf Steinberger. „Multilingual and cross-lingual news analysis in the Europe Media Monitor (EMM)“. In: *Information Retrieval Facility Conference*. Springer. 2013, S. 1–4.

[UG14] Alper Kursat Uysal und Serkan Gunal. „The impact of *Preprocessing* on text classification“. In: *Information Processing & Management* 50.1 (2014), S. 104–112.

[Uni10] Princeton University. *About WordNet.* 2010. URL: https://wordnet.princeton.edu (besucht am 30. 03. 2020).

[Xu18] Shuo Xu. „Bayesian Naïve Bayes classifiers to text classification“. In: *Journal of Information Science* 44.1 (2018), S. 48–59.

[Zhe+18] Guanjie Zheng u. a. „DRN: A deep reinforcement learning framework for news recommendation“. In: *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee. 2018, S. 167–176.

[Zuo+18] Zheming Zuo u. a. „Grooming detection using fuzzy-rough feature selection and text classification“. In: *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE. 2018, S. 1–8.

Abbildungsverzeichnis

1. Beispiel zum *Kernel-Trick* 6
2. Konfusionsmatrix 6

3. Schematische Darstellung der Verfahren zum *Preprocessing* 15
4. Schematische Darstellung der Verfahren zur *Feature Selection* 17

5. Schematische Darstellung des Evaluationsverfahren 21

6. Recall und Precision nach Anzahl betrachteter Features 23
7. Recall und Precision nach Feature Extractions-Verfahren 24
8. Ergebnisse zu Stemming und Lemmatisierung 24
9. Veränderung Recall und Precision bei Verwendung von Entfernung von Konjunktionen 25
10. Recall und Precision nach Feature Gewichtungsverfahren 25
11. Recall und Precision getrennt nach Stopword Entfernung 26
12. Satzzeichen-Entfernung und Zahlen-Ersetzung 26
13. Veränderung Recall und Precision bei Verwendung von Satzzeichen-Entfernung und Zahlen-Ersetzung 27

Tabellenverzeichnis

1. Übersicht der besten Klassifizierer nach Metrik 28

Selbstständigkeitserklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Bonn, 15. Juni 2020

Gina Caroline Muuss

A Ergebnisse

Auf den folgenden Seiten findet sich eine Übersichtstabelle über die Ergebnisse aller evaluierten Verfahren. Die Tabelle ist nach den Spalten von links nach rechts sortiert. Für die Metriken wurden die Zellen farblich markiert. Die Ergebnisse wurden für jede Metrik gleichmäßig in sieben Gruppen eingeteilt. Diese Gruppen sind dann gefärbt. Ein dunkles blau signalisiert einen besonders guten Wert, ein dunkles gelb einen besonders schlechten. Die Farben sind in Abbildung 1 dargestellt. Die mittleren Werte sind weiß markiert. Die besten Werte für jede Metrik sind fett hervorgehoben.

Abbildung 1.: *Die Farbskala*

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Keine	Ja	Ja	MRDC	Anzahl	100	0.721	0.480	0.576	0.720
Ja	Keine	Ja	Ja	MRDC	Anzahl	10000	0.815	0.497	0.617	0.814
Ja	Keine	Ja	Ja	MRDC	Binär	100	0.751	0.440	0.555	0.749
Ja	Keine	Ja	Ja	MRDC	Binär	10000	0.788	0.506	0.616	0.787
Ja	Keine	Ja	Ja	MRDC	tf-idf	100	0.709	0.485	0.575	0.709
Ja	Keine	Ja	Ja	MRDC	tf-idf	10000	0.800	0.495	0.611	0.798
Ja	Keine	Ja	Ja	RDC	Anzahl	100	0.774	0.476	0.589	0.772
Ja	Keine	Ja	Ja	RDC	Anzahl	10000	0.781	0.654	0.711	0.780
Ja	Keine	Ja	Ja	RDC	Binär	100	0.789	0.457	0.578	0.787

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Keine	Ja	Ja	RDC	Binär	10000	0.812	0.647	0.720	0.811
Ja	Keine	Ja	Ja	RDC	tf-idf	100	0.772	0.481	0.593	0.770
Ja	Keine	Ja	Ja	RDC	tf-idf	10000	0.793	0.677	0.730	0.793
Ja	Keine	Ja	Nein	MRDC	Anzahl	100	0.719	0.480	0.575	0.718
Ja	Keine	Ja	Nein	MRDC	Anzahl	10000	0.802	0.494	0.611	0.801
Ja	Keine	Ja	Nein	MRDC	Binär	100	0.741	0.441	0.553	0.740
Ja	Keine	Ja	Nein	MRDC	Binär	10000	0.759	0.504	0.605	0.758
Ja	Keine	Ja	Nein	MRDC	tf-idf	100	0.713	0.485	0.576	0.712
Ja	Keine	Ja	Nein	MRDC	tf-idf	10000	0.797	0.498	0.613	0.796
Ja	Keine	Ja	Nein	RDC	Anzahl	100	0.766	0.478	0.589	0.765
Ja	Keine	Ja	Nein	RDC	Anzahl	10000	0.830	0.654	0.731	0.829
Ja	Keine	Ja	Nein	RDC	Binär	100	0.791	0.450	0.574	0.789
Ja	Keine	Ja	Nein	RDC	Binär	10000	0.789	0.656	0.716	0.788
Ja	Keine	Ja	Nein	RDC	tf-idf	100	0.760	0.480	0.588	0.758
Ja	Keine	Ja	Nein	RDC	tf-idf	10000	0.794	0.687	0.736	0.794
Ja	Keine	Nein	Ja	MRDC	Anzahl	100	0.720	0.482	0.577	0.719
Ja	Keine	Nein	Ja	MRDC	Anzahl	10000	0.793	0.499	0.612	0.792
Ja	Keine	Nein	Ja	MRDC	Binär	100	0.749	0.442	0.556	0.747
Ja	Keine	Nein	Ja	MRDC	Binär	10000	0.780	0.502	0.611	0.779
Ja	Keine	Nein	Ja	MRDC	tf-idf	100	0.718	0.482	0.576	0.717
Ja	Keine	Nein	Ja	MRDC	tf-idf	10000	0.798	0.498	0.613	0.797
Ja	Keine	Nein	Ja	RDC	Anzahl	100	0.769	0.479	0.590	0.768
Ja	Keine	Nein	Ja	RDC	Anzahl	10000	0.803	0.663	0.723	0.802
Ja	Keine	Nein	Ja	RDC	Binär	100	0.793	0.454	0.577	0.791

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Keine	Nein	Ja	RDC	Binär	10000	0.804	0.666	0.728	0.803
Ja	Keine	Nein	Ja	RDC	tf-idf	100	0.771	0.481	0.592	0.769
Ja	Keine	Nein	Ja	RDC	tf-idf	10000	0.811	0.683	0.741	0.811
Ja	Keine	Nein	Nein	MRDC	Anzahl	100	0.717	0.480	0.575	0.716
Ja	Keine	Nein	Nein	MRDC	Anzahl	10000	0.804	0.499	0.615	0.803
Ja	Keine	Nein	Nein	MRDC	Binär	100	0.740	0.440	0.552	0.739
Ja	Keine	Nein	Nein	MRDC	Binär	10000	0.791	0.519	0.626	0.790
Ja	Keine	Nein	Nein	MRDC	tf-idf	100	0.706	0.486	0.575	0.705
Ja	Keine	Nein	Nein	MRDC	tf-idf	10000	0.795	0.501	0.614	0.793
Ja	Keine	Nein	Nein	RDC	Anzahl	100	0.766	0.481	0.590	0.764
Ja	Keine	Nein	Nein	RDC	Anzahl	10000	0.792	0.662	0.720	0.791
Ja	Keine	Nein	Nein	RDC	Binär	100	0.801	0.446	0.573	0.799
Ja	Keine	Nein	Nein	RDC	Binär	10000	0.774	0.678	0.721	0.774
Ja	Keine	Nein	Nein	RDC	tf-idf	100	0.761	0.478	0.587	0.759
Ja	Keine	Nein	Nein	RDC	tf-idf	10000	0.810	0.690	**0.745**	0.810
Ja	Lemmatisierung	Ja	Ja	MRDC	Anzahl	100	0.723	0.482	0.578	0.722
Ja	Lemmatisierung	Ja	Ja	MRDC	Anzahl	10000	0.806	0.495	0.613	0.805
Ja	Lemmatisierung	Ja	Ja	MRDC	Binär	100	0.767	0.436	0.556	0.766
Ja	Lemmatisierung	Ja	Ja	MRDC	Binär	10000	0.778	0.502	0.610	0.776
Ja	Lemmatisierung	Ja	Ja	MRDC	tf-idf	100	0.711	0.490	0.579	0.710
Ja	Lemmatisierung	Ja	Ja	MRDC	tf-idf	10000	0.814	0.495	0.615	0.813
Ja	Lemmatisierung	Ja	Ja	RDC	Anzahl	100	0.774	0.481	0.593	0.773
Ja	Lemmatisierung	Ja	Ja	RDC	Anzahl	10000	0.782	0.656	0.712	0.782
Ja	Lemmatisierung	Ja	Ja	RDC	Binär	100	0.806	0.453	0.579	0.804

