Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Igor Arkhipov

Extraktion und Stimmungsanalyse von Tweets bezüglich bestimmter Schlüsselwörter

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Igor Arkhipov

Extraktion und Stimmungsanalyse von Tweets bezüglich bestimmter Schlüsselwörter

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr. Michael Neitzke

Eingereicht am: 13. April 2018

Igor Arkhipov

Thema der Arbeit
Extraktion und Stimmungsanalyse von Tweets bezüglich bestimmter Schlüsselwörter

Stichwörter
Twitter, Data Mining, Textklassifikation, Computerlinguistik, Machine Learning, Stimmungsanalyse

Kurzzusammenfassung
Die vorliegende Bachelorarbeit beschreibt die Herangehensweise zur Bewertung der Nachrichten im Kurznachrichtendienst „Twitter“ hinsichtlich ihrer Tonalität. Als Ziel wird die Klassifizierung von Meinungen zu den eingegebenen Schlüsselwörtern gesetzt, ob also eine positive oder negative Stimmung vorliegt.

Igor Arkhipov

Title of the paper
Retrieval and Sentiment Analysis of Tweets including specific keywords

Keywords
Twitter, Data Mining, Text Classification, Computational Linguistics, Machine Learning, Sentiment Analysis

Abstract
This bachelor thesis describes the approach to classify the messages in the short message system "Twitter" with regard to their tonality. The objective is defined as opinion classification for specific keywords. Either their mentioning has a positive sentiment or a negative one.

Inhaltsverzeichnis

1 Einleitung

Seit ihrer Gründung haben die sozialen Netzwerke wie Facebook[1] und Instagram[2] schnell an Popularität gewonnen. Heutzutage teilen Millionen von Menschen ihre Eindrücke mit ihren Freunden und Bekannten weltweit. Die Benutzer von weit verbreiteten drahtlosen Netzwerken oder des mobilen Internets können die Nachrichten so gut wie zu jedem beliebigen Zeitpunkt verschicken. So veröffentlichen zum Beispiel die Journalisten ihre Textberichte in Echtzeit oder auch die Touristen, die ihre Eindrücke über aktuelle Reisen von überall her auf der Welt mit anderen teilen.

Viele namhafte Unternehmen und Organisationen, wie auch prominente Persönlichkeiten, die mehr Präsenz in der Öffentlichkeit zeigen wollen, besitzen ihre Profile in beliebten sozialen Netzwerken. Sie teilen ihre Gedanken und Ansichten im Kreis interessierter Personen mit und erkundigen sich über die öffentliche Meinung in Bezug auf von ihnen getroffene Entscheidungen.

Während man früher hauptsächlich auf das Feedback des Freundes- bzw. Bekanntenkreises angewiesen war, werden die Bewertungen in Online-Shops, Blogs, sowie auf den Seiten spezialisierter Ressourcen in den letzten Jahren immer wichtiger.

Laut einer Umfrage von Dimensional Research [DimensionalResearch (2013)] glauben 88% der Befragten daran, dass die positiven oder negativen Kommentare im Internet ihre Kaufentscheidung beeinflussen.

Daher bieten die sozialen Netzwerke den Forschern verschiedene Möglichkeiten, eine detaillierte Analyse der Ansichten der Nutzer durchzuführen. Zum Beispiel wurde ein

[1] https://www.facebook.com (09.04.2018)
[2] https://www.instagram.com (09.04.2018)

Forschungsprojekt „Puls der Nation“ [Cha u. a. (2010)] entwickelt, um die Stimmung der US-Bürger während des Tages anhand des Kurznachrichtendienstes „Twitter“ zu bestimmen.

„Twitter“[3] ist derzeit einer der beliebtesten sozialen Netzwerke. Die Anzahl der monatlich aktiven Nutzer hat längst 300 Millionen überschritten [Statista (2017)], und all diese Nutzer veröffentlichen mehr als 500 Millionen Nachrichten pro Tag [Krikorian (2013)]. Die Beschränkung der Länge der Nachricht (diese darf beim Twitter maximal 280 Zeichen lang sein), als auch Slang, Rechtschreib- und Grammatikfehler machen die Suche und Analyse der Meinungen zur einer nicht trivialen Aufgabe.

Wenn die Unternehmen die Kundenmeinung über ihre Produkte oder Dienstleistungen verstehen wollen, führen sie meistens traditionelle Umfragen durch, organisieren die Fokusgruppen oder beauftragen eine Beraterfirma. Dies könnte durch entwickelte Tools zur steuerbaren automatisierten Analyse von Meinungen vermieden oder reduziert werden. Es ist praktisch unmöglich, eine Reihe von Foren oder Webseiten manuell zu verarbeiten, um interessante Ansichten zu extrahieren, ihre Stimmung zu erkennen, die resultierenden Daten zu verallgemeinern und das Ergebnis in einer leicht verständlichen Form auszugeben.

