INSTITUT FÜR INFORMATIK
Datenbanken und Informationssysteme

Universitätsstr. 1 D–40225 Düsseldorf

Heinrich Heine Universität Düsseldorf

# Klassifikation von Tweets zur Erkennung von Trollen

**Robin Kösters**

Bachelorarbeit

| | |
|---|---|
| Beginn der Arbeit: | 18. September 2020 |
| Abgabe der Arbeit: | 18. Dezember 2020 |
| Gutachter: | Prof. Dr. Stefan Conrad |
| | Prof. Dr. Martin Mauve |

## Erklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbstständig verfasst habe. Ich habe dazu keine anderen als die angegebenen Quellen und Hilfsmittel verwendet.

Düsseldorf, den 18. Dezember 2020

Robin Kösters

## Zusammenfassung

Hier kommt eine ca. einseitige Zusammenfassung der Arbeit rein.

# Inhaltsverzeichnis

# 1 Einleitung

## 1.1 Motivation

Die US-Präsidentschaftswahl 2016 gilt vielen politischen Beobachtern als eindrucksvollstes Beispiel dafür, wie von staatlicher Seite organisierte Desinformationskampagnen den politischen Diskurs nachhaltig verzerren können. Bereits zwei Jahre nach der Wahl kamen britische Wissenschaftler in Zusammenarbeit mit der Firma Graphika zu der Erkenntnis, dass ein Unternehmen mit dem Namen „Internet Research Agency“ (IRA), welches dem russischen Staat nahesteht, im großen Stil versucht hat, amerikanische Wähler via Fake-Postings in sozialen Netzwerken zu beeinflussen. In ihrem Bericht geben die Forscher an, dass mehr als 30 Millionen Benutzer im Zeitraum von 2015 bis 2017 in Berührung mit von der IRA erstellten Inhalten gekommen sind. Die Urheber jener destruktiver Inhalte werden Trolle genannt. (Howard et al., 2019)
In Twitter, welches das hauptsächlich betrachtete soziale Netzwerk in dieser Arbeit sein soll, machen Trolle besonders von der Hashtag-Setzung Gebrauch. Bei einem *Hashtag* handelt es sich um eine durch den Ersteller eines Beitrags vorgenommene Themenfestlegung bzw. -klassifizierung. Eine nicht durch Leerzeichen unterbrochene Zeichenkette wird durch Voranstellen eines Rautezeichens (engl. *hash*) zu einem Hashtag. In der Folge kann ein *Tweet* (plattformeigene Bezeichnung für einen Beitrag) durch das Benutzen der allgemeinen Suchfunktion gefunden werden. Abbildung 1 zeigt einen beispielhaften Tweet der IRA, welcher dem der Arbeit zugrundeliegenden Datensatz entnommen wurde und die Hashtag-Nutzung durch einen Troll illustriert.

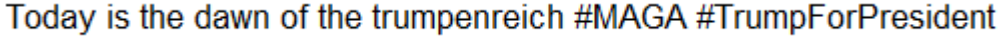


Abbildung 1: Beispiel eines IRA-Tweets

Kommt es innerhalb eines gewissen Zeitraums zur gehäuften Benutzung eines bestimmten Hashtags, so wird dieser in den „Twitter-Trends“, eine Art Ranking der momentan meistgenutzten Hashtags, aufgeführt. Dies macht die Aktualität bzw. Relevanz eines gesellschaftlichen Themas direkt ablesbar. Für Trolle bietet dies aber die Möglichkeit durch geschickte zeitliche Abstimmung bestimmte Hashtags zu Trend-Hashtags zu machen und den Nutzern so aktiv die Relevanz bestimmter Themen vorzutäuschen.
Insgesamt sind mit den genannten Methoden verschiedenste Arten der Manipulation seitens Trollen denkbar. So können Trolle eine politische Partei oder einen Kandidaten unterstützen und versuchen, ihren Mitbewerbern zu schaden. Beispiele dafür gibt es auch in Deutschland. Propagandaforscherin Lisa-Maria Neudert erwähnt in einem Interview mit Deutschlandfunk Kultur, dass das ultrarechte Netzwerk „Reconquista Germanica“ beabsichtigte, „die AfD so gut wie möglich zu verstärken [...] “, und dass seine Trolle die Partei im Endeffekt „größer erscheinen lassen als sie ist“ (Jabs, 2017). Ein weitere tragende Säule von Desinformationskampagnen, welche meist in Verbindung zu Trollen steht, sind die *Fake News*. Hierbei handelt es sich um Falschmeldungen bzw. Nachrichten ohne Wahrheitsgehalt, welche verbreitet werden, um die Öffentlichkeit zu manipulieren. Trolle treten hier in der Regel wechselweise als Urheber und Multiplikatoren solcher Meldungen auf.

Alle bisher beschriebenen Strategien sind gemeinsam als integrative Gesamtstrategie dazu geeignet, um die Ausgänge demokratischer Wahlen zu beeinflussen. Die Initiative „ichbinhier e.V." und das Londoner „Institute for Strategic Dialogue" (ISD) kamen in einer Studie beispielsweise zu der Schlüsselerkenntis, dass rechtsextreme Trollnetzwerke im Bundestagswahlkampf 2017 Urheber einer ausgedehnten und erfolgreichen „pro-AfD-Wahlkampagne" waren (Kreißel et al., 2018). Das Ziel der Desinformationskampagnen, gleich ob aus dem Inland oder Ausland gesteuert, ist die Destabilisierung der demokratischen Institutionen bzw. der Demokratie als Ganzes. Aus diesem Grund möchte ich in dieser Arbeit meinen Teil dazu beitragen, dass eine Lösung für dieses Problem gefunden wird.

## 1.2 Datensätze

Im Rahmen dieser Arbeit soll ein Verfahren entwickelt werden, welches die zuverlässige Erkennung von Troll-Inhalten auf Twitter ermöglicht. Hierbei kommen verschiedenste Techniken der Textklassifikation zum Einsatz. Diese werden auf zwei Datensätze angewandt, auf deren Eigenschaften und Ursprünge an dieser Stelle eingegangen werden soll. Die Grundmenge des ersten Datensatzes ist eine von Linvill und Warren (2018) veröffentlichte Sammlung von rund 3 Millionen Tweets der IRA. Hier wurden die Tweets mit den zehn häufigsten Hashtags (siehe Tabelle 1) entnommen.

| Platz | Hashtag | Platz | Hashtag |
|---|---|---|---|
| 1 | #news | 6 | #topNews |
| 2 | #sports | 7 | #MAGA |
| 3 | #politics | 8 | #BlackLivesMatter |
| 4 | #world | 9 | #health |
| 5 | #local | 10 | #tcot |

Tabelle 1: Hashtags aus Datensatz 1

Beim zweiten Datensatz handelt es sich um im Vorfeld eigens extrahierte Tweets von echten Profilen. Damit thematische Ähnlichkeit besteht wurden nur Tweets, welche die zehn Hashtags aus dem anderen Datensatz enthalten, ausgewählt. Es wurde streng darauf geachtet, dass die Datensätze disjunkt sind. Tabelle 2 zeigt zum Vergleich die wichtigsten Kennzahlen beider Datensätze.

| Eigenschaft | Troll | Nichttroll |
|---|---|---|
| Anzahl Tweets | 303.036 | 324.873 |
| unterschiedliche Autoren | 770 | 68.706 |
| Zeitraum | 01/2015 - 09/2017 | 01/15 - 05/2018 |
| durchschnittliche Länge | 78,64 Zeichen | 122,27 Zeichen |

Tabelle 2: Vergleich der Datensätze

# 2 Grundlagen des Text Mining

Im nachfolgenden Kapitel werden jene Konzepte des *Text Minings* erläutert, welche grundlegend für das Verständnis von Methoden der Textklassifikation sind. Abbildung 2 zeigt die Phasen, die beginnend beim Textkorpus (Sammlung der Ausgangstexte) bis zur abschließenden Einstufung durchlaufen werden. Hieran werde ich mich bei den Ausführungen orientieren.