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Lemmatisierung	Ja	Ja	RDC	Binär	10000	0.788	0.646	0.710	0.788
Ja	Lemmatisierung	Ja	Ja	RDC	tf-idf	100	0.775	0.482	0.594	0.774
Ja	Lemmatisierung	Ja	Ja	RDC	tf-idf	10000	0.798	0.683	0.736	0.798
Ja	Lemmatisierung	Ja	Nein	MRDC	Anzahl	100	0.718	0.480	0.575	0.717
Ja	Lemmatisierung	Ja	Nein	MRDC	Anzahl	10000	0.816	0.498	0.618	0.814
Ja	Lemmatisierung	Ja	Nein	MRDC	Binär	100	0.742	0.441	0.553	0.741
Ja	Lemmatisierung	Ja	Nein	MRDC	Binär	10000	0.787	0.506	0.616	0.786
Ja	Lemmatisierung	Ja	Nein	MRDC	tf-idf	100	0.712	0.485	0.576	0.711
Ja	Lemmatisierung	Ja	Nein	MRDC	tf-idf	10000	0.804	0.498	0.615	0.803
Ja	Lemmatisierung	Ja	Nein	RDC	Anzahl	100	0.765	0.481	0.590	0.764
Ja	Lemmatisierung	Ja	Nein	RDC	Anzahl	10000	0.834	0.656	0.734	0.833
Ja	Lemmatisierung	Ja	Nein	RDC	Binär	100	0.798	0.449	0.574	0.796
Ja	Lemmatisierung	Ja	Nein	RDC	Binär	10000	0.795	0.660	0.721	0.795
Ja	Lemmatisierung	Ja	Nein	RDC	tf-idf	100	0.759	0.479	0.588	0.758
Ja	Lemmatisierung	Ja	Nein	RDC	tf-idf	10000	0.793	0.686	0.735	0.793
Ja	Lemmatisierung	Nein	Ja	MRDC	Anzahl	100	0.726	0.478	0.576	0.725
Ja	Lemmatisierung	Nein	Ja	MRDC	Anzahl	10000	0.806	0.499	0.616	0.805
Ja	Lemmatisierung	Nein	Ja	MRDC	Binär	100	0.758	0.443	0.558	0.756
Ja	Lemmatisierung	Nein	Ja	MRDC	Binär	10000	0.798	0.508	0.620	0.797
Ja	Lemmatisierung	Nein	Ja	MRDC	tf-idf	100	0.728	0.481	0.578	0.727
Ja	Lemmatisierung	Nein	Ja	MRDC	tf-idf	10000	0.794	0.494	0.609	0.793
Ja	Lemmatisierung	Nein	Ja	RDC	Anzahl	100	0.774	0.482	0.594	0.773
Ja	Lemmatisierung	Nein	Ja	RDC	Anzahl	10000	0.822	0.658	0.730	0.822
Ja	Lemmatisierung	Nein	Ja	RDC	Binär	100	0.807	0.453	0.580	0.805

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Lemmatisierung	Nein	Ja	RDC	Binär	10000	0.814	0.658	0.728	0.814
Ja	Lemmatisierung	Nein	Ja	RDC	tf-idf	100	0.775	0.482	0.594	0.774
Ja	Lemmatisierung	Nein	Ja	RDC	tf-idf	10000	0.800	0.685	0.737	0.799
Ja	Lemmatisierung	Nein	Nein	MRDC	Anzahl	100	0.718	0.481	0.575	0.717
Ja	Lemmatisierung	Nein	Nein	MRDC	Anzahl	10000	0.821	0.503	0.623	0.819
Ja	Lemmatisierung	Nein	Nein	MRDC	Binär	100	0.749	0.434	0.549	0.748
Ja	Lemmatisierung	Nein	Nein	MRDC	Binär	10000	0.790	0.518	0.625	0.789
Ja	Lemmatisierung	Nein	Nein	MRDC	tf-idf	100	0.709	0.482	0.573	0.708
Ja	Lemmatisierung	Nein	Nein	MRDC	tf-idf	10000	0.792	0.498	0.611	0.791
Ja	Lemmatisierung	Nein	Nein	RDC	Anzahl	100	0.762	0.483	0.591	0.761
Ja	Lemmatisierung	Nein	Nein	RDC	Anzahl	10000	0.752	0.674	0.710	0.752
Ja	Lemmatisierung	Nein	Nein	RDC	Binär	100	0.795	0.453	0.577	0.793
Ja	Lemmatisierung	Nein	Nein	RDC	Binär	10000	0.790	0.669	0.723	0.789
Ja	Lemmatisierung	Nein	Nein	RDC	tf-idf	100	0.760	0.480	0.588	0.759
Ja	Lemmatisierung	Nein	Nein	RDC	tf-idf	10000	0.804	0.690	0.742	0.804
Ja	Stemming	Ja	Ja	MRDC	Anzahl	100	0.806	0.305	0.440	0.802
Ja	Stemming	Ja	Ja	MRDC	Anzahl	10000	0.811	0.505	0.622	0.810
Ja	Stemming	Ja	Ja	MRDC	Binär	100	0.778	0.329	0.458	0.775
Ja	Stemming	Ja	Ja	MRDC	Binär	10000	0.753	0.516	0.612	0.752
Ja	Stemming	Ja	Ja	MRDC	tf-idf	100	0.685	0.382	0.479	0.684
Ja	Stemming	Ja	Ja	MRDC	tf-idf	10000	0.764	0.504	0.607	0.763
Ja	Stemming	Ja	Ja	RDC	Anzahl	100	0.736	0.430	0.536	0.735
Ja	Stemming	Ja	Ja	RDC	Anzahl	10000	0.774	0.644	0.702	0.773
Ja	Stemming	Ja	Ja	RDC	Binär	100	0.743	0.414	0.528	0.742

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Stemming	Ja	Ja	RDC	Binär	10000	0.803	0.632	0.707	0.803
Ja	Stemming	Ja	Ja	RDC	tf-idf	100	0.686	0.441	0.533	0.685
Ja	Stemming	Ja	Ja	RDC	tf-idf	10000	0.789	0.671	0.725	0.789
Ja	Stemming	Ja	Nein	MRDC	Anzahl	100	0.795	0.308	0.438	0.791
Ja	Stemming	Ja	Nein	MRDC	Anzahl	10000	0.813	0.511	0.627	0.812
Ja	Stemming	Ja	Nein	MRDC	Binär	100	0.717	0.356	0.465	0.715
Ja	Stemming	Ja	Nein	MRDC	Binär	10000	0.764	0.524	0.621	0.763
Ja	Stemming	Ja	Nein	MRDC	tf-idf	100	0.617	0.420	0.487	0.616
Ja	Stemming	Ja	Nein	MRDC	tf-idf	10000	0.782	0.514	0.620	0.781
Ja	Stemming	Ja	Nein	RDC	Anzahl	100	0.743	0.406	0.517	0.741
Ja	Stemming	Ja	Nein	RDC	Anzahl	10000	0.786	0.648	0.710	0.786
Ja	Stemming	Ja	Nein	RDC	Binär	100	0.708	0.419	0.524	0.707
Ja	Stemming	Ja	Nein	RDC	Binär	10000	0.809	0.645	0.717	0.809
Ja	Stemming	Ja	Nein	RDC	tf-idf	100	0.693	0.439	0.532	0.692
Ja	Stemming	Ja	Nein	RDC	tf-idf	10000	0.797	0.677	0.732	0.797
Ja	Stemming	Nein	Ja	MRDC	Anzahl	100	0.817	0.306	0.442	0.813
Ja	Stemming	Nein	Ja	MRDC	Anzahl	10000	0.815	0.507	0.625	0.814
Ja	Stemming	Nein	Ja	MRDC	Binär	100	0.752	0.342	0.462	0.750
Ja	Stemming	Nein	Ja	MRDC	Binär	10000	0.766	0.519	0.619	0.765
Ja	Stemming	Nein	Ja	MRDC	tf-idf	100	0.659	0.387	0.478	0.657
Ja	Stemming	Nein	Ja	MRDC	tf-idf	10000	0.767	0.505	0.609	0.766
Ja	Stemming	Nein	Ja	RDC	Anzahl	100	0.741	0.431	0.537	0.739
Ja	Stemming	Nein	Ja	RDC	Anzahl	10000	0.852	0.642	0.732	0.851
Ja	Stemming	Nein	Ja	RDC	Binär	100	0.743	0.416	0.529	0.741