Die Lösung solcher Aufgabe liegt im Schnittpunkt der Informationssuche und der Computerlinguistik und gehört zum Bereich der Meinungsextraktion und der Stimmungsanalyse (Engl.: Opinion mining, Sentiment analysis). Die Meinungsextraktion bedeutet, dass emotionales Lexikon und Meinungen in Bezug auf die im Text enthaltenen Objekte automatisch extrahiert werden. Die Stimmungsanalyse klassifiziert die Texte auf Grund der Stimmung oder Tonalität, die die emotionale Haltung des Autors ausdrückt.

Derartige Aufgaben sind vor relativ kurzer Zeit entstanden. Die Stimmungsanalyse zieht große Unternehmen aufgrund der erheblichen Verbreitung von Social Media mehr und mehr an. Die Marktforscher müssen immer mehr Medien auf der Suche nach Referenzen auf ihre Marken überwachen. Zur Zeit gibt es noch keinen optimalen Algorithmus, der alle Aufgaben im Bereich der Stimmungsanalyse zufriedenstellend löst. Dies macht das Problem der Meinungsextraktion und ihre Bewertung so aktuell.

[3] https://twitter.com (09.04.2018)

2 Aufgabenstellung

Wir gehen von einer Reihe von Nachrichten (auch als Dokumente bezeichnet) aus, die die Meinungen ihrer Autoren über ein Ereignis, Phänomen oder Objekt in einem bestimmten Fachgebiet darstellen. Sie drücken die emotional gefärbten Meinungen ihrer Autoren aus, ohne dabei eine konkrete Bewertung zu enthalten.

Im Rahmen dieser Arbeit soll ein Verfahren für automatische Erfassung und Analyse der Meinungen zu einem von dem Benutzer eingegebenen Thema entwickelt werden. Für die Verarbeitung und Analyse werden englischsprachige Quellen benutzt.

Um dieses Ziel zu erreichen, wurden folgende Aufgaben gesetzt:

1. Einen Überblick über die existierenden Methoden für die automatische Analyse der emotionalen Tonalität von Texten zu geben.

2. Eine Analyse von Textmerkmalen von Nachrichten in sozialen Netzwerken im Zusammenhang mit der Entwicklung von Methoden zur Analyse ihrer emotionalen Tonalität durchführen.

3. Eine Methode für die automatische Analyse von emotional gefärbten Nachrichten im sozialen Netzwerk Twitter entwickeln.

4. Eine Webanwendung für die praktische Verwendung dieser Methode als „Proof of Concept“ bereitstellen.

3 Überblick über den Themenbereich

Der folgende Überblick befasst sich mit Problemen, die sich aus der Stimmungsanalyse von Texten ergeben, und führt in die damit zusammenhängende Terminologie ein. Eine besondere Aufmerksamkeit wird auf die Methoden zur Bestimmung der Tonalität der emotional gefärbten Texte gelegt.

3.1 Terminologie

Zwei wichtige Begriffe im aktuellen Fachgebiet sind „**Stimmungsanalyse**“ (Engl.: Sentiment analysis) und „**Meinungsanalyse**“ (Engl.: Opinion mining).

Unter einer Stimmungsanalyse von Texten versteht man die Aufgabe der automatischen Analyse von Meinungen und emotional gefärbten Lexik, die im Text ausgedrückt sind. [Pang und Lee (2008)]

Bei solcher Untersuchung der Tonalität des Textes wird davon ausgegangen, dass die Textinformation im Internet in zwei Klassen unterteilt wird und zwar Fakten und Meinungen [Pang und Lee (2008)]. Die Fakten sind objektive Aussagen über bestimmte Entitäten oder Ereignisse. Die Meinungen sind subjektive Aussagen, die die Haltung einer Person oder die Wahrnehmung eines Ereignisses oder eines Unternehmens widerspiegeln. Die meisten bestehenden Studien, die die Verarbeitung natürlicher Sprache untersuchen, konzentrieren sich auf das Sammeln und Extrahieren von sachlichen Informationen. Ein Beispiel dafür ist die traditionelle Informationssuche im Allgemeinen und insbesondere die Websuche.

Als **Objekt** (Engl.: object, topic) der Stimmungsanalyse kann jede Entität gelten, zu der eine Meinung geäußert wird. Zum Beispiel könnte es ein Produkt, eine Dienstleistung, eine Organisation oder eine Veranstaltung sein. Ein Objekt kann viele *Komponenten* (Bestandteile) und *Attribute* (Eigenschaften) haben, die zusammen eine Menge von **Aspekten** (Engl.: features, aspects, facets) bilden [Liu (2010a)].

Ein bestimmtes Handymodell ist ein Objekt. Es besteht aus Komponenten (der Bildschirm, die Kamera, der Akku ...) und hat eine Reihe von Attributen (die Sprachqualität, die Größe ...), die zusammen eine Aspektmenge bilden. Die Meinung kann sowohl über das Objekt selbst als auch über irgendwelche Aspekte ausgedrückt werden. In der Beispielnachricht „Ich mag das neue iPhone. Die Qualität des Bildschirms ist einfach erstaunlich!“ drückt der erste Satz eine positive Meinung über das Objekt „das iPhone“ und der zweite - über seinen Aspekt „der Bildschirm“ aus.