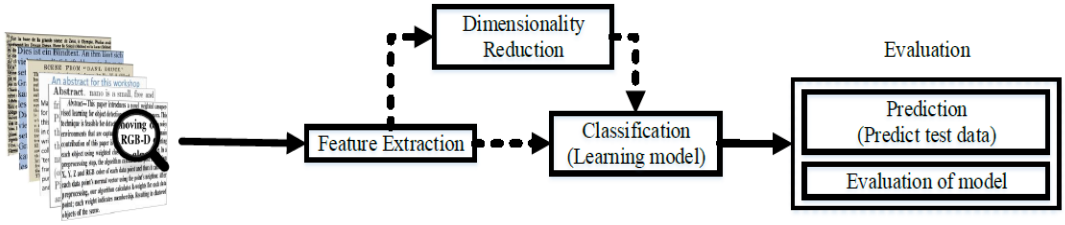


Abbildung 2: Pipeline der Textklassifikation (Kowsari et al., 2019)

## 2.1 Preprocessing

Der rohe, unbehandelte Text wird beim *Preprocessing* (deutsch: Vorbehandlung) in eine Form überführt, die eine nachfolgende Repräsentation der Daten durch Vektoren und ähnliche Methoden ermöglicht. Je nach gewünschter Anwendung sind hier verschiedene Vorverarbeitungsverfahren in Betracht zu ziehen.
Der Begriff Wortsegmentierung bzw. **Tokenisierung** bezeichnet die Zerteilung des Ausgangstexts in kleinere Einheiten, sogenannte *Token* (Manning und Schütze, 1999). In der Regel handelt es sich hierbei um einzelne Wörter, Zahlen und Satzzeichen. Als grundsätzliches Abgrenzungszeichen gelten in den meisten Sprachen die *Whitespace*-Zeichen (Leerzeichen, Tab, Newline-Zeichen). Dies wird auch auf die zu analysierenden Tweets zutreffen, da diese fast ausschließlich in englischer Sprache verfasst sind.

| Er | rief | : | „ | Baum | fällt | ! | “ |
|---|---|---|---|---|---|---|---|

Abbildung 3: Tokenisierung eines Beispielsatzes

Eine weitere Methode der Vorverarbeitung ist das Entfernen von **Stoppwörtern**. Dies sind sehr häufig auftretende Wörter wie „der“, „die“, „das“, „und“ oder „von“, welche vornehmlich eine grammatikalische Funktion und keinen Informationsgehalt haben. (vgl. ebd., S. 533)
Bei der **Lemmatisierung** (Airio, 2006) werden mehrere Wörter unterschiedlicher Erscheinungsform, welche aber die gleiche Bedeutung haben, auf eine gemeinsame Grundform

(genannt Lemma) zurückgeführt. So lassen sich beispielsweise die deklinierten Substantive „Wortes“, „Wörter“, „Wörtern“auf „Wort“ zurückführen, während die Grundform der konjugierten Verben „schrieb“, „schreibe“, „schreibst“ und „schreibt“ der Infinitiv „schreiben“ist. Ein verwandtes Konzept wird **Stemming** (Airio, 2006) genannt. Der Unterschied zur Lemmatisierung besteht darin, dass das daraus resultierende Grundwort (hier: Stamm) kein natürlichsprachliches Wort sein muss, sondern in der Regel ein um Präfix und Suffix beschnittes Wort ist. Ein Beispiel hierfür ist die Reduzierung der Wörter „gehen“, „umgehen“und „zugehen“ auf den Stamm „geh-“.
Für die Lemmatisierung eines Textes wird auch das **Part-of-Speech (POS) Tagging** (Kumawat und Jain, 2015) benötigt. Hierbei wird einem Token seine Wortart (z.B. Nomen, Verb, Adjektiv) oder ein anderes Label wie „Interpunktion“ zugewiesen. Werkzeuge, die dieses Verfahren anwenden, werden POS-*Tagger* genannt.

## 2.2 Feature Extraction

Bevor eine Klassifikation vorgenommen werden kann, müssen die Merkmale aus den vorliegenden Texten gewonnen und mit mathematischen Methoden strukturiert bzw. modelliert werden. Hierbei fällt die Wahl zumeist auf Vektorisierung.
Die einfachste Vektorisierungstechnik ist die **Bag-of-Words** (BoW) (Ramos, 2003). Hier wird ein Text durch einen Vektor mit den Begriffshäufigkeiten (auch: *Term Frequencies* (**TF**)) aller zuvor extrahierten Tokens des Textkorpus repräsentiert. Bei einem Beispielkorpus mit den beiden Texten „my coffee is too hot“ und „my tea is too cold“ und dem zuvor extrahierten Vokabular

{ "my", "coffee", "hot", "tea", "cold"}

ergibt sich die folgende Repräsentation:

$v_1$ = [ 1, 1, 1, 0, 0 ]
$v_2$ = [ 1, 0, 0, 1, 1 ]

Eine Erweiterung dieser TF-Methode ist *Term Frequency-Inverse Document Frequency* (**TF-IDF**) (Ramos, 2003). Hier wird die Vorkommenshäufigkeit anders gewichtet, um dem Aspekt gerecht zu werden, dass einige Begriffe überproportional oft vorkommen. Gleichung 1 zeigt die Gewichtung

$$w(d,t) = TF(d,t) \cdot \log\left(\frac{N}{DF(t)}\right) \tag{1}$$

wobei $N$ die Anzahl der Texte im Korpus und $DF(t)$ die Anzahl der Texte, welche den Begriff $t$ enthalten, ist.
Eine Möglichkeit, Wortkombinationen bzw. bestimmte Formulierungen als Merkmal zu berücksichtigen ist das **N-Gramm** (Kowsari et al., 2019). Dies ist die Zusammenfassung von $N$ aufeinanderfolgenden Token in einem Text. Folglich handelt es sich bei den Elementen einer Bag of Words um 1-Gramme. Das nachfolgende Beispiel zeigt die Repräsentation des Textes „Hier sehen Sie ein Beispiel.“ mit 2-Grammen:

{ “Hier sehen “, “sehen Sie“, “Sie ein“, “ein Beispiel“}

Alle vorausgegangenen Methoden der Merkmalsextraktion haben gemeinsam, dass sie keinen Aufschluss über Zusammenhänge der Wortbedeutungen (Semantik) geben. Dieses Problem versuchen die Techniken der **Worteinbettung** zu lösen. Ein prominentes Beispiel ist **Word2Vec** (Mikolov et al., 2013). Die Grundidee ist hier, dass Begriffe, die eine ähnliche Bedeutung haben, in einem ähnlichen Kontext verwendet werden. Auf dieser Basis wird ein Vektor für jedes Wort im Vokabular durch ein neuronales Netz erzeugt. Die Entfernungen im daraus entstehenden Vektorraum geben dabei semantische Ähnlichkeiten wieder.

## 2.3 Dimensionalitätsreduktion

Bei sehr umfangreichen Datensätzen wie den mehr als 600.000 Tweets in dieser Arbeit werden in der Anwendung der zuvor beschriebenen Verfahren meist hochdimensionale Vektoren erzeugt. In der Folge werden viele Operationen bei späteren Algorithmen der Textklassifikation eine hohe Zeit- und Speicherkomplexität besitzen. Diesem Effekt versucht man im Voraus durch Dimensionalitätsreduktion entgegenzuwirken.
Ein erstes verwendetes Verfahren ist die Hauptkomponentenanalyse bzw. **Principal Component Analysis** (PCA) (Joliffe, 2002). Mit diesem ist es möglich, in der Punktwolke der vorhandenen Vektoren all jene Vektor-Komponenten herauszufinden, die für die größte Varianz verantwortlich sind, also den größten Informationsgehalt haben. Der mathematische Mechanismus dahinter ist die Hauptachsentransformation: Es wird eine Ladungsmatrix aus den Eigenvektoren der Kovarianzmatrix gebildet, aus welcher der Anteil der Varianz jeder Komponente an der Gesamtvarianz ersichtlich ist. In der Folge können Komponenten, welche wenig Varianz beitragen, ohne nennenswerten Informationsverlust verworfen werden.
Eine andere Möglichkeit ist die **Nichtnegative Matrixfaktorisierung** (NMF) (Lee und Seung, 1999). Hier wird aus den $n$ Texten mit insgesamt $m$ Wörtern eine $m \times n$ Matrix gebildet, welche approximativ so faktorisiert wird, dass

$$V \approx WH$$

gilt, wobei W eine $m \times r$ Matrix und H eine $r \times n$ Matrix ist. Die $r$ Spalten von $W$ enthalten semantisch verwandte Wörter, welche zusammen einen Kontext bzw. ein Thema bilden. Voraussetzung für dieses Verfahren ist die Nichtnegativität von $V$.