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Ja	Stemming	Nein	Ja	RDC	Binär	10000	0.783	0.645	0.707	0.782
Ja	Stemming	Nein	Ja	RDC	tf-idf	100	0.689	0.442	0.534	0.687
Ja	Stemming	Nein	Ja	RDC	tf-idf	10000	0.791	0.677	0.728	0.790
Ja	Stemming	Nein	Nein	MRDC	Anzahl	100	0.814	0.295	0.433	0.810
Ja	Stemming	Nein	Nein	MRDC	Anzahl	10000	0.800	0.513	0.625	0.799
Ja	Stemming	Nein	Nein	MRDC	Binär	100	0.716	0.357	0.465	0.714
Ja	Stemming	Nein	Nein	MRDC	Binär	10000	0.762	0.526	0.621	0.761
Ja	Stemming	Nein	Nein	MRDC	tf-idf	100	0.623	0.420	0.487	0.622
Ja	Stemming	Nein	Nein	MRDC	tf-idf	10000	0.776	0.518	0.621	0.774
Ja	Stemming	Nein	Nein	RDC	Anzahl	100	0.744	0.408	0.519	0.742
Ja	Stemming	Nein	Nein	RDC	Anzahl	10000	0.823	0.642	0.721	0.823
Ja	Stemming	Nein	Nein	RDC	Binär	100	0.700	0.422	0.524	0.698
Ja	Stemming	Nein	Nein	RDC	Binär	10000	0.788	0.646	0.710	0.788
Ja	Stemming	Nein	Nein	RDC	tf-idf	100	0.695	0.438	0.531	0.694
Ja	Stemming	Nein	Nein	RDC	tf-idf	10000	0.804	0.678	0.735	0.804
Nein	Keine	Ja	Ja	MRDC	Anzahl	100	0.719	0.480	0.575	0.718
Nein	Keine	Ja	Ja	MRDC	Anzahl	10000	0.813	0.496	0.615	0.812
Nein	Keine	Ja	Ja	MRDC	Binär	100	0.740	0.448	0.557	0.739
Nein	Keine	Ja	Ja	MRDC	Binär	10000	0.785	0.510	0.618	0.784
Nein	Keine	Ja	Ja	MRDC	tf-idf	100	0.710	0.483	0.575	0.710
Nein	Keine	Ja	Ja	MRDC	tf-idf	10000	0.803	0.494	0.612	0.802
Nein	Keine	Ja	Ja	RDC	Anzahl	100	0.775	0.477	0.590	0.773
Nein	Keine	Ja	Ja	RDC	Anzahl	10000	0.764	0.655	0.704	0.764
Nein	Keine	Ja	Ja	RDC	Binär	100	0.789	0.457	0.578	0.788

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Nein	Keine	Ja	Ja	RDC	Binär	10000	0.818	0.644	0.721	0.818
Nein	Keine	Ja	Ja	RDC	tf-idf	100	0.773	0.483	0.594	0.772
Nein	Keine	Ja	Ja	RDC	tf-idf	10000	0.800	0.681	0.735	0.800
Nein	Keine	Ja	Nein	MRDC	Anzahl	100	0.720	0.480	0.575	0.719
Nein	Keine	Ja	Nein	MRDC	Anzahl	10000	0.818	0.497	0.618	0.817
Nein	Keine	Ja	Nein	MRDC	Binär	100	0.744	0.440	0.553	0.742
Nein	Keine	Ja	Nein	MRDC	Binär	10000	0.786	0.512	0.620	0.785
Nein	Keine	Ja	Nein	MRDC	tf-idf	100	0.712	0.483	0.575	0.711
Nein	Keine	Ja	Nein	MRDC	tf-idf	10000	0.795	0.501	0.615	0.793
Nein	Keine	Ja	Nein	RDC	Anzahl	100	0.769	0.480	0.591	0.767
Nein	Keine	Ja	Nein	RDC	Anzahl	10000	0.819	0.657	0.727	0.818
Nein	Keine	Ja	Nein	RDC	Binär	100	0.787	0.449	0.572	0.786
Nein	Keine	Ja	Nein	RDC	Binär	10000	0.795	0.659	0.720	0.795
Nein	Keine	Ja	Nein	RDC	tf-idf	100	0.759	0.479	0.588	0.758
Nein	Keine	Ja	Nein	RDC	tf-idf	10000	0.789	0.691	0.736	0.788
Nein	Keine	Nein	Ja	MRDC	Anzahl	100	0.722	0.479	0.575	0.721
Nein	Keine	Nein	Ja	MRDC	Anzahl	10000	0.816	0.501	0.620	0.815
Nein	Keine	Nein	Ja	MRDC	Binär	100	0.752	0.440	0.555	0.750
Nein	Keine	Nein	Ja	MRDC	Binär	10000	0.789	0.508	0.618	0.788
Nein	Keine	Nein	Ja	MRDC	tf-idf	100	0.710	0.485	0.576	0.710
Nein	Keine	Nein	Ja	MRDC	tf-idf	10000	0.799	0.501	0.616	0.798
Nein	Keine	Nein	Ja	RDC	Anzahl	100	0.771	0.480	0.591	0.770
Nein	Keine	Nein	Ja	RDC	Anzahl	10000	0.833	0.657	0.734	0.832
Nein	Keine	Nein	Ja	RDC	Binär	100	0.795	0.456	0.579	0.793

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Nein	Keine	Nein	Ja	RDC	Binär	10000	0.790	0.668	0.723	0.790
Nein	Keine	Nein	Ja	RDC	tf-idf	100	0.769	0.480	0.591	0.767
Nein	Keine	Nein	Ja	RDC	tf-idf	10000	0.804	0.684	0.739	0.804
Nein	Keine	Nein	Nein	MRDC	Anzahl	100	0.718	0.482	0.576	0.717
Nein	Keine	Nein	Nein	MRDC	Anzahl	10000	0.804	0.507	0.621	0.803
Nein	Keine	Nein	Nein	MRDC	Binär	100	0.738	0.444	0.554	0.737
Nein	Keine	Nein	Nein	MRDC	Binär	10000	0.798	0.517	0.627	0.797
Nein	Keine	Nein	Nein	MRDC	tf-idf	100	0.709	0.485	0.576	0.709
Nein	Keine	Nein	Nein	MRDC	tf-idf	10000	0.803	0.502	0.617	0.802
Nein	Keine	Nein	Nein	RDC	Anzahl	100	0.766	0.480	0.590	0.765
Nein	Keine	Nein	Nein	RDC	Anzahl	10000	0.774	0.674	0.719	0.774
Nein	Keine	Nein	Nein	RDC	Binär	100	0.788	0.452	0.574	0.787
Nein	Keine	Nein	Nein	RDC	Binär	10000	0.793	0.673	0.727	0.793
Nein	Keine	Nein	Nein	RDC	tf-idf	100	0.761	0.481	0.589	0.759
Nein	Keine	Nein	Nein	RDC	tf-idf	10000	0.805	0.692	0.743	0.805
Nein	Lemmatisierung	Ja	Ja	MRDC	Anzahl	100	0.720	0.483	0.577	0.719
Nein	Lemmatisierung	Ja	Ja	MRDC	Anzahl	10000	0.811	0.496	0.615	0.810
Nein	Lemmatisierung	Ja	Ja	MRDC	Binär	100	0.747	0.443	0.556	0.746
Nein	Lemmatisierung	Ja	Ja	MRDC	Binär	10000	0.777	0.508	0.614	0.775
Nein	Lemmatisierung	Ja	Ja	MRDC	tf-idf	100	0.710	0.487	0.577	0.710
Nein	Lemmatisierung	Ja	Ja	MRDC	tf-idf	10000	0.793	0.492	0.607	0.792
Nein	Lemmatisierung	Ja	Ja	RDC	Anzahl	100	0.770	0.478	0.590	0.769
Nein	Lemmatisierung	Ja	Ja	RDC	Anzahl	10000	0.772	0.666	0.715	0.772
Nein	Lemmatisierung	Ja	Ja	RDC	Binär	100	0.790	0.455	0.577	0.789