Der **Meinungssteller** (Engl.: opinion holder, opinion source) ist eine Person oder eine Organisation, die diese Meinung ausdrückt.

Der **Meinungston** (Engl.: opinion orientation) ist eine emotionale Einschätzung des Meinungsstellers zu einem Objekt und / oder einem seiner Aspekte, zum Beispiel *Positiv* oder *Negativ*. Streng genommen kann die Stimmung eine größere Anzahl von Werten annehmen, diese zwei Klassen werden jedoch am häufigsten verwendet. In einigen Werken [Pang und Lee (2008), Liu (2010b)] ist auch der Wert *Neutral* zwischen Positiv und Negativ definiert, und in den anderen wird der Mittelwert lediglich als Abwesenheit der subjektiven Bewertung betrachtet. In dieser Arbeit hält sich der Autor an die letzte Interpretation wegen der Schwierigkeit, eine zwischenstehende Bedeutung zu formalisieren. Zum Beispiel ist die Auswertung in der Aussage „Ein mittelmäßiger Bildschirm“ eher negativ, trotz der Tatsache, dass das Wort „mittelmäßig“ im Wörterbuch als „durchschnittlich, einigermaßen“ [Dudenredaktion (2017)] definiert ist.

Das **Modell eines Objekts** o ist eine endliche Menge von Aspekten $F = \{f_1, f_2, ..., f_n\}$, die das Objekt selbst als einen besonderen Aspekt einschließt.

Die **Meinungen** werden in zwei Arten unterteilt:

1. die einfache Meinung (Engl.: direct opinion). Eine einfache Meinung ist formell als ein Tupel $\langle entity, feature, sentiment\ value, holder, time \rangle$ definiert, wobei der Autor der Meinung $holder$ dem Aspekt $feature$ des Objekts $entity$ eine Schätzung $sentiment\ value$ zum Zeitpunkt $time$ gab. Zum Beispiel in der Aussage „Ubuntu bietet eine sehr schöne graphische Oberfläche an“ gab der Autor der Meinung eine positive Bewertung „schöne“ dem Aspekt „Oberfläche“ des Objekts „Ubuntu“.

2. die vergleichbare Meinung (Engl.: comparative opinion). Eine vergleichbare Meinung kann wiederum in drei Arten unterteilt werden [Jindal und Liu (2006)]:

   a) der Vergleich der Aspekte von Objekten zugunsten eines Objekts (Engl.: Non-equal gradable);

   b) die Gleichstellung der Aspekte verschiedener Objekte (Engl.: equative);

   c) die Überlegenheit eines Objekts über andere Objekte (Engl.: superlative)

   Die Vergleiche des ersten Untertyps haben die Form „der Aspekt des Objekts 1 übertrifft den Aspekt des Objekts 2“ wie im Beispiel: „Der Galaxy Tab Bildschirm ist schärfer als bei den Konkurrenten“. Der zweite Untertyp drückt die Ähnlichkeit von Aspekten verschiedener Objekte aus, zum Beispiel: „Sowohl Android als auch iOS sind für die mobile Anwendungsentwicklung gleichermaßen geeignet.“ Ein Beispiel für den dritten Typ lautet folgenderweise: „Das Gerät von Canon erwies sich als die billigste Kamera in diesem Laden.“

   Die vergleichbare Meinung ist formell als Tupel $\langle E_1, E_2, A, po, holder, time \rangle$ definiert, wobei E_1 und E_2 die Sätze von verglichenen Objekten zu bestimmendem Aspekt A sind. po bedeutet die Menge der Objekte, die der Autor ($holder$) bevorzugt hat. $time$ ist der Zeitpunkt, wann die Meinung veröffentlicht wurde.

   Zum Beispiel hat der Autor der Ansicht „Die graphische Oberfläche in Ubuntu ist einfacher als in Windows 8“ das „Ubuntu“-Objekt gegenüber dem „Windows 8“-Objekt bevorzugt, was dem „Oberfläche“ Aspekt angeht.

Im Unterschied zum Tupel der ersten Art, enthält das Tupel für die vergleichbare Meinung keine Einschätzung der Emotionen des Autors.

Bei der Stimmungsanalyse des Textes gibt es einen Begriff **Subjektivität**, der oft mit der Meinung zusammen vorkommt. In [Pang und Lee (2008)] ist die folgende Definition von objektiven und subjektiven Sätzen gegeben: „Ein objektiver Satz drückt sachliche Informationen über etwas aus, während ein subjektiver Satz persönliche Gefühle und Annahmen von jemandem ausdrückt."

Die Sätze, die eine Meinung enthalten, sind in der Regel subjektiv, so dass die Analyse des Textes für subjektive Informationen oft eine Teilaufgabe der Bestimmung der Tonalität des Textes ist.

Diese Bachelorarbeit verwendet auch Twitter-spezifische Terminologie.

Unter **Mikroblog** versteht man den Analog des üblichen Web-Blogs (Online-Tagebuch), wobei die Länge der einzelnen Nachricht in Twitter auf 280 Zeichen begrenzt ist.