# 3 Klassifikationsverfahren

Nachdem die Tweets vorbearbeitet, die Merkmale gewonnen und die Dimensionen zwecks verbesserter Performance reduziert wurden, können sie nun als Trainingsdatensatz für ein Klassifikationsverfahren verwendet werden. Im nachfolgenden Kapitel werden verschiedene solcher Verfahren vorgestellt und im Hinblick auf Voraussetzungen, Stärken und Schwächen analysiert. Ein besonderes Augenmerk soll in der Analyse auf jenen Parametern liegen, welche die Ergebnisse beim vorliegenden Datensatz maßgeblich beeinflussen können.

## 3.1 k-Nearest-Neighbor-Algorithmus

Der k-Nearest-Neighbor-Algorithmus (KNN) ist eine parameterfreie Methode der Klassifikation (Guo et al., 2004). Der zu klassifizierende Text wird zunächst analog zu den Trainingstexten vektorisiert. Über ein geeignetes Abstandsmaß werden nun die $k$ räumlich nächsten Nachbarn bestimmt. Der Text wird nun der Klasse zugeordnet, der die Mehrheit der Nachbarn angehören. Abbildung 4 zeigt die k-Nearest-Neighbor-Klassifikation eines Punktes $x$ für $k = 3$. Da die meisten Nachbarn hier dem Troll-Datensatz angehören, würde der zu $x$ gehörige Tweet somit als Troll-Tweet klassifiziert werden.
Die Wahl von $k$ ist entscheidend für die Qualität des Ergebnisses. Entscheidet man sich beispielsweise für ein zu kleines $k$, so ist möglich, dass vereinzelte Ausreißer die Genauigkeit trüben. Ist es auf der anderen Seite zu groß, so werden wahrscheinlich zu weit entfernte Punkte bei der Klassifikation miteinbezogen, was das Ergebnis wiederum verfälschen kann. Ferner ist bei Vorhandensein von 2 Klassen ein ungerades $k$ zu wählen, da andernfalls ein Unentschieden möglich ist.

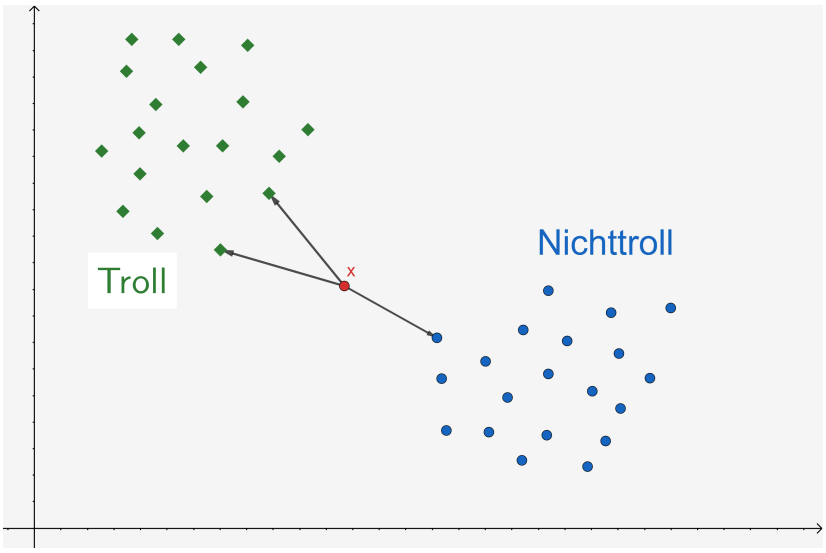


Abbildung 4: KNN-Klassifizierung für $k = 3$

Eine zwingende Voraussetzung, um zuverlässig mit dem KNN-Algorithmus in diesem Projekt arbeiten zu können, ist die Dimensionalitätsreduktion. Die Merkmalsextraktion bringt bei den mehr als 600.000 Tweets hochdimensionale Vektoren hervor. Unbehandelt wären aufgrund der komponentenweisen Abstandsmessung Laufzeiten von einigen Minuten zu erwarten.
Die Vorteile dieses Verfahrens sind die einfache Implementierung und seine Eignung für alle möglichen Ausprägungen von Merkmalsräumen. Als Schwächen werden die bereits angesprochenen Probleme mit der Performance angesehen.

## 3.2 Naiver Bayes-Klassifikator

Der Naive Bayes-Klassifikator (NB) ist ein statistisches Verfahren der Klassifikation (Rish, 2001). Die Grundlage der Berechnung bildet hier der Satz von Bayes, bekannt aus der Wahrscheinlichkeitstheorie. In seiner herkömmlichen Interpretation beschreibt dieser die Berechnung der Wahrscheinlichkeit, dass ein Ereignis dem anderen vorausgegangen ist. Wendet man dies auf einen gegebenen Text $t \in T$ und die Klasse $k_i \in K$ an, so erhält man die Formel für die Wahrscheinlichkeit, dass $t$ der Klasse $k_i$ angehört (siehe Gleichung 2).

$$P(k_i|t) = \frac{P(t|k_i) \cdot P(k_i)}{P(t)} \tag{2}$$

Der Klassifikator bestimmt nun diejenige Klasse $k_i$, für die der Wert dieser Formel maximal ist. Mathematisch formuliert:

$$k = arg \max_{k_i \in K} P(k_i|t) = arg \max_{k_i \in K} \frac{P(t|k_i) \cdot P(k_i)}{P(t)} \tag{3}$$

Die Werte für $P(k_i)$ und $P(t)$ werden a priori über relative Anteilshäufigkeiten, also über den Anteil einer Klasse und den Anteil eines Textes am gesamten Merkmalraum, bestimmt. Für die Maximierung wird meist die Maximum-Likelihood-Methode benutzt.
Jeder Klasse wird vor der Klassifizierung eine bestimmte Form der Wahrscheinlichkeitsverteilung, meist eine Normalverteilung, und die stochastische Unabhängigkeit der Merkmale unterstellt. Letztere Annahme ist „naiv“ weil nicht immer zutreffend, der Algorithmus liefert in der Praxis bei vielen Anwendungen dennoch solide Ergebnisse. Wird diese Annahme hingegen durch stark korrelierte Daten gröber verletzt, so sind Genauigkeitsverluste zu erwarten, was eine Schwäche dieses Verfahrens ist. Im Rahmen der Evaluation werden Ergebnisse unter Annahme einer Normalverteilung und einer Multinomialverteilung verglichen.

### 3.3 Support Vector Machine

Ein anderes Lernmodell trägt den Namen Support Vector Machine (SVM) (Nayak et al., 2015). Die Grundidee ist hier, die vorliegenden Klassen im multidimensionalen Raum durch Einschub einer Hyperebene räumlich zu trennen. Die Ebene wird dabei so platziert, dass der Abstand zu den nächstliegenden Vektoren der beiden Klassen, den sogenannten Stützvektoren (engl. *support vectors*) maximal ist. Abbildung 5 zeigt beispielhaft zwei solcher durch eine Hyperebene (rote Gerade) getrennte Klassen in einem zweidimensionalen Merkmalraum.

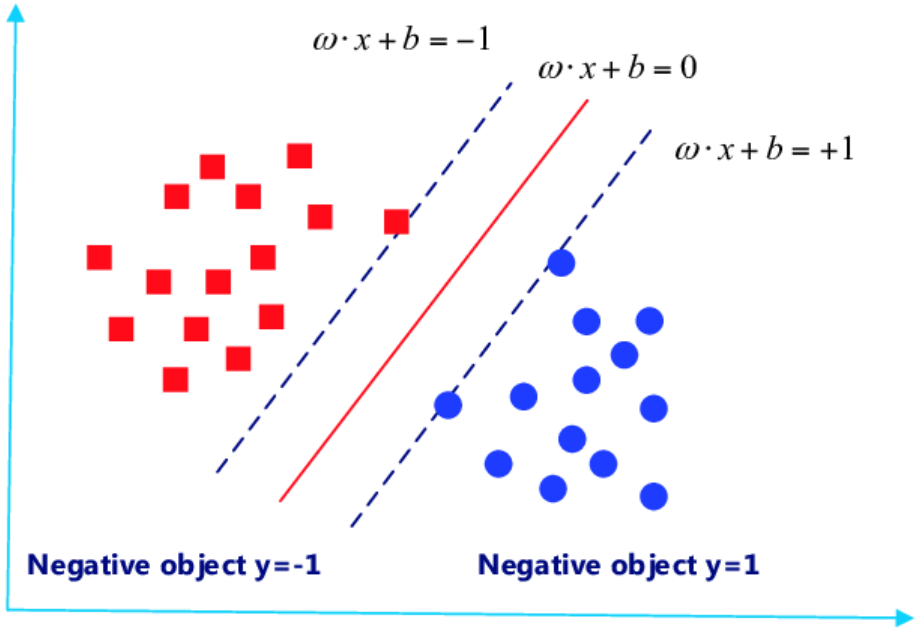


Abbildung 5: SVM-Klassifizierung (Zhou et al., 2016)