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Nein	Lemmatisierung	Ja	Ja	RDC	Binär	10000	0.801	0.646	0.715	0.801
Nein	Lemmatisierung	Ja	Ja	RDC	tf-idf	100	0.770	0.480	0.591	0.768
Nein	Lemmatisierung	Ja	Ja	RDC	tf-idf	10000	0.782	0.681	0.728	0.782
Nein	Lemmatisierung	Ja	Nein	MRDC	Anzahl	100	0.716	0.482	0.576	0.715
Nein	Lemmatisierung	Ja	Nein	MRDC	Anzahl	10000	0.814	0.499	0.618	0.812
Nein	Lemmatisierung	Ja	Nein	MRDC	Binär	100	0.736	0.443	0.553	0.735
Nein	Lemmatisierung	Ja	Nein	MRDC	Binär	10000	0.793	0.511	0.621	0.791
Nein	Lemmatisierung	Ja	Nein	MRDC	tf-idf	100	0.703	0.487	0.575	0.703
Nein	Lemmatisierung	Ja	Nein	MRDC	tf-idf	10000	0.807	0.496	0.614	0.806
Nein	Lemmatisierung	Ja	Nein	RDC	Anzahl	100	0.766	0.480	0.590	0.765
Nein	Lemmatisierung	Ja	Nein	RDC	Anzahl	10000	0.830	0.654	0.731	0.829
Nein	Lemmatisierung	Ja	Nein	RDC	Binär	100	0.794	0.448	0.573	0.793
Nein	Lemmatisierung	Ja	Nein	RDC	Binär	10000	0.800	0.655	0.720	0.800
Nein	Lemmatisierung	Ja	Nein	RDC	tf-idf	100	0.761	0.481	0.589	0.759
Nein	Lemmatisierung	Ja	Nein	RDC	tf-idf	10000	0.788	0.684	0.731	0.788
Nein	Lemmatisierung	Nein	Ja	MRDC	Anzahl	100	0.723	0.483	0.578	0.722
Nein	Lemmatisierung	Nein	Ja	MRDC	Anzahl	10000	0.824	0.499	0.621	0.823
Nein	Lemmatisierung	Nein	Ja	MRDC	Binär	100	0.746	0.446	0.557	0.744
Nein	Lemmatisierung	Nein	Ja	MRDC	Binär	10000	0.798	0.506	0.619	0.797
Nein	Lemmatisierung	Nein	Ja	MRDC	tf-idf	100	0.718	0.480	0.575	0.717
Nein	Lemmatisierung	Nein	Ja	MRDC	tf-idf	10000	0.804	0.498	0.615	0.803
Nein	Lemmatisierung	Nein	Ja	RDC	Anzahl	100	0.771	0.478	0.590	0.770
Nein	Lemmatisierung	Nein	Ja	RDC	Anzahl	10000	0.834	0.655	0.733	0.833
Nein	Lemmatisierung	Nein	Ja	RDC	Binär	100	0.788	0.455	0.576	0.787

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Nein	Lemmatisierung	Nein	Ja	RDC	Binär	10000	0.809	0.665	0.729	0.808
Nein	Lemmatisierung	Nein	Ja	RDC	tf-idf	100	0.773	0.483	0.594	0.772
Nein	Lemmatisierung	Nein	Ja	RDC	tf-idf	10000	0.802	0.687	0.739	0.801
Nein	Lemmatisierung	Nein	Nein	MRDC	Anzahl	100	0.721	0.481	0.576	0.720
Nein	Lemmatisierung	Nein	Nein	MRDC	Anzahl	10000	0.800	0.501	0.616	0.799
Nein	Lemmatisierung	Nein	Nein	MRDC	Binär	100	0.739	0.440	0.551	0.737
Nein	Lemmatisierung	Nein	Nein	MRDC	Binär	10000	0.787	0.515	0.622	0.785
Nein	Lemmatisierung	Nein	Nein	MRDC	tf-idf	100	0.710	0.484	0.575	0.710
Nein	Lemmatisierung	Nein	Nein	MRDC	tf-idf	10000	0.803	0.501	0.617	0.802
Nein	Lemmatisierung	Nein	Nein	RDC	Anzahl	100	0.760	0.481	0.589	0.758
Nein	Lemmatisierung	Nein	Nein	RDC	Anzahl	10000	0.755	0.672	0.711	0.755
Nein	Lemmatisierung	Nein	Nein	RDC	Binär	100	0.797	0.447	0.573	0.795
Nein	Lemmatisierung	Nein	Nein	RDC	Binär	10000	0.786	0.671	0.723	0.786
Nein	Lemmatisierung	Nein	Nein	RDC	tf-idf	100	0.760	0.479	0.588	0.759
Nein	Lemmatisierung	Nein	Nein	RDC	tf-idf	10000	0.793	**0.694**	0.739	0.793
Nein	Stemming	Ja	Ja	MRDC	Anzahl	100	0.809	0.308	0.443	0.805
Nein	Stemming	Ja	Ja	MRDC	Anzahl	10000	0.807	0.505	0.621	0.806
Nein	Stemming	Ja	Ja	MRDC	Binär	100	0.776	0.329	0.457	0.773
Nein	Stemming	Ja	Ja	MRDC	Binär	10000	0.760	0.515	0.613	0.759
Nein	Stemming	Ja	Ja	MRDC	tf-idf	100	0.656	0.386	0.476	0.654
Nein	Stemming	Ja	Ja	MRDC	tf-idf	10000	0.761	0.505	0.607	0.760
Nein	Stemming	Ja	Ja	RDC	Anzahl	100	0.737	0.428	0.534	0.735
Nein	Stemming	Ja	Ja	RDC	Anzahl	10000	0.768	0.646	0.701	0.768
Nein	Stemming	Ja	Ja	RDC	Binär	100	0.742	0.412	0.526	0.740

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Nein	Stemming	Ja	Ja	RDC	Binär	10000	0.808	0.628	0.707	0.807
Nein	Stemming	Ja	Ja	RDC	tf-idf	100	0.687	0.442	0.534	0.686
Nein	Stemming	Ja	Ja	RDC	tf-idf	10000	0.813	0.667	0.732	0.812
Nein	Stemming	Ja	Nein	MRDC	Anzahl	100	0.786	0.309	0.437	0.783
Nein	Stemming	Ja	Nein	MRDC	Anzahl	10000	0.820	0.511	0.629	0.818
Nein	Stemming	Ja	Nein	MRDC	Binär	100	0.717	0.354	0.463	0.715
Nein	Stemming	Ja	Nein	MRDC	Binär	10000	0.758	0.520	0.617	0.757
Nein	Stemming	Ja	Nein	MRDC	tf-idf	100	0.645	0.406	0.484	0.644
Nein	Stemming	Ja	Nein	MRDC	tf-idf	10000	0.780	0.509	0.616	0.778
Nein	Stemming	Ja	Nein	RDC	Anzahl	100	0.744	0.406	0.518	0.742
Nein	Stemming	Ja	Nein	RDC	Anzahl	10000	0.805	0.652	0.720	0.805
Nein	Stemming	Ja	Nein	RDC	Binär	100	0.707	0.421	0.525	0.705
Nein	Stemming	Ja	Nein	RDC	Binär	10000	0.807	0.642	0.715	0.806
Nein	Stemming	Ja	Nein	RDC	tf-idf	100	0.694	0.437	0.530	0.693
Nein	Stemming	Ja	Nein	RDC	tf-idf	10000	0.790	0.679	0.730	0.790
Nein	Stemming	Nein	Ja	MRDC	Anzahl	100	0.799	0.307	0.440	0.795
Nein	Stemming	Nein	Ja	MRDC	Anzahl	10000	0.804	0.508	0.623	0.803
Nein	Stemming	Nein	Ja	MRDC	Binär	100	0.754	0.344	0.464	0.751
Nein	Stemming	Nein	Ja	MRDC	Binär	10000	0.757	0.515	0.613	0.756
Nein	Stemming	Nein	Ja	MRDC	tf-idf	100	0.660	0.386	0.477	0.658
Nein	Stemming	Nein	Ja	MRDC	tf-idf	10000	0.784	0.511	0.618	0.783
Nein	Stemming	Nein	Ja	RDC	Anzahl	100	0.730	0.430	0.534	0.729
Nein	Stemming	Nein	Ja	RDC	Anzahl	10000	**0.860**	0.642	0.735	**0.860**
Nein	Stemming	Nein	Ja	RDC	Binär	100	0.732	0.415	0.526	0.730