Der **Tweet** ist die einzelne Nachricht des Benutzers. Solche Nachricht kann beliebigen Text, eine Erwähnung eines anderen Benutzers, Hyperlinks oder Hashtags enthalten.

Der **Retweet** bedeutet die Fähigkeit, die Nachricht eines anderen Benutzers auf ihre eigene Pinnwand weiterzuleiten. Somit wird die Autorenschaft erhalten.

Das **Hashtag** ist ein Wort oder eine Phrase, denen ein # Zeichen vorangeht. Die Benutzer können eine Gruppe von Nachrichten nach Thema mit ihrer Hilfe zusammenführen.

Die **Erwähnung** (Engl.: mention) ist eine Referenz auf einen anderen Benutzer. Sie beginnt mit dem @-Zeichen und wird von dem Username eines anderen Benutzers gefolgt. Manchmal wird die Erwähnung statt eines Hashtags zum Gruppieren von Nachrichten nach Themen verwendet.

3.2 Aufgaben der Stimmungsanalyse

In der Computer-Linguistik gilt ein natürlichsprachlicher Text als unstrukturierte Information. Anhand der Definition der Meinung bestimmen wir in diesem Fall, wie diese Information strukturiert werden muss. Bei Problemen, die unter den Begriff „die Stimmungsanalyse des Textes“ fallen, wird bestimmt, wie die Daten aus dem Text in der natürlichen Sprache extrahiert, analysiert und strukturiert werden.

Zu den wichtigsten Aufgaben im Bereich der Stimmungsanalyse gehören [Pang und Lee (2008)]:

1. Die Aufgabe, Dokumente oder Teile eines Dokuments zu identifizieren, die eine Meinung enthalten (Engl.: subjectivity detection). Für Dokumente wird diese Aufgabe auf die Aufgabe der binären Textklassifikation in Klassen von subjektiven (mit einer Meinung) und objektiven (mit Fakten) Dokumenten reduziert.

2. Die Aufgabe der Bestimmung der Tonalität. Diese Aufgabe wird wiederum auf eine binäre Textklassifikation in Klassen von positiv und negativ gefärbten Dokumenten reduziert.

3. Anschließend ist es erforderlich, die Informationen zu Stellungnahmen kurz zusammen zu fassen, wie z.B.:

    a) die Aggregation von Kundenbewertungen;

    b) die Identifizierung von Autorengruppen mit ähnlichen Ansichten;

    c) die Identifikation von Übereinstimmungspunkten und Widersprüchen;

    d) die textuelle oder statistische Verallgemeinerung von Dokumenten.

Die Aufgabe der Bestimmung der Tonalität wird am häufigsten angetroffen und wird weiter in der Arbeit genauer betrachtet.

3.3 Bestimmung der Tonalität des Textes

Die Aufgabe, die Tonalität des Textes zu bestimmen, wird wie folgt formuliert: „Ist die emotionale Stimmung des Textes positiv oder negativ?“ Diese Aufgabenstellung wird auf zwei Ebenen betrachtet:

1. auf der Dokumentebene;

2. auf der Satzebene.

Bevor wir zu diesen Ebenen übergehen, werden mögliche Probleme bei der Bestimmung der Tonalität betrachtet.

3.3.1 Probleme bei der Bestimmung der Tonalität des Textes

Die Probleme, die bei der Bestimmung der Tonalität des Textes auftreten, können in allgemeine und Mikroblog-spezifische Probleme unterteilt werden.

Zu den allgemeinen Problemen gehören die folgenden:

1. Die Abhängigkeit des Tonalitätswertes von dem Fachgebiet. Obwohl im Filmlexikon das Wort „unvorhersehbar“ eine positive Färbung haben kann, kann das gleiche Wort im Bereich des Kundendienstes eine negative Bedeutung hervorrufen.

   In der Praxis werden die Benutzeranforderungen nicht notwendigerweise auf einen Bereich begrenzt, so dass die mögliche Kategorisierung von Text in zwei Durchgängen aufgeteilt werden kann: zuerst wird eine thematische Klassifizierung des Dokuments durchgeführt, danach folgt die Bestimmung der Tonalität.

2. Die Verwendung von Negation kann die Tonalität des Restes des Ausdrucks umdrehen [Wiegand u. a. (2010)]. Betrachten wir die Äußerung: „Früher mochte ich Nokia Handys sehr gerne. Die Bauqualität des Lumia 800 ist in der Tat sehr hoch. Jedoch schädigt das Windows Phone 7 den Ruf komplett.“ Im ersten und zweiten

Satz spricht der Autor eine positive Meinung aus, aber aufgrund des Einsatzes der Negation im dritten Satz ist der allgemeine Ton für das Objekt „Nokia“ negativ.

3. Die Verwendung von Sarkasmus in den Nachrichten geht schlecht mit der Erkennung der Tonalität zusammen. Die sarkastischen Aussagen können einen gemeinsamen Ton haben, der das Gegenteil von dem Ton einzelner Wörter („Ein ausgezeichnetes Buch für die Schlaflosen!“) ist. Alternativ kann ein Satz eine Stellungnahme in einer latenten Form (die Frage „Wo bin ich?“ im Rahmen des Tests für den GPS-Navigator) ausdrücken. In diesem Fall kann der Sarkasmus auch von einem Menschen kaum erkannt werden.