Für ein zu klassifizierendes Objekt wird nun mithilfe einer Entscheidungsfunktion festgestellt, auf welcher Seite der Ebene es liegt und kann so eindeutig einer Klasse zugeordnet werden. Ist der Einschub einer Hyperebene nicht möglich, d.h. sind die Klassen nicht linear trennbar, so behilft man sich mit dem sogenannten „Kernel-Trick“: Der Merkmalraum wird auf einen höherdimensionalen Raum abgebildet, in welchem die Klassen dann linear trennbar sind. Diese Transformation benötigt eine hohe Rechenleistung und stellt deshalb bei stark überlappenden Klassen eine Schwäche dieses Verfahrens dar. Bei weniger überlappenden oder linear trennbaren Klassen ist der Algorithmus in der Einstufung jedoch sehr schnell. Ein weiterer Vorteil ist der relativ geringe Speicherverbrauch, da zur Klassifikation nur ein Teil der Trainingsdaten gebraucht wird.
Zwei Parameter, die die Ergebnisse beeinflussen können, sind zum einen der Bestrafungsparameter $C$ und der Gamma-Wert. $C$ gibt an, wie stark die Gefahr der Falschklassifizierung berücksichtigt werden soll. Je niedriger der Wert, desto leichter wird es, eine Trennung zu vollziehen und dabei einzelne Ausreißer zu ignorieren. Allerdings findet dann bei mäßiger bis starker Überlappung keine eindeutige Trennung der Klassen statt.

Je höher der Wert, desto strenger versucht der Algorithmus, alle Punkte in der Entscheidung miteinzubeziehen. Hier besteht die Gefahr der Überanpassung. Der Gamma-Wert gibt an, wie stark einzelne Punkte beim Kernel-Trick Einfluss auf die Gruppierung haben, also wie granular die Gruppen sind. Bei linear trennbaren Klassen kommt er demnach nicht zur Anwendung. Um zu guten Ergebnissen zu kommen, ist eine gute Balance der beiden Werte, in jedem Fall aber die Anpassung an den Datensatz von Nöten.

## 3.4 Entscheidungsbäume

Die Grundlage für ein weiteres Klassifikationsverfahren sind die sogenannten Entscheidungsbäume (engl. *decision trees*) (Quelle). Hierbei handelt es sich um herkömmliche, aus der Graphentheorie bekannte Bäume, welche eine Entscheidungsfolge abbilden. Für jedes zu klassifizierende Objekt wird der Baum einmal durchlaufen. An jedem inneren Knoten wird nun der Wert eines extrahierten Merkmals abgefragt. Je nach Ergebnis des Vergleichs wird entschieden, welches Kind als nächstes besucht wird. Die Blätter beinhalten die möglichen Klassen, sodass nach der letzten Entscheidung mit Ankommen am Ende des Baumes die Klassifikation vollzogen ist. Abbildung 6 zeigt beispielhaft einen Entscheidungsbaum für die Klassifikation eines Messestandorts anhand von drei Merkmalen. Die möglichen Klassen für einen Standort sind „geeignet“ und „ungeeignet“.

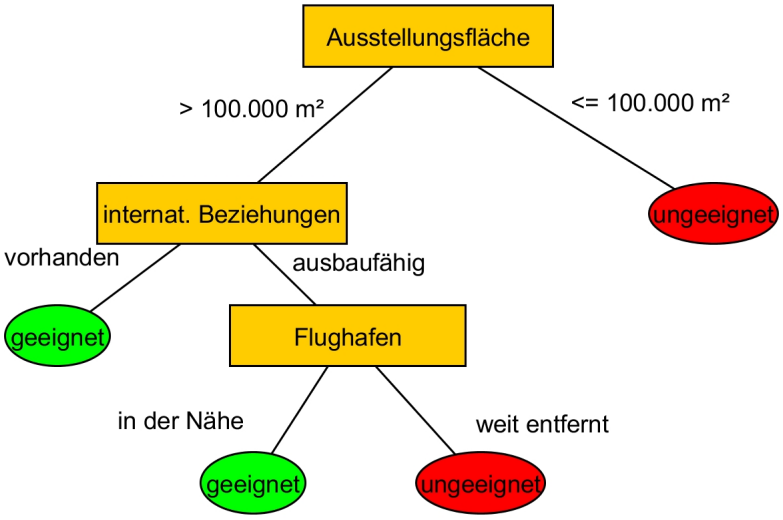


Abbildung 6: Entscheidungsbaum zur Klassifizierung eines Messestandorts

Das Training bzw. das Lernen besteht bei diesem Verfahren im Aufbau des Baumes. Beginnend bei der Wurzel wird nun dasjenige Merkmal bestimmt, welches den größten Fortschritt in der Klassifikationsentscheidung bringt. Ein gebräuchliches Maß dafür ist die *Gini Impurity*, welche die Homogenität bzw. Heterogenität der Daten anzeigt. Gleichsam anwendbar ist hier auch die Entropie, welche den mittleren Informationsgehalt des Merkmalvektors misst. Das gefundene Merkmal dient nun zur Aufteilung der ursprüng-

lichen Datenmenge. Auf die neu entstehenden Mengen wird diese Methode nun rekursiv angewendet bis Mengen erzeugt werden, deren Elemente alle der gleichen Klasse angehören.
Durch die klare Strukturierung und grafische Darstellbarkeit ist die Entscheidungsbaum-Klassifikation als Konzept leicht verständlich. Ein weiterer Vorteil ist eine automatische Merkmalfilterung beim Aufbau des Baums: Merkmale, die durch Gini Impurity bzw. Entropie als irrelevant erkannt wurden, können leicht weggelassen werden. Hier sind Zeit- und Speichergewinne möglich.
Eine Schwäche dieses Verfahrens ist die schwache Robustheit der Entscheidungsbäume. Dies bedeutet, dass kleine Änderungen im Ausgangsdatensatz zu sehr großen Veränderungen in der Datenstruktur und der anschließenden Klassifikation führen können.

## 3.5 Mehrschichtiges Perzeptron

Das mehrschichtige Perzeptron (engl. *multi-layer perceptron, MLP*) ist ein Vertreter der künstlichen neuronalen Netze (Quelle). In seinem grundlegenden Aufbau ähnelt es einem Logikgatter. Die grundlegenden Bausteine, analog zu Gehirnzellen Neuronen genannt, sind in Schichten angeordnet. Die Eingabeschicht nimmt den Merkmalvektor komponentenweise entgegen, während die Neuronen der Ausgabeschicht das Klassifikationsergebnis ausgeben. Die dazwischenliegenden Schichten werden versteckte Schichten bzw. *hidden layers* genannt. Abbildung 7 illustiert ein Beispiel für Netz, welches einen vierdimensionalen Vektor verarbeitet.