Konjunktionen entfernt	Wortstamm-reduktion	Stopwords entfernt	Zahlen ersetzt und Satzzeichen entfernt	Feature Selection Methode	Feature Gewichtung	Anzahl Features	Recall	Precision	F_1	F_{19}
Nein	Stemming	Nein	Ja	RDC	Binär	10000	0.774	0.647	0.704	0.774
Nein	Stemming	Nein	Ja	RDC	tf-idf	100	0.684	0.442	0.533	0.682
Nein	Stemming	Nein	Ja	RDC	tf-idf	10000	0.806	0.678	0.736	0.805
Nein	Stemming	Nein	Nein	MRDC	Anzahl	100	0.801	0.305	0.435	0.797
Nein	Stemming	Nein	Nein	MRDC	Anzahl	10000	0.816	0.514	0.630	0.814
Nein	Stemming	Nein	Nein	MRDC	Binär	100	0.718	0.356	0.465	0.716
Nein	Stemming	Nein	Nein	MRDC	Binär	10000	0.760	0.533	0.626	0.759
Nein	Stemming	Nein	Nein	MRDC	tf-idf	100	0.649	0.406	0.485	0.648
Nein	Stemming	Nein	Nein	MRDC	tf-idf	10000	0.762	0.513	0.613	0.761
Nein	Stemming	Nein	Nein	RDC	Anzahl	100	0.740	0.406	0.517	0.738
Nein	Stemming	Nein	Nein	RDC	Anzahl	10000	0.779	0.650	0.706	0.779
Nein	Stemming	Nein	Nein	RDC	Binär	100	0.704	0.419	0.523	0.702
Nein	Stemming	Nein	Nein	RDC	Binär	10000	0.793	0.645	0.711	0.792
Nein	Stemming	Nein	Nein	RDC	tf-idf	100	0.684	0.437	0.527	0.683
Nein	Stemming	Nein	Nein	RDC	tf-idf	10000	0.813	0.684	0.743	0.813