   In einem der aktuellen Artikel in diesem Bereich [Davidov u. a. (2010b)] waren die Autoren in der Lage, die Genauigkeit von bis zu 78% für die Sammlung von Rezensionen zu Produkten zu erreichen.

4. Der Wert des Meinungstons hängt auch davon ab, wer die Analyse durchführen will. Wobei für die Firma „Samsung“ die Aussage „Das Galaxy S7 zeigt einen ausgezeichneten Verkauf :)“ einen positiven Ton bringt, bedeutet die gleiche Aussage für die Firma „Apple“ das Gegenteil.

Zu den Twitter-spezifischen Problemen gehören:

1. Ein großes Wörterbuch mit nicht häufig verwendeten Wörtern. Basierend auf der Studie [Saif u. a. (2012)] werden 93% der gefundenen Wörter weniger als 10 Mal verwendet (78% im Fall der Lernstichprobe mit den IMDB-Filmen). Dies passiert aufgrund der häufigen Verwendung von Umgangssprache, beabsichtigter und unbeabsichtigter Verstümmelung der Rechtschreibung oder wegen der unterschiedlichen Schreibweise der gleichen Wörter (eine Kombination aus Groß- und Kleinbuchstaben).

2. Die geringe Anzahl von Wörtern in jeder Nachricht. Aufgrund der Beschränkung der Nachrichtenlänge bis vor kurzem [Rosen (2017)] auf 140 Zeichen beträgt die durchschnittliche Nachrichtenlänge in Englisch 14 Wörter oder 78 Zeichen [Go u. a. (2009)].

3. Große Datenmengen für die lokale Verarbeitung wegen der täglichen Veröffentlichung von mehr als 500 Millionen Nachrichten [Krikorian (2013)].

3.3.2 Bestimmung der Tonalität des Dokuments

Die Aufgabe besteht darin, die Tonalität des Dokuments als Ganzes zu bestimmen. Darüber hinaus kann der zugehörige Text des Dokuments gleichzeitig die Sätze sowohl mit negativer als auch positiver emotionaler Färbung enthalten.

Auf der Dokumentebene wird davon ausgegangen, dass das Dokument eine Meinung zu einem Objekt enthält. Die Reviews von Waren und Dienstleistungen auf spezialisierten Internetressourcen erfüllen diese Annahme.

Der Bestimmung der Tonalität des Dokuments wurden viele frühere Arbeiten über die Stimmungsanalyse des Textes gewidmet. In den meisten modernen Systemen der Meinungsanalyse wird dieses Problem nicht berücksichtigt, weil das Ergebnis - die Tonalität des Dokuments im Durchschnitt - als nicht-informative Schätzung der in dem Dokument geäußerten Meinung gilt. [Liu (2012)]

In vielen Anwendungen muss der Benutzer vorher genau wissen, welche Aspekte von Entitäten von den Kunden beliebt oder unbeliebt werden. Typische Reviews stellen solche Details bereit, aber diese Details werden bei der Tonalität des Dokuments als Ganzes außer Acht gelassen.

3.3.3 Bestimmung der Tonalität des Satzes

Die Aufgabe, die Tonalität des Satzes zu bestimmen, wird nicht nur als Subtask der Analyse der Tonalität des Dokuments, sondern auch als eigenständige Aufgabe gelöst, zum Beispiel bei der Analyse von Kurznachrichten und Kommentaren in sozialen Netzwerken.

Oft findet die Bestimmung der Tonalität des Satzes in zwei Stufen statt [Pang und Lee (2008)]. Zuerst wird der Satz auf Subjektivität geprüft. Wenn der Satz Informationen von

einer subjektiven Natur enthält, dann ist es höchstwahrscheinlich eine Meinung. Als nächstes wird die Tonalität des subjektiven Satzes bestimmt. Andernfalls enthält der Satz eine sachliche Information und wird nicht weiter berücksichtigt.

Die Methoden zur Bestimmung der Tonalität einzelner Sätze sind für die höhere Genauigkeit der Meinungsanalyse im Text vorgesehen: zusammen mit den Methoden zur Extraktion von Aspekten kann man eine ausführliche Analyse der Meinung des Autors über alle Aspekte der im Text betroffenen Objekte durchführen.

3.3.4 Bestimmung der Tonalität als Klassifizierungsproblem

Auf spezialisierten Ressourcen wird die Kundenmeinung über ein Produkt oft von zugehöriger Bewertung nach einer bestimmten Skala begleitet. Daher wird in den meisten Studien die Bestimmung der Tonalität als Aufgabe der Klassifizierung eines Textes betrachtet.