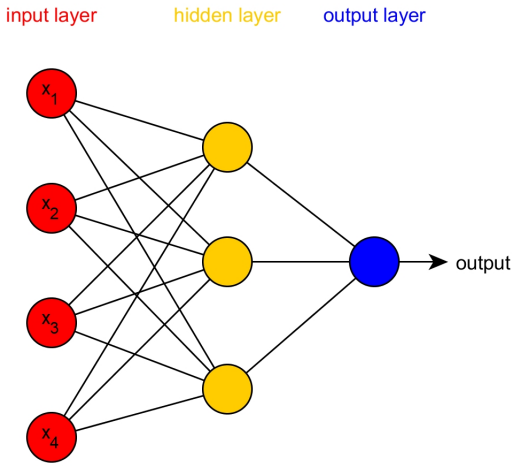


Abbildung 7: Beispiel für mehrschichtiges Perzeptron

Jedes Neuron einer Schicht hat eine Verbindung, ähnlich einer Kante, zu einem Neuron (oft auch zu allen) der nächsten Schicht. Die Verbindung trägt ein Gewicht. Dazu trägt auch jedes Neuron einer versteckten Schicht und der Output-Schicht ein sogenanntes Verzerrungsgewicht bzw. *bias weight* $b$.
Der Lernprozess läuft nun wie folgt ab: Beginnend bei den Input-Neuronen wird für jede

Eingabe eine gegebene Aktivierungsfunktion berechnet. Das Ergebnis wird an die nachfolgenden Neuronen weitergeschickt und erneut die Aktivierungsfunktion berechnet. Ist es schließlich in der Output-Schicht angekommen, so wird das Ergebnis interpretiert und mit der vorher deklarierten Klasse verglichen. Ist das Ergebnis falsch, so werden die Gewichte des Netzes von hinten nach vorne neujustiert. Man spricht hier von *Backpropagation*. Auf diese Weise lernt das Netz mit jedem Fehler dazu und die Klassifikation wird zunehmend genauer.
Als Aktivierungsfunktion werden klassischerweise drei verschiedene Funktionen herangezogen. Die erste Funktion ist der *Tangens hyperbolicus* (kurz tanh).

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4}$$

Oft verwendet wird auch die folgende logistische Funktion

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

Eine neuere Form der Aktivierungsfunktion ist die *rectified linear unit* (kurz: ReLu). Sie ist folgendermaßen definiert:

$$f(x) = \max(0, x) \tag{6}$$

# 4 Gütekriterien

Die bereits besprochenen Klassifikationsverfahren werden nach dem „Train-and-Test"-Prinzip auf den vorliegenden Datensatz angewandt. Die Grundmenge der Tweets wird hierbei in eine Trainings- und eine Testmenge aufgeteilt. Die Trainingsmenge dient dem jeweiligen Verfahren als Lerneingabe, während die Testmenge zur Beurteilung der Performance herangezogen wird. Nach der abgeschlossenen Testklassifikation liegen Ergebnisse in Form der den Tweets zugeordneten Labels (Troll oder Nichttroll) vor. An diesem Punkt muss nun geklärt werden, in welcher Weise eine Auswertung vorgenommen werden kann. Wie kann die Qualität eines binären Klassifikators gemessen werden? Was für eine Aussagekraft haben verschiedene Gütekriterien? Welche Nachteile gibt es? Welche Kriterien zeigen die Stärken, welche die Schwächen eines Verfahrens auf? Diese Fragen sollen in diesem Kapitel besprochen werden, ehe die Ergebnisse evaluiert werden können. Neben Qualität im Sinne von korrekt oder falsch soll es hier auch um Qualität im Sinne von schnell (Zeitkomplexität) gehen.

## 4.1 Konfusionsmatrix

Da bei den Tweets der Testmenge im Voraus die richtige Klasse bekannt war, kann für jedes klassifizierte Element eine Korrekt- oder Falschklassifizierung festgestellt werden. Typischerweise wird bei einer binären Klassifikation eine der beiden Klassen als positiv (im Sinne von „bestätigend") und die andere als negativ aufgefasst. Der gegebene Zweck der Arbeit (Trollerkennung) und die Semantik der beiden Labelbezeichnungen legen in diesem Fall nahe, dass „Troll" als positiv („Troll gefunden") und „Nichttroll" als negativ („keinen Troll gefunden") aufgefasst wird. Kombiniert ergeben sich hieraus nun folgende Kennzahlen:

- $r_p$ ist die Anzahl richtig positiver, $r_n$ die Anzahl richtig negativer Klassifizierungen.
- $f_p$ bezeichnet die Anzahl falsch positiver, $f_n$ die Zahl falsch negativer Klassifizierungen.

Eine häufige Darstellung von Korrekt- und Falschklassifikation ist die sogenannte Konfusionsmatrix oder Wahrheitsmatrix (siehe Tabelle 3), in welcher die genannten Kennwerte direkt ablesbar sind.

| | | korrekte Klasse | |
|---|---|---|---|
| | | Troll ($r_p + f_n$) | Nichttroll ($f_p + r_n$) |
| Test-ergebnis | Troll ($r_p + f_p$) | $r_p$ | $f_p$ |
| | Nichttroll ($f_n + r_n$) | $f_n$ | $r_n$ |

Tabelle 3: Konfusionsmatrix der Trollerkennung

Die genannte Werte bilden die Berechnungsgrundlage aller nachfolgend genannten Gütemaße. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet.

## 4.2 Treffergenauigkeit

Die Treffergenauigkeit (engl. **accuracy**) ist das am häufigsten für die Auswertung von Klassifikationsergebnissen gebrauchte Gütemaß. Sie ist folgendermaßen definiert:

$$accuracy = \frac{r_p + r_n}{r_p + f_p + r_n + f_n} = \frac{r_p + r_n}{N} \tag{7}$$

Wie Formel 7 zu entnehmen ist, handelt es sich hierbei um den Anteil der richtig klassifizierten Objekte an der Gesamtzahl $N$ aller Objekte. Deshalb nennt man dieses Maß auch Korrektklassifizierungsrate.
Die Treffergenauigkeit ist von allen Maßen in der Gesamtschau am intuitivsten, da sie die Verlässlichkeit der Klassifikation, also die Wahrscheinlichkeit, dass die getroffene Vorhersage korrekt ist, misst. Für die Trollerkennung ist dies von besonderer Bedeutung: Sowohl das Nichterkennen eines Trolls als auch die „Falschbeschuldigung“eines echten Benutzers können kritische Konsequenzen nach sich ziehen. Aus diesem Grund ist es wichtig zu wissen, mit welcher Wahrscheinlichkeit eine Verwechslung auszuschließen ist.

## 4.3 Sensitivität und Spezifität

Die **Sensitivität** (engl. *recall*) ist ähnlich definiert wie die Treffergenauigkeit, mit dem Unterschied, dass hier die positiven Werte (Trollmenge) isoliert betrachtet werden. Man könnte sie aus diesem Grund auch als „Troll-Korrektklassifizierungsrate“ bezeichnen. Sie beschreibt demnach den Anteil der richtig erkannten Troll-Tweets an der Gesamtzahl aller Troll-Tweets.

$$recall = \frac{r_p}{r_p + f_n} \tag{8}$$

Analog dazu ist die **Spezifität** (engl. *specifity*) definiert, sie gibt den Anteil der korrekt als Nichtroll-Tweets erkannten Texte an der Gesamtzahl aller Nichttroll-Tweets an.

$$specifity = \frac{r_n}{r_n + f_p} \tag{9}$$

Die Betrachtung dieser Werte ist vor allem dann von Vorteil, wenn sie sich merkbar voneinander unterscheiden. In diesem Fall liefert die Treffergenauigkeit allein nicht genügend Aufschluss über die Korrektklassifizierung von Trollen und Nichtrollen und es lohnt sich, diese Einzelbetrachtung vorzunehmen.
Auch wenn eine Verwechslung, wie bereits ausgeführt, in jedem Fall problematisch ist, so ist die Falschklassifizierung eines realen Users besonders verheerend, da die getroffenen Gegenmaßnahmen (i.d.R. Sperrungen) unrechtmäßig wären. Aus diesem Grund ist die Spezifität für die Trollerkennung, sofern damit Gegenmaßnahmen einhergehen, von besonderer Bedeutung. Da Trolle speziell in großer Gruppenzahl Wirksamkeit erreichen, ist das fälschliche „Freisprechen“ einiger Trolle weniger kritisch, solange der Rest überwiegend korrekt klassifiziert wird. Dies macht die Sensitivität zweitrangig, aber dennoch wichtig. Diese Art der Klassifikation, bei der $f_p$ besonders niedrig gehalten werden soll, nennt man **konservative** Klassifikation.

## 4.4 Relevanz und Segreganz

Die **Relevanz** (engl. *precision*) verfolgt im Vergleich zur Sensitivität einen umgekehrten Ansatz. Während letztere die Frage „Wieviel Prozent der Trolle werden richtig erkannt?“ beantwortet, geht es hier um die Behandlung der Frage „Wieviel Prozent der Troll-Vorhersagen stimmen?“. Definiert ist die Relevanz als Anteil der richtig positiven Vorhersagen an der Gesamtzahl aller positiven Vorhersagen.

$$precision = \frac{r_p}{r_p + f_p} \tag{10}$$

Das negative Äquivalent zur Relevanz ist die **Segreganz** (engl. *negative predictive value*, NPV). Sie beschreibt das Verhältnis von den korrekt negativen Vorhersagen an der Gesamtzahl aller negativen Vorhersagen.

$$NPV = \frac{r_n}{r_n + f_n} \tag{11}$$

Der Fokus bei diesen Maßen liegt nicht auf der Abdeckung der jeweiligen Klassen, sondern auf der Vorhersagegenauigkeit. Es gelten hier ähnliche Implikationen im Bezug auf sich ergebende Probleme wie bei den zuletztgenannten Maßen.