B Grafiken

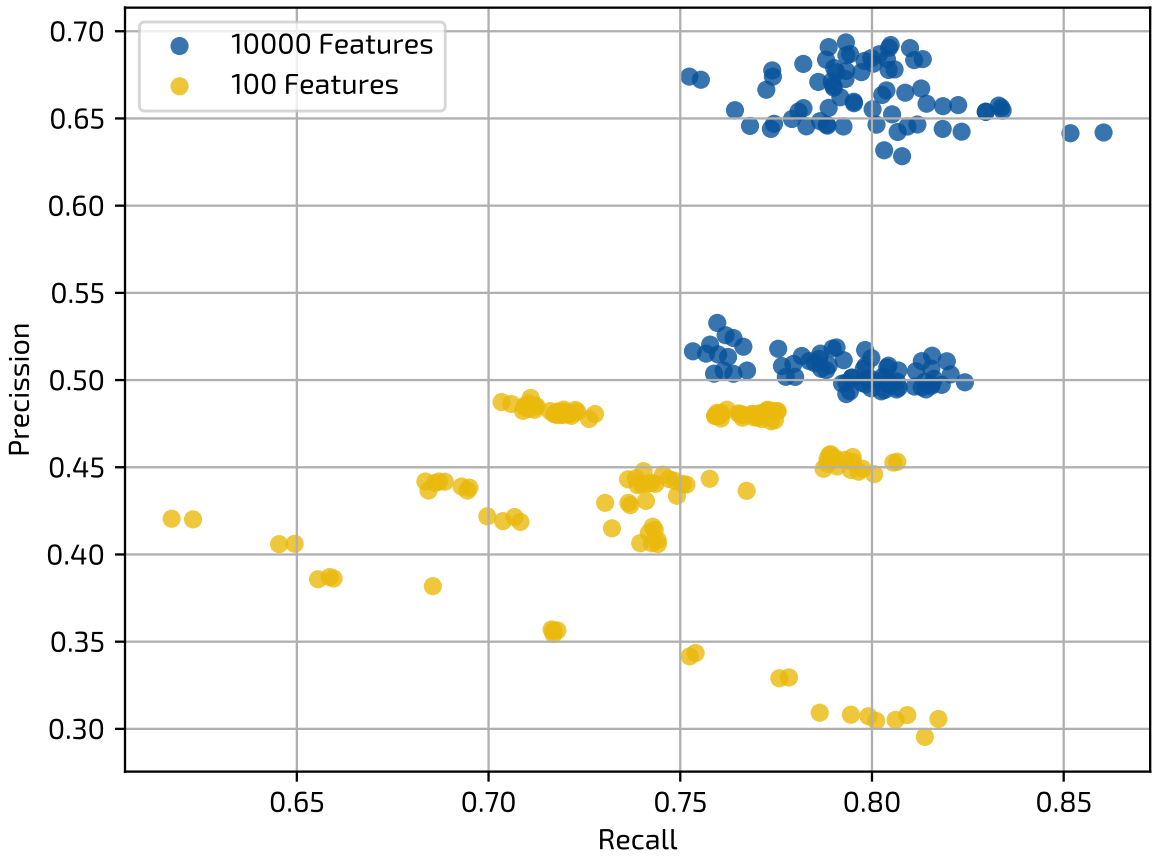


Abbildung 6.: *Recall und Precision nach Anzahl betrachteter Features*

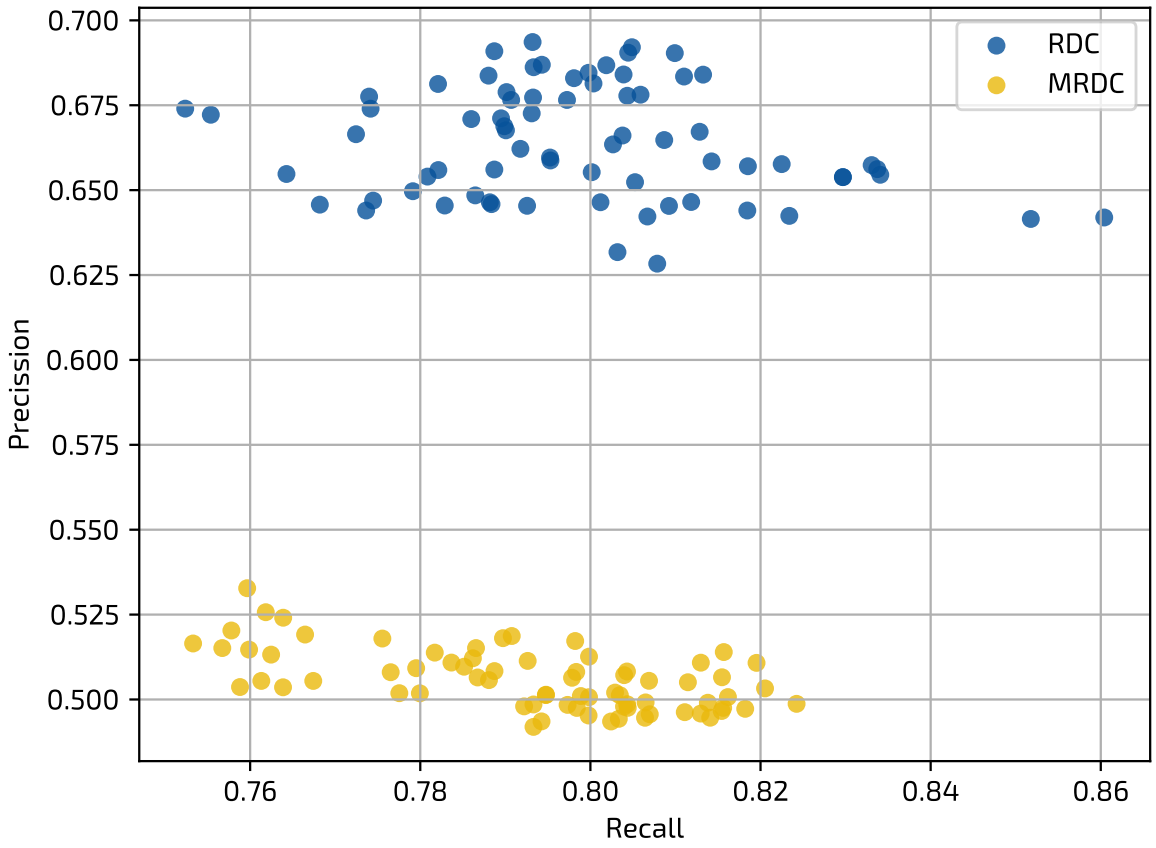
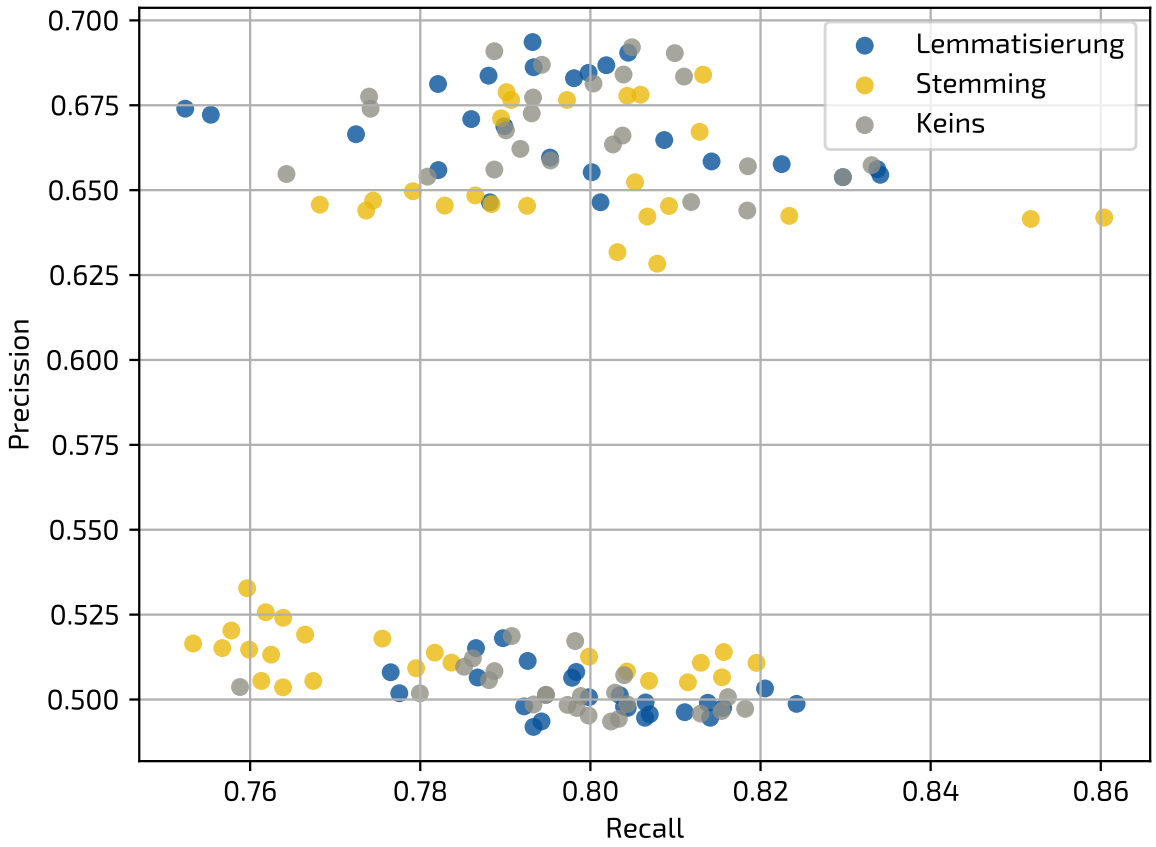
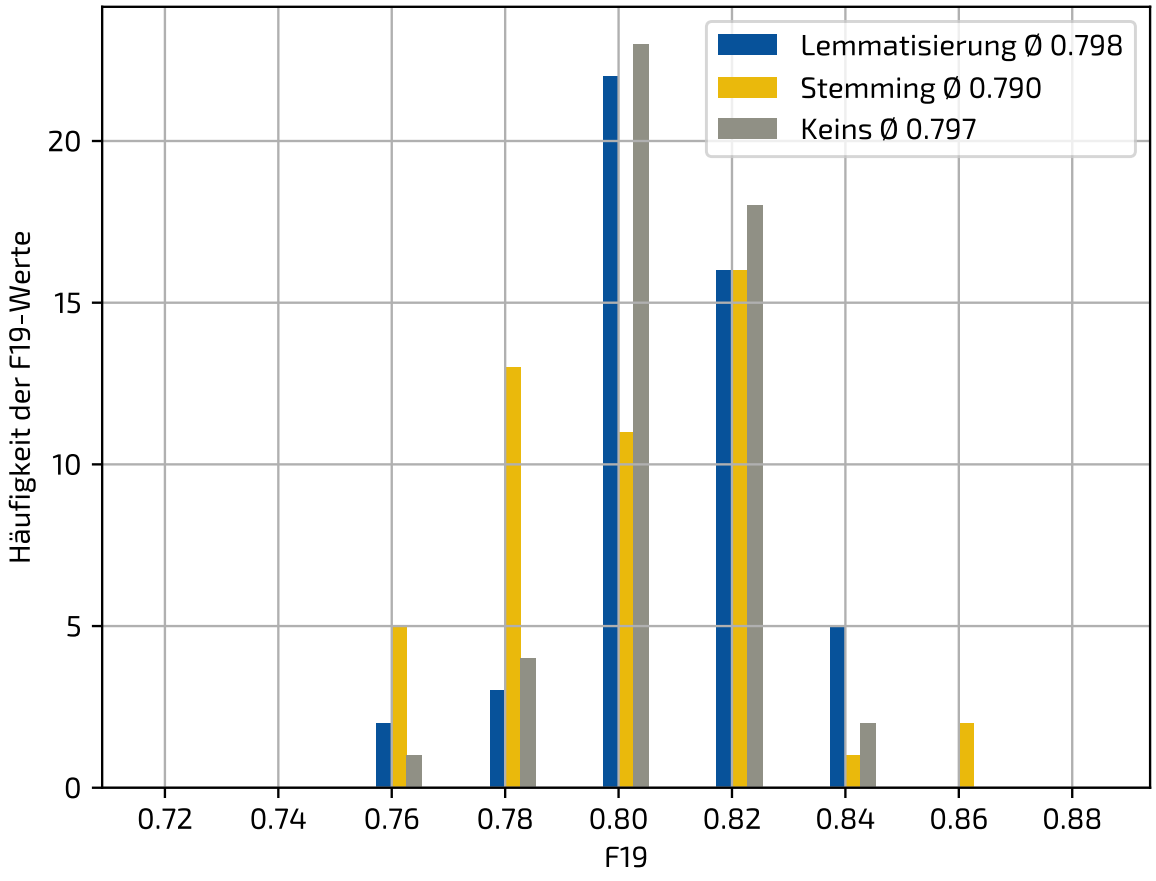