Die Klassen können die Bewertungsnoten des Meinungsautors sein, und die Merkmale werden aus dem Meinungstext extrahiert: oft werden die N-Grammme von Wörtern (Folgen von N linguistischen Elementen) und ihre Sprachteile als Attribute ausgewählt. Um das Klassifizierungsproblem zu lösen, werden verschiedene Methoden des maschinellen Lernens verwendet. [Pang u. a. (2002)]

3.4 Methoden der automatischen Bestimmung der Tonalität

Die wichtigsten Ansätze zur Definition der Tonalität können in die folgenden Kategorien unterteilt werden:

- regelbasierten Methoden;
- lexikonbasierten Methoden;

- Lernmethoden mit Überwachung;
- Lernmethoden ohne Überwachung.

In weiteren Kapiteln werden diese Ansätze detailliert beschrieben.

3.4.1 Regelbasierte Methoden zur Bestimmung der Tonalität

Die regelbasierte Methode (Engl.: rule-based approach) zur Bestimmung der Tonalität besteht darin, einen Satz von Regeln anzuwenden, die von Experten auf der Grundlage einer Domänenanalyse identifiziert werden. [Kan (2012)]

Ein Beispiel für eine solche Regel ist: wenn die Äußerung ein oder mehrere positive Adjektive aus der Menge $\{$„gut“, „$qualitativ$“, „$frisch$“, $\dots\}$ und keine Adjektive aus der Menge $\{$„$inkompetent$“, „$schrecklich$“, „$gruselig$“, $\dots\}$ enthält, dann ist der Meinungston „positiv“.

Solcher Ansatz kann mit Hilfe eines großen Regelsatzes potentiell gute Ergebnisse leisten.

Es gibt aber folgende Nachteile:

1. Das Erstellen eines Regelsatzes ist ziemlich zeit- und arbeitsaufwendig.
2. Die Anwendung dieses Ansatzes auf Mikroblogs kann aufgrund von „verrauschten“ Daten problematisch sein.

3.4.2 Lexikonbasierte Methoden zur Bestimmung der Tonalität

Es gibt auch Methoden, die das Wörterbuch der emotional gefärbten Wörtern (Engl.: affective lexicons) [Denecke (2008), Thelwall u. a. (2010)] und das Wörterbuch der Symbolen, die Emotionen bezeichnen, [Davidov u. a. (2010b), Thelwall u. a. (2010), Thelwall u. a. (2011)] verwenden. Bei solchen lexikonbasierten Methoden hat jedes Wort ein Gewicht,

das seine emotionale Farbe charakterisiert. Oft werden die Wörterbücher mit Hilfe von Drittanbietern wie WordNet[4] kompiliert, dann werden die Begriffe des Wörterbuchs beispielsweise manuell gewichtet [Thelwall u. a. (2011)].

In Studien, die auf derartige Methoden basieren, wird die Tonalität des Textes auf verschiedene Weise bestimmt. In [Denecke (2008)] hat der Text drei unabhängige Bewertungen: positiv, negativ und objektiv. Die Tonalität des Dokuments besteht aus den Tonalitäten seiner Sätze. Die Bestimmung der Tonalität des Satzes besteht darin, die normalisierten Summen der Gewichte der Begriffe im Text zu berechnen: jeder Begriff hat ein positives, negatives und objektives Gewicht (w_1, w_2, w_3), wobei $w_i \in \{0, 1\}, \sum w_i = 1$.

In [Thelwall u. a. (2010)] werden kurze Textnachrichten unabhängig auf einer negativen und positiven Skala ausgewertet: von -5 bis -1 und von 1 bis 5. Jeder Begriff des Wörterbuchs hat zwei Gewichte, die in der gleichen Weise bestimmt werden.

Das positive Gewicht der Textnachricht ist gleich dem Gewicht des Begriffs mit der maximalen positiven Einschätzung. Wenn der Text keine Begriffe zur Verfügung stellt, die im Wörterbuch enthalten sind, wird ihm ein minimales positives Gewicht zugewiesen. Ähnlich wird das negative Gewicht der Nachricht bestimmt.

Das Hauptproblem bei den lexikonbasierten Methoden ist der Prozess der Erstellung eines Wörterbuchs. Um eine Methode zu erhalten, die das Dokument mit hoher Genauigkeit klassifiziert, müssen die Begriffe des Wörterbuchs ein Gewicht haben, das dem Themenbereich des Dokuments entspricht. Zum Beispiel ist das Wort „groß“ in Bezug auf die Menge an Festplattenspeicher eine positive Eigenschaft, ist aber gleichzeitig negativ in Bezug auf die Größe des Mobiltelefons.

3.4.3 Lernmethoden mit der Überwachung

Die Lernmethoden mit der Überwachung (Engl.: supervised learning) gehören zu dem Abschnitt des maschinellen Lernens. Der Klassifizierungsalgorithmus wird auf der Basis einer Trainingsstichprobe trainiert, die aus Dokumenten besteht, deren Klassen im Voraus

[4] http://wordnet.princeton.edu (09.04.2018)

bekannt sind. [Pang u. a. (2002), Pang und Lee (2004), Whitelaw u. a. (2005), Kennedy und Inkpen (2006)]

Die Vorteile des Ansatzes sind folgende:

- Gute Genauigkeit bei der Bestimmung der Tonalität.
- Auf der Basis des Trainingsmusters identifiziert der Klassifikator eigenständig die Merkmale, die die Tonalität beeinflussen. Somit wird das Problem der Abhängigkeit vom Themenbereich gelöst, indem eine Trainingsprobe aus dem gleichen Sachgebiet verwendet wird.
- Es gibt viele Möglichkeiten, die Genauigkeit zu verbessern.