## 4.5 F-Score

Der **F-Score** kombiniert die bereits beschriebenen Gütemaße Sensitivität und Relevanz und setzt sie mittels des harmonischen Mittels zueinander in Beziehung.

$$F_\alpha = (1 + \alpha^2) \cdot \frac{precision \cdot recall}{\alpha^2 \cdot precision + recall} \tag{12}$$

Formel 12 zeigt die Formel für ein allgemeines $\alpha$, welches die gewählte Gewichtung beschreibt. Die am häufigsten verwendete Gewichtung ist $\alpha = 1$:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{13}$$

# 5 Evaluation

Im nachfolgenden Kapitel sollen nun die Ergebnisse der praktischen Anwendung der Klassifikationstechniken auf die vorliegenden Datensätze ausgewertet werden. Wie in den vorangegangenen Kapiteln besprochen, wird erwartet, dass verschiedene Techniken der Merkmalsextraktion, verschiedene Level der Dimensionalitätsreduktion und verschiedene Hyperparametereinstellungen die Ergebnisse qualitativ beeinflussen werden. Eine Methode, die es ermöglicht, diese Unterschiede zu messen und gleichzeitig die beste Kombination zu bestimmen, ist die sogenannte Hyperparameteroptimierung. Hierbei ...

## 5.1 Implementierung

Zwecks Nachvollziehbarkeit und Transparenz soll an dieser Stelle kurz auf die parallel erfolgte praktische Anwendung der beschriebenen Techniken eingegangen werden.
Im Rahmen dieser Abschlussarbeit wurde ein Kommandozeilen-Tool namens „TrollDetector“ [1] in der Programmiersprache Python entwickelt. Dieses hat mehrere Funktionen. Zum einen ist damit möglich, ein beliebiges Klassifikationsverfahren mit selbst gewählten Hyperparametern auf den Datensatz anzuwenden. Dies geschieht nach dem bereits erwähnten „Train-and-Test“-Verfahren. Hier sind auch Merkmalsextraktion (z.B. TF vs. TF-IDF) und Dimensionalitätsreduktion direkt steuerbar.
Eine weitere Funktion ist die Hyperparameteroptimierung für jedes Verfahren mit anschließender Auswertung der Ergebnisse. Für die Optimierung wird eine Rastersuche verwendet. Bei dieser Methode wird das jeweilige Klassifikationsverfahren auf einer fest definierten Untermenge aller möglichen Kombinationen von Hyperparametern ausgeführt. Anschließend werden die Ergebnisse der Kombinationen ausgewertet.
Schließlich ist es mit dem Programm noch möglich, einen Vergleich der fünf Klassifikatoren anzustellen. Die voreingestellten Hyperparameter sind jene, welche bei der Hyperparameteroptimierung am besten abgeschnitten haben, sodass eine Vergleichbarkeit hergestellt wird.
Zur Verarbeitung und Klassifikation des Datensatzes wurden ausschließlich Klassen und Funktionen aus der „scikit-learn“-Bibliothek von Pedregosa et al. (2011) verwendet.

[1] https://github.com/rokoe102/trolldetector

## 5.2 Hyperparameteroptimierungen

Zu Beginn soll jedes Verfahren einzeln auf die Eignung als Trollerkennungsinstrument untersucht werden. Hierzu werden die Ergebnisse der Hyperparameteroptimierungen diskutiert. Folgende Hyperparameter teilen sich dabei alle Verfahren, da die Merkmalsextraktion bei ihnen gleich abläuft:

| Hyperparameter | gewählte Werte |
|---|---|
| Merkmalgewichtung | TF, TF-IDF |
| Stoppwort-Filterung | keine, englisch |
| n-Gramm-Extraktion | 1-Gramme, 1+2-Gramme |

Tabelle 4: gemeinsame Hyperparameter aller Verfahren

Alle anderen Parameter sind verfahrensspezifisch.

### 5.2.1 k-Nearest-Neighbor-Algorithmus

Beim KNN-Algorithmus sind neben den Hyperparametern, welche mit anderen Verfahren geteilt werden, die Abstandsmetrik und der k-Wert für die zu untersuchenden Nachbarn gegeben.
Tabelle 5 zeigt die mittleren Ergebnisse verschiedener Ausprägungen aller Hyperparameter einerseits im Vergleich untereinander und im Vergleich zur durchschnittlichen und zur besten Performance. Hieraus lassen sich verschiedene Schlussfolgerungen ziehen:

| Hyperparameter | Genauigkeit | Relevanz | Segreganz | Sensitivität | Spezifität | $F_1$ |
|---|---|---|---|---|---|---|
| TF | **0.951** | **0.937** | 0.965 | 0.964 | **0.937** | **0.949** |
| TF-IDF | 0.949 | 0.932 | **0.966** | 0.964 | 0.935 | 0.948 |
| engl. Stoppwörter | 0.947 | 0.931 | 0.962 | 0.961 | 0.934 | 0.946 |
| keine Filterung | **0.953** | **0.937** | **0.968** | **0.967** | **0.940** | **0.952** |
| 1-Gramme | **0.955** | **0.941** | **0.969** | **0.967** | **0.967** | **0.954** |
| 1+2-Gramme | 0.945 | 0.928 | 0.962 | 0.961 | 0.961 | 0.944 |
| $k = 5$ | **0.953** | **0.938** | **0.967** | **0.966** | **0.940** | **0.952** |
| $k = 15$ | 0.950 | 0.934 | 0.965 | 0.964 | 0.936 | 0.949 |
| $k = 25$ | 0.948 | 0.931 | 0.964 | 0.962 | 0.934 | 0.947 |
| euklid. Metr. | 0.950 | **0.935** | 0.965 | 0.963 | 0.937 | 0.949 |
| Manhattan | 0.950 | 0.934 | **0.966** | **0.964** | 0.937 | 0.949 |
| Maximum | 0.960 | 0.948 | 0.973 | 0.972 | 0.950 | 0.959 |
| durchschnittl. | 0.950 | 0.934 | 0.965 | 0.964 | 0.937 | 0.949 |

Tabelle 5: Ergebnisse bei KNN

Insgesamt sind beim KNN-Algorithmus ausgezeichnete Zahlen zu beobachten. Im Maximum liegen nahezu alle Kennzahlen über 95%, positiv auffallend sind hier Werte der Segreganz und der Sensitivität von über 97%. Durchschnittlich werden immer Punktzahlen von über 93% erreicht. Ein Unterschied im Abschneiden unterschiedlicher Parametereinstellungen ist bei der Merkmalsgewichtung zu erkennen: TF erreicht hier ein wenig bessere Punktzahlen als TF-IDF. Die Filterung von englischen Stoppwörtern bringt im Mittel keine Verbesserung hervor. Extrahiert man nur 1-Gramme, anstatt 1-Gramme und 2-Gramme, erreicht man bis zu 1% höhere Punktzahlen.
Bei den verfahrensspezifischen Hyperparametern gibt es nur wenige Schwankungen. Beim k-Wert deutet sich an, dass ein einstelliger Wert besser abschneidet als ein zweistelliger, die Verbesserungen belaufen sich aber nur auf höchstens 0,5%. Bei den Metriken ist auffällig, dass unterschiedliche Arten der Abstandsmessungen nahezu keinen Unterschied in der Qualität hervorbringen, die Abweichungen betragen hier höchstens 0,1%.
Angesichts seiner durchweg hohen Punktzahlen in allen möglichen Gütemaßen ist der k-Nearest-Neighbor-Algorithmus sehr gut für die Erkennung von IRA-ähnlichen Trollen geeignet.