Abbildung 7.: *Recall und Precision nach Feature Extractionsverfahren*



(A) *Precision gegen Recall*



(B) *Verteilung F_{19}*

Abbildung 8.: *Ergebnisse zu Stemming und Lemmatisierung*

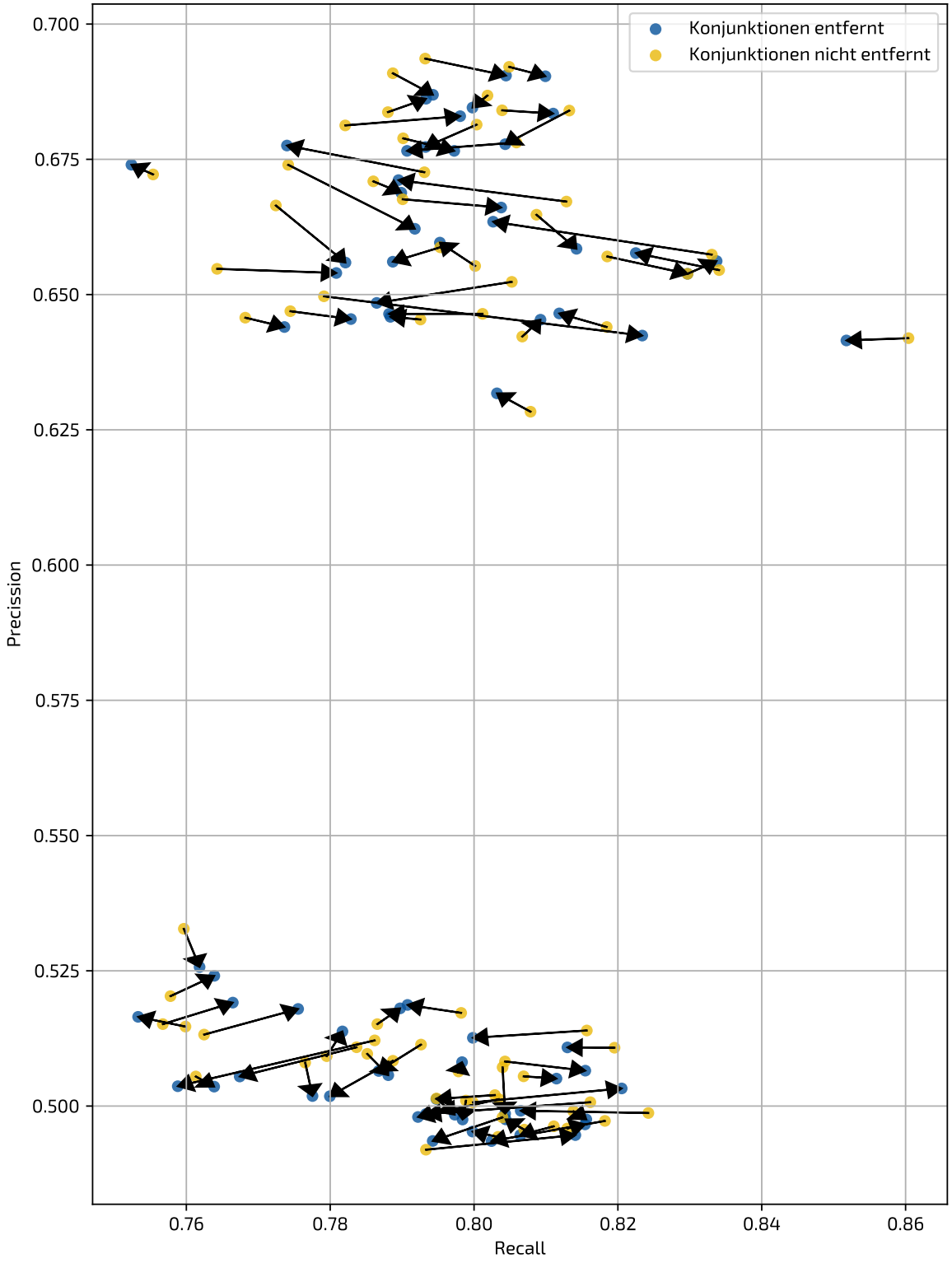


Abbildung 9.: *Veränderung Recall und Precision bei Verwendung von Entfernung von Konjunktionen*