Es gibt aber auch Nachteile:

- Eine markierte Trainingsprobe ist erforderlich.
- Die Ergebnisse können stark vom gewählten Algorithmus, seinen Parametern und der Trainingsprobe abhängen.

3.4.4 Lernmethoden ohne Überwachung

Die Lernmethoden ohne Überwachung (Engl.: unsupervised learning) stellen einen weiteren Teil des maschinellen Lernens dar. Der Unterschied zu dem überwachten Lernen besteht darin, dass eine Trainingsprobe in diesem Fall aus Dokumenten besteht, deren Klassen vorher unbekannt sind (oder diese Information nicht vom Algorithmus verwendet wird).

Also, für diesen Ansatz ist kein markierter Datensatz erforderlich. Allerdings ist die Genauigkeit solcher Methoden normalerweise niedriger als bei Algorithmen, die das überwachtes Training verwenden. [Lin und He (2009)]

3.5 Schlussfolgerung

Um das Problem der Bestimmung der Tonalität von Texten zu lösen, sind sowohl die Lernalgorithmen mit dem Lehrer als auch die lexikonbasierten Methoden wirksam und verbreitet.

Die Lernalgorithmen mit dem Lehrer haben die Anforderung, dass die Vorbereitung eines Trainingskorpus[5] mit den Beispielen aus dem Themenbereich notwendig ist, in dem der Klassifikator verwendet wird. Das lexikonbasierte Verfahren hat jedoch auch ein ähnliches Problem: die Gewichte der Terme eines Wörterbuchs, das für einen Fachbereich zusammengestellt wurde, können für ein anderes nicht angemessen sein.

[5] Ein Korpus ist eine Sammlung von Schriften, z.B. eine Sammlung von Tweets.

4 Forschung und Lösung des Problems

Die Hauptaufgabe dieser Bachelorarbeit ist die Umsetzung der Methode für die Analyse von emotional gefärbten Nachrichten im sozialen Netzwerk Twitter zu einem vom Benutzer eingegebenen Thema. Die Lösung dieses Problems kann in folgende Schritte aufgeteilt werden:

1. Untersuchung der Merkmale von Nachrichten im sozialen Netzwerk Twitter.

2. Umsetzung der Methode zur Bestimmung der Tonalität der Nachrichten unter Berücksichtigung dieser Merkmale und Prüfung ihrer Wirksamkeit.

4.1 Untersuchung der Merkmale von Nachrichten im sozialen Netzwerk Twitter

Im Zusammenhang mit der Analyse der emotionalen Färbung eines Textes gibt es folgende Hauptmerkmale von Nachrichten in Twitter:

1. Die Länge der Nachrichten beträgt nur 280 Zeichen. Die Mehrheit der Benutzer schreibt sogar weniger als 140 Zeichen pro Nachricht. [Rosen (2017)]

2. Das Vorhandensein von Abkürzungen, grammatikalischen Fehlern und Slang.

3. Das Vorhandensein von Sonderzeichen.

4. Die Links zu anderen Benutzern und externen Ressourcen sind weit verbreitet.

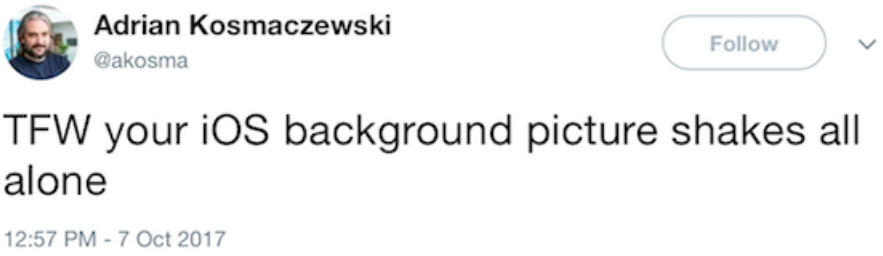

Abbildung 4.1: Tweet Beispiel
Quelle: https://twitter.com/akosma (07.10.2017)

In [Jansen u. a. (2009), Barbosa und Feng (2010), Thelwall u. a. (2010), Thelwall u. a. (2011), Tan u. a. (2011)] sind diese Merkmale der entscheidende Faktor bei der Entwicklung von Methoden. Zum Beispiel in [Thelwall u. a. (2010)] wird ein Algorithmus verwendet, um die Tonalität einer Nachricht mithilfe der lexikonbasierten Methode zu bestimmen. Das Wörterbuch enthält die Symbole und Wörter, die verschiedene Emotionen bezeichnen. Absichtliche grammatikalische Fehler in den Wortschatzbegriffen, wie die Wiederholung von Konsonantbuchstaben in Wörtern, zum Beispiel „Siegggg!“ anstelle von „Sieg“, erhöhen das Anfangsgewicht der Begriffe.