### 5.2.2 Naiver Bayes-Klassifikator

Der einzige verfahrensspezifische Hyperparameter des Naiven Bayes-Klassifikators ist die angenommene Verteilung der Merkmalvektoren. Getestet wurde mit einer Normalverteilung, einer Multinomialverteilung und einer komplementären Verteilung nach >Rennie et al. (2003)<.
Die Ergebnisse in Tabelle 6 lassen folgende Schlüsse zu:
Dieses Verfahren schneidet je nach Parameter-Einstellung sehr unterschiedlich ab. So liefert die Annahme einer Normalverteilung oder einer komplementären Verteilung Punktzahlen von 75 - 85%.

| Hyperparameter | Genauigkeit | Relevanz | Segreganz | Sensitivität | Spezifität | $F_1$ |
|---|---|---|---|---|---|---|
| TF | **0.747** | 0.772 | **0.767** | **0.685** | 0.804 | **0.694** |
| TF-IDF | 0.702 | **0.798** | 0.736 | 0.571 | **0.824** | 0.536 |
| engl. Stoppwörter | 0.715 | 0.751 | 0.731 | 0.620 | 0.803 | **0.616** |
| keine Filterung | **0.734** | **0.819** | **0.77**1 | **0.636** | **0.825** | 0.614 |
| 1-Gramme | 0.735 | 0.811 | 0.761 | 0.642 | 0.642 | 0.631 |
| 1+2-Gramme | 0.713 | 0.758 | 0.741 | 0.613 | 0.613 | 0.599 |
| Normal | 0.807 | 0.762 | 0.868 | 0.878 | 0.741 | 0.815 |
| Multinomial | 0.585 | 0.841 | 0.568 | 0.198 | 0.947 | 0.253 |
| Komplementär | 0.781 | 0.752 | 0.818 | 0.808 | 0.755 | 0.777 |
| Maximum | 0.834 | 1.000 | 0.949 | 0.958 | 1.000 | 0.855 |
| durchschnittl. | 0.724 | 0.785 | 0.751 | 0.628 | 0.814 | 0.615 |

Tabelle 6: Ergebnisse bei NB

Bei einer Multinomialverteilung sind die Ergebnisse im Durchschnitt deutlich schlechter. Beispielsweise ist die mittlere Treffergenauigkeit 58,5% und die mittlere Sensitivität bei nur 19,8%. Es gibt hierbei sehr starke, gleichzeitige Ausreißer nach oben und unten: In Kombination mit TF-IDF wird eine Spezifität von 100% bei einer Sensitivität von 0% erreicht, weshalb die Treffergenauigkeit hier etwa 50% beträgt.
Bei der Merkmalgewichtung schneiden TF und TF-IDF in einer Hälfte der Gütemaße besser ab. Das Unterlassung einer Filterung von englischen Stoppwörtern führt auch bei diesem Verfahren zu leicht besseren Ergebnissen von 2 - 6%.
Lässt man die Annahme einer Multinomialverteilung außen vor, so werden mit dieser Methode Punktzahlen von durchschnittlich 75% - 85% erreicht, was gerade mit Einstellung der richtigen Hyperparameter zu relativ verlässlichen Ergebnissen führt. Unter diesen Voraussetzungen ist das Verfahren zur Trollerkennung durchaus geeignet.

### 5.2.3 Support Vector Machine

Die Klassifikation mit einer Support Vector Machine hat wie in Kapitel 3.3 beschrieben den Bestrafungsparameter C als einzigen verfahrensspezifischen Hyperparameter. Bei Ausführung der Optimierung kommen folgende Ergebnisse (Tabelle 7) zustande:
Insgesamt liefert die SVM im Mittel Punktzahlen von mindestens 80% in allen betrachteten Gütemaßen. Beste Werte sind Segreganz und Sensitivität: Durchschnittlich werden hier ca. 93% und im Maximum 97% erreicht.
Unterschiedliche Merkmalgewichtungen führen auch hier zu unterschiedlichen Ergebnissen. Sowohl TF als auch TF-IDF schneiden in drei von sechs Gütemaßen besser ab als das jeweils andere. Der stärkste Unterschied betrifft Segreganz und Sensitivität: Hier schneidet TF 6% besser ab. Bei allen anderen Arten der Merkmalsextraktion gibt es nur wenige Schwankungen von höchstens 2%.
Beim Bestrafungsparameter C sind bis auf 0,1% keine Schwankungen in den unterschiedlichen Ausprägungen auszumachen.

| Hyperparameter | Genauigkeit | Relevanz | Segreganz | Sensitivität | Spezifität | $F_1$ |
|---|---|---|---|---|---|---|
| TF | **0.875** | 0.813 | **0.959** | **0.964** | 0.793 | 0.882 |
| TF-IDF | 0.868 | **0.838** | 0.900 | 0.900 | **0.837** | **0.868** |
| engl. Stoppwörter | 0.865 | 0.816 | 0.929 | 0.931 | 0.804 | 0.869 |
| keine Filterung | **0.878** | **0.834** | **0.930** | **0.933** | **0.826** | **0.880** |
| 1-Gramme | 0.870 | 0.820 | **0.932** | **0.935** | **0.935** | 0.874 |
| 1+2-Gramme | **0.873** | **0.830** | 0.928 | 0.929 | 0.929 | **0.876** |
| $C = 1.00$ | 0.872 | 0.826 | 0.930 | 0.932 | 0.815 | 0.875 |
| $C = 0.75$ | 0.871 | 0.825 | 0.929 | 0.932 | 0.815 | 0.875 |
| $C = 0.50$ | 0.871 | 0.825 | 0.930 | 0.932 | 0.814 | 0.875 |
| Maximum | 0.892 | 0.867 | 0.970 | 0.974 | 0869 | 0.892 |
| durchschnittl. | 0.871 | 0.825 | 0.930 | 0.932 | 0.815 | 0.875 |

Tabelle 7: Ergebnisse bei SVM

Insgesamt ist die SVM-Klassifikation mit den erreichten Punktzahlen ziemlich verlässlich und daher für die Trollerkennung gut geeignet. Von Vorteil ist auch, dass die durchweg hohe Segreganz für ein niedriges $f_n$ spricht, was eine konservative Klassifikation, welche für die Trollerkennung wichtig ist, gewährleistet. Eine positive Eigenschaft ist außerdem die Stabilität des Verfahrens: Selbst mit wenigen bewussten oder zufälligen Einstellungen liefert das Verfahren immer noch gute Ergebnisse, was für eine Art Benutzerfreundlichkeit spricht.

### 5.2.4 Entscheidungsbäume

Bei der Entscheidungsbaum-Klassifikation sind neben den von allen geteilten Hyperparametern auch das Aufteilungskriterium gegeben. Die zwei Möglichkeiten sind hier die in Kapitel 3.4 beschriebenen Maße Gini Impurity und Entropie. Aus Tabelle 8 lassen sich folgende Schlüsse ziehen:
Die Klassifikation erreicht im Mittel in allen Gütekriterien sehr hohe Werte von 94%. Die maximalen Werte liegen mit ca. 95% nur 1% darüber. Es lässt sich aus diesem Grund leicht erkennen, dass hier nur sehr geringe Schwankungen vorhanden sind. Dies ist auch im Vergleich unter den Hyperparametern zu erkennen: Zwischen TF und TF-IDF, der Filterung englischer Stoppwörter und keiner Filterung und der Auswahl der jeweiligen n-Gramme liegen jeweils nur 1% Unterschied. Die Aufteilungskriterien Gini Impurity und Entropie unterscheiden sich nur um 0,2%.
Diese ausgezeichneten Punktzahlen legen nahe, dass eine zuverlässige Trollerkennung mit diesem Verfahren sehr gut zu leisten ist. Sowohl die geringen Schwankungen unter den Hyperparametern, als auch die geringen Unterschiede in den Gütekriterien sprechen für eine hohe Stabilität des Verfahrens. Dies bedeutet auch hier, dass selbst zufällige Einstellungen niemals zu schlechten Ergebnissen führen können.

| Hyperparameter | Genauigkeit | Relevanz | Segreganz | Sensitivität | Spezifität | $F_1$ |
|---|---|---|---|---|---|---|
| TF | 0.945 | 0.938 | 0.951 | 0.948 | 0.941 | 0.943 |
| TF-IDF | 0.936 | 0.930 | 0.941 | 0.937 | 0.934 | 0.934 |
| engl. Stoppwörter | 0.938 | 0.932 | 0.943 | 0.939 | 0.936 | 0.936 |
| keine Filterung | 0.942 | 0.935 | 0.949 | 0.946 | 0.939 | 0.941 |
| 1-Gramme | 0.944 | 0.937 | 0.951 | 0.948 | 0.948 | 0.943 |
| 1+2-Gramme | 0.936 | 0.931 | 0.941 | 0.937 | 0.937 | 0.934 |
| Gini | 0.939 | 0.933 | 0.945 | 0.942 | 0.937 | 0.937 |
| Entropie | 0.941 | 0.935 | 0.947 | 0.944 | 0.939 | 0.939 |
| Maximum | 0.950 | 0.944 | 0.957 | 0.954 | 0.947 | 0.949 |
| durchschnittl. | 0.940 | 0.934 | 0.946 | 0.943 | 0.938 | 0.938 |