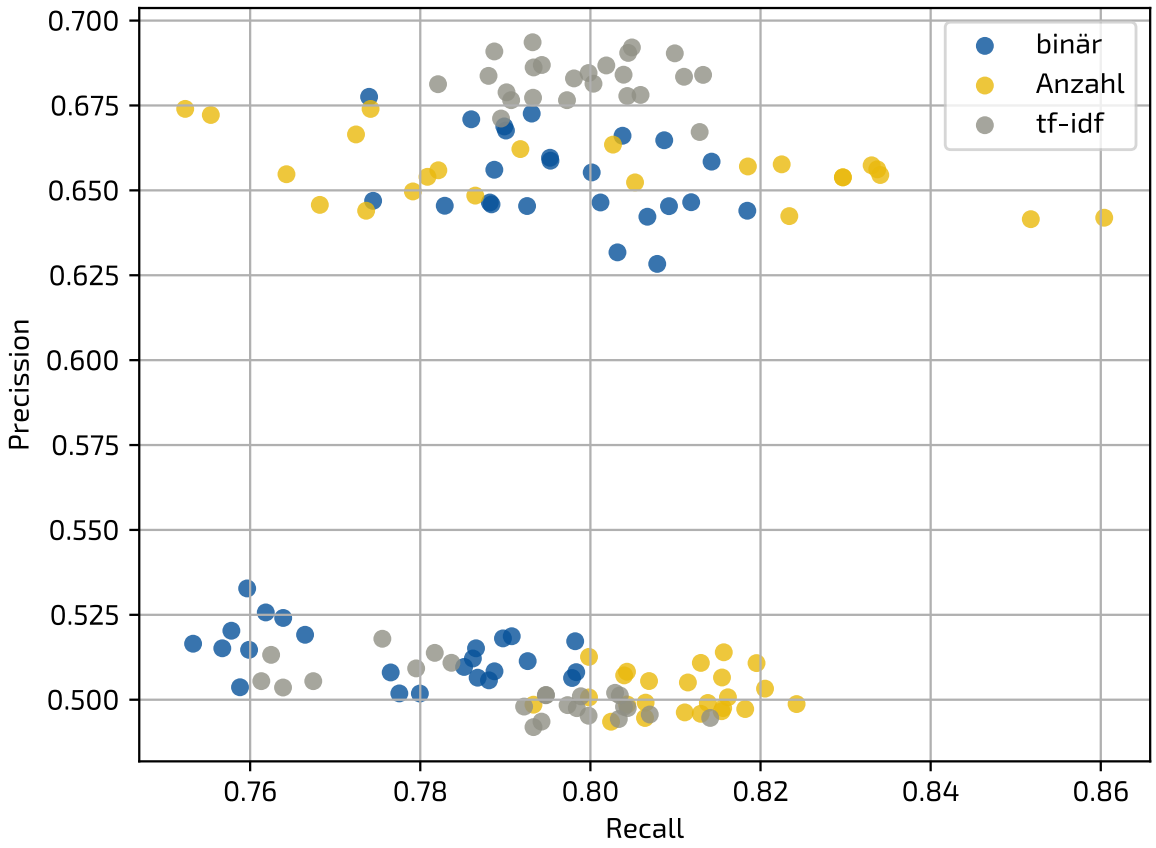


Abbildung 10.: *Recall und Precision nach Feature Gewichtungsverfahren*

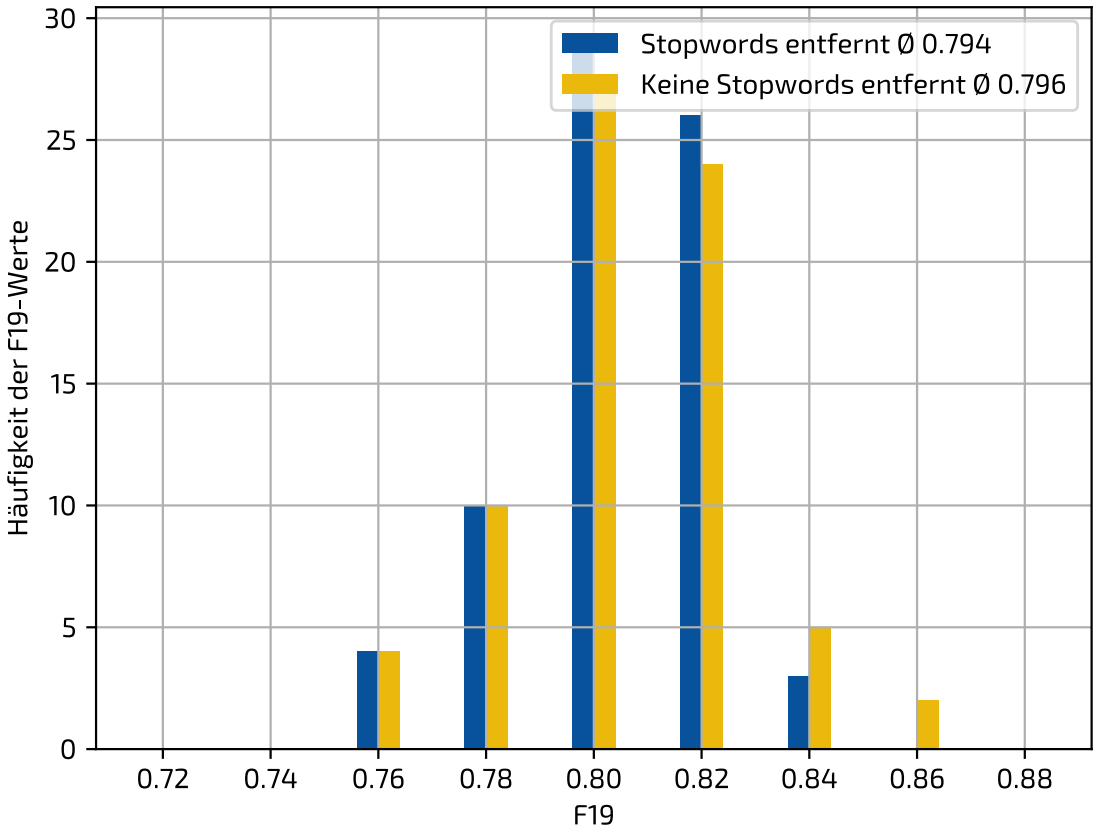
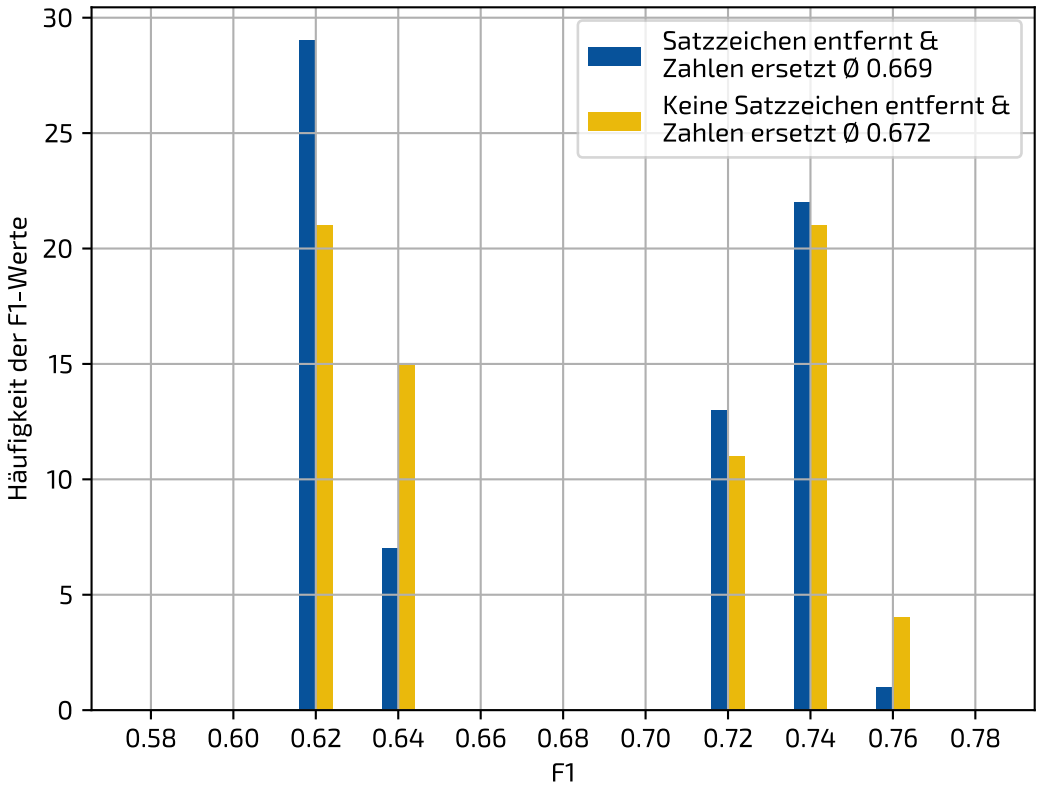
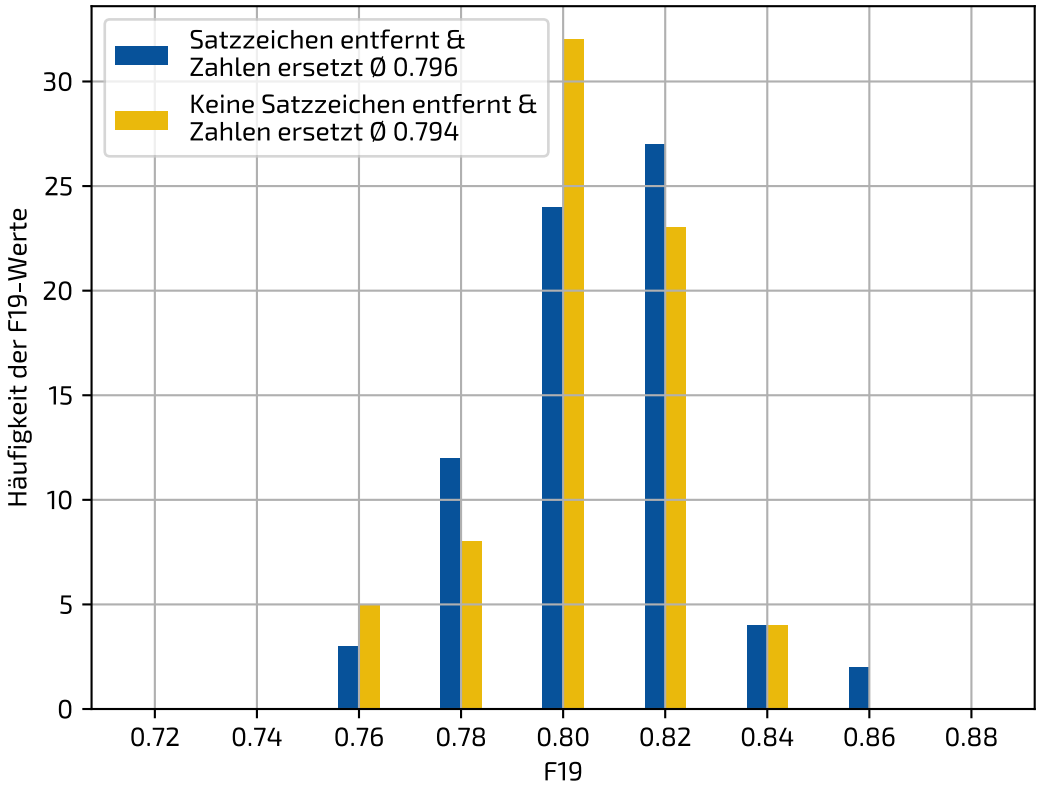


Abbildung 11.: *Recall und Precision getrennt nach Stopword Entfernung*



(A) *Verteilung* F_1



(B) *Verteilung* F_{19}

Abbildung 12.: *Satzzeichen-Entfernung und Zahlen-Ersetzung*

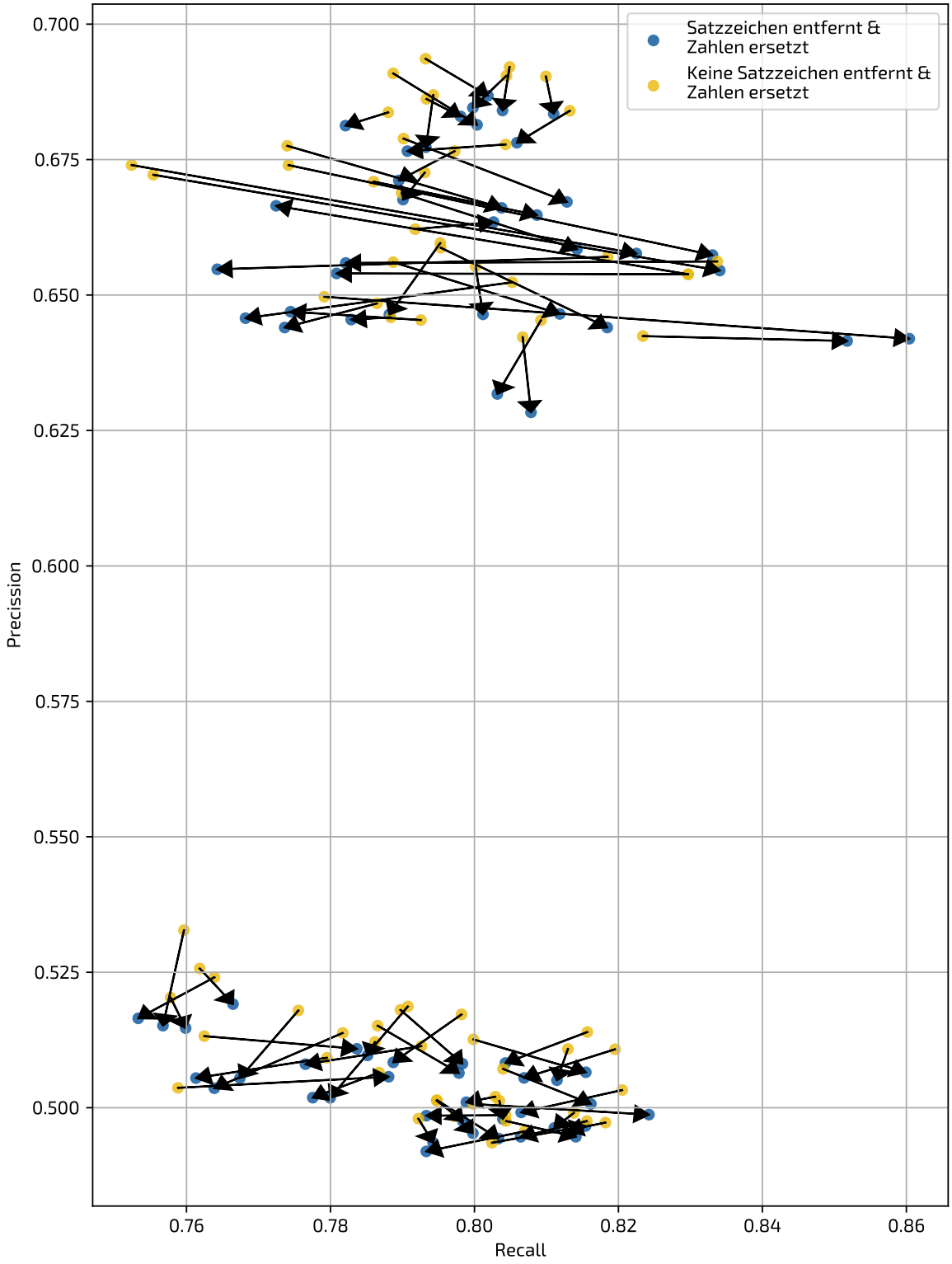


Abbildung 13.: *Veränderung Recall und Precision bei Verwendung von Satzzeichen-Entfernung und Zahlen-Ersetzung*