Tabelle 8: Ergebnisse der Entscheidungsbaum-Klassifikation

### 5.2.5 Mehrschichtiges Perzeptron

Der wichtigste einstellbare Hyperparameter bei der Klassifikation mit einem Mehrschichtigen Perzeptron ist die Aktivierungsfunktion. Bei der Hyperparameteroptimierung wurden die drei in Kapitel 3.5 beschriebenen Aktivierungsfunktionen getestet.

| Hyperparameter | Genauigkeit | Relevanz | Segreganz | Sensitivität | Spezifität | $F_1$ |
|---|---|---|---|---|---|---|
| TF | **0.904** | 0.868 | **0.947** | **0.948** | 0.863 | **0.906** |
| TF-IDF | 0.888 | 0.868 | 0.909 | 0.907 | **0.870** | 0.887 |
| engl. Stoppwörter | 0.891 | 0.862 | 0.925 | 0.924 | 0.860 | 0.892 |
| keine Filterung | **0.901** | **0.873** | **0.931** | **0.931** | **0.873** | **0.901** |
| 1-Gramme | **0.898** | 0.868 | **0.932** | **0.932** | **0.932** | **0.898** |
| 1+2-Gramme | 0.894 | 0.868 | 0.925 | 0.923 | 0.923 | 0.894 |
| relu | **0.931** | **0.917** | **0.945** | **0.942** | **0.920** | **0.929** |
| tanh | 0.887 | 0.858 | 0.918 | 0.918 | 0.858 | 0.887 |
| logistic | 0.871 | 0.829 | 0.922 | 0.923 | 0.822 | 0.873 |
| Maximum | 0.937 | 0.928 | 0.968 | 0.972 | 0.933 | 0.936 |
| durchschnittl. | 0.896 | 0.868 | 0.928 | 0.928 | 0.866 | 0.896 |

Tabelle 9: Ergebnisse der MLP-Klassifikation

Tabelle 9 sind nun folgende Aussagen zu entnehmen: Im Durchschnitt liefert die MLP-Klassifikation Werte von mindestens 86%. Maximal werden Punktzahlen von 93 - 97% erreicht. Letztere Werte sind überwiegend auf die Aktivierungsfunktion ReLu zurückzuführen, welche in allen Gütemaßen 3 - 6% besser abschneidet als die anderen beiden.
Auch bei den Parametern der Merkmalsextraktion sind diesmal deutlichere Unterschiede festzustellen. Die Merkmalgewichtung mit TF liefert, analog zu allen anderen Klassifikationsverfahren, in den meisten Gütemaßen deutlich bessere Ergebnisse als TF-IDF. Filtert man englische Stoppwörter aus den Tweets, führt dies zu 1% schlechteren Ergebnissen, als wenn man dies unterlassen würde. Die ausschließliche Extraktion von 1-Grammen ist leicht besser als die Extraktion von 1- und 2-Grammen.
Die MLP-Klassifikation ist durch ihre hohen Punktzahlen in all ihren Varianten grundsätzlich für die Trollerkennung geeignet. In besonderem Maße gilt dies, wenn ReLu als Aktivierungsfunktion eingesetzt wird. Mit gut eingestellten Hyperparametern ist dieses Verfahren äußerst verlässlich. Interessant ist in diesem Zusammenhang folgendes: Die Trainingsphase mit Backpropagation wird bei einem Mehrschichtigen Perzeptron dann beendet, wenn sich der Klassifikationsscore $n$ Iterationen in Folge nicht um einen Toleranzwert $tol$ ändert. Um eine akzeptable Laufzeit zu gewährleisten, wurden die Werte bei der Hyperparameteroptimierung auf $n = 5$ und $tol = 0,25\%$ gesetzt. Dies bedeutet, dass sich bei Inkaufnahme einer langen Laufzeit die Punktzahlen in dem ein oder anderen Gütemaß um wenige Prozentpunkte verändern können.

## 5.3 Verfahren im Vergleich

Die

| Verfahren | Genauigkeit | Relevanz | Segreganz | Sensitivität | Spezifität | $F_1$ |
|---|---|---|---|---|---|---|
| KNN | **0.958** | **0.943** | **0.973** | **0.972** | **0.945** | **0.957** |
| NB | 0.830 | 0.779 | 0.898 | 0.907 | 0.759 | 0.838 |
| SVM | 0.892 | 0.866 | 0.919 | 0.918 | 0.868 | 0.892 |
| Baum | 0.943 | 0.937 | 0.950 | 0.947 | 0.940 | 0.942 |
| MLP | 0.934 | 0.926 | 0.943 | 0.939 | 0.929 | 0.932 |
| durchschnittl. | 0.911 | 0.890 | 0.937 | 0.937 | 0.888 | 0.912 |

Tabelle 10: Ergebnisse im Vergleich

## Literatur

Eija Airio (2006). „Word normalization and decompounding in mono-and bilingual IR“. In: *Information Retrieval* 9.3, S. 2–3.

Gongde Guo, Hui Wang, David Bell und Yaxin Bi (Aug. 2004). „KNN Model-Based Approach in Classification“. In.

Philip N Howard, Bharath Ganesh, Dimitra Liotsiou, John Kelly und Camille François (2019). *The IRA, social media and political polarization in the United States, 2012-2018*, S. 3.

Thorsten Jabs (2017). *„AfD hat 30 Prozent des Social-Media-Traffics ausgemacht“*. URL: https://www.deutschlandfunkkultur.de/rueckblick-auf-den-wahlkampf-im-netz-afd-hat-30-prozent-des.1008.de.html?dram:article_id=396594.

I.T. Joliffe (2002). *Principal Component Analysis*. 2. Aufl. Springer Series in Statistics. Springer.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes und Donald Brown (2019). „Text Classification Algorithms: A Survey“. In: *Information* 10.4.

Philip Kreißel, Julia Ebner, Alexander Urban und Jakob Guhl (Juli 2018). *Hass auf Knopfdruck*. Rechtsextreme Trollfabriken und das Ökosystem koordinierter Hasskampagnen im Netz. URL: http://www.isdglobal.org/wp-content/uploads/2018/07/ISD_Ich_Bin_Hier_2.pdf.

Deepika Kumawat und Vinesh Jain (Mai 2015). „POS Tagging Approaches: A Comparison“. In: *International Journal of Computer Applications* 118.6, S. 32.

Daniel D. Lee und H. Sebastian Seung (1999). „Learning the parts of objects by non-negative matrix factorization“. In: *Nature* 401, S. 788–791.

Darren L. Linvill und Patrick L. Warren (2018). *Troll Factories: The Internet Research Agency and State-Sponsored Agenda Building*.

Christopher D. Manning und Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, S. 124–125.

Tomas Mikolov, Kai Chen, Greg Corrado und Jeffrey Dean (2013). „Efficient Estimation of Word Representations in Vector Space“. In: arXiv: 1301.3781.

Janmenjoy Nayak, Bighnaraj Naik und Behera H. S. (2015). „A Comprehensive Survey on Support Vector Machine in Data Mining Tasks: Applications Challenges“. In: *International Journal of Database Theory and Application* 8.1, S. 169–186.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay (2011). „Scikit-learn: Machine Learning in Python“. In: *Journal of Machine Learning Research* 12, S. 2825–2830.

Juan Ramos (2003). *Using TF-IDF to Determine Word Relevance in Document Queries*.

Irina Rish (Jan. 2001). „An Empirical Study of the Naïve Bayes Classifier“. In: *IJCAI 2001 Work Empir Methods Artif Intell* 3.

Qifan Zhou, Hai Zhang, Zahra Lari, Zhenbo Liu und Naser El-Sheimy (Okt. 2016). „Design and Implementation of Foot-Mounted Inertial Sensor Based Wearable Electronic Device for Game Play Application“. In: *Sensors* 16, S. 1752.

# Abbildungsverzeichnis



# Tabellenverzeichnis