4.2 Implementierung des Algorithmus zur Erkennung von Kurznachrichtentonalität

Da die Ansätze, die auf überwachtem Training basieren, gute Ergebnisse bei der Analyse von traditionellen Blogs und Review-Sites leisten, wurde entschieden die überwachten maschinellen Lernmethoden zu verwenden.

Die Aufgabe, einen effektiven Algorithmus zur Bestimmung der Nachrichtentonalität zu implementieren, umfasst folgende Punkte:

1. Die Auswahl von Maßnahmen, mit denen die Wirksamkeit von Algorithmen ausgewertet wird.

2. Die Auswahl des Klassifikationsalgorithmus.

3. Die Auswahl der geeigneten Trainingsprobe.

4. Die Leistungstests.

Vor der Erläuterung dieser Punkte werden auch überwachtes Lernen und das Verfahren zur Datendarstellung in Bezug auf die aktuelle Aufgabe detailliert beschrieben.

4.2.1 Aufgabe der Klassifikation beim überwachten Lernen

Im Allgemeinen ist die Aufgabe der Textklassifikation wie folgt definiert [Manning u. a. (2008)]:

sei $d \in \mathbb{X}$ ein Dokument, wobei $\mathbb{X}$ ein Dokumentbereich ist, und einen festen Satz von Klassen $\mathbb{C} = \{c_1, c_2, \ldots, c_m\}$ gegeben. Unter Dokumentbereich wird eine Art hochdimensionaler Raum verstanden. Es ist notwendig, aus dem Trainingsbeispiel (der Satz von Dokumenten mit bekannten Klassen) $\mathbb{D} = \{\langle d, c\rangle | \langle d, c\rangle \in \mathbb{X} \times \mathbb{C}\}$ die Klassifizierungsfunktion (oder den Klassifikator) $\Gamma(\mathbb{D}) = \gamma$ unter Verwendung der Lernmethode Γ zu erhalten, die die Dokumente den Klassen zuordnet: $\gamma : \mathbb{X} \to \mathbb{C}$.

Bei der Bestimmung der Tonalität besteht die Menge $\mathbb{C}$ aus zwei Elementen: *positiv* und *negativ*.

4.2.2 Datendarstellung

Alle Dokumente aus dem Trainings- und Teststichprobe sind n-dimensionale Merkmalsvektoren (Engl.: feature vector).

Bei den Problemen der natürlichsprachigen Textverarbeitung ist die Darstellung von Dokumenten in Form von N-Grammen beliebt, wobei unter einem N-Gramm eine Folge von N Wörtern, die durch Leerzeichen getrennt sind, verstanden wird.

Für $N = 1$ besteht diese Sequenz aus einem Wort und wird Monogramm genannt. Das Modell, das eine Darstellung als Monogramme verwendet, wird ein „Bag-of-Words"-Modell genannt, weil die Wörter unabhängig voneinander betrachtet werden. Für $N = 2$ wird eine solche Sequenz als Bigramm bezeichnet, und so weiter.

Zum Beispiel für den Ausdruck „Es hat geregnet" besteht die Menge der Monogramme aus den Elementen {„Es", „hat", „$geregnet$"}. Und die Menge der Bigramme enthält entsprechend {„$Es\ hat$", „$hat\ geregnet$"}. Es sei darauf hingewiesen, dass je größer N, desto genauer reflektieren die N-Gramme den ursprünglichen Text. Gleichzeitig wird der Wert solcher N-Gramme für die Modellierung von einem beliebigen Text reduziert. Daher werden bei Textklassifikationsproblemen die Fälle mit $N > 3$ selten betrachtet.

Theoretisch ermöglicht die Verwendung von Bigrammen den Wert der Tonalität genauer zu modellieren. Wenn wir die Wortverbindung „gefällt nicht" als eine Sammlung von unabhängigen Wörtern betrachten, gibt es ein Wort mit einem negativen und ein Wort mit positiven Ton. Somit kann der Klassifikator entscheiden, dass der allgemeine Ton sich der Null nähert. Andererseits ist es unwahrscheinlich, dass das Bigramm „gefällt nicht" in einem positiven Kontext verwendet wird.

Also, das Dokument wird als Vektor $d = (\omega_1, \omega_2, \ldots, \omega_{\|\mathbb{V}\|})$ definiert, wobei $\mathbb{V}$ die Menge aller einzigartigen Begriffe aus der Lernstichprobe ist, und ω_i das Gewicht des i-ten Terms ist.

Zu den beliebten Wiegemethoden eines Terms gehören [Manning u. a. (2008), Jurafsky und Martin (2009)]:

1. Das boolesche Gewicht - $\omega_i = 1$, wenn der Begriff im Dokument vorhanden ist, sonst 0.

2. Die Termhäufigkeit (Engl.: TF - term frequency) ist die Häufigkeit des Auftretens des Terms t in d.

$$\omega_i = tf(t_i, d) = n_i \tag{4.1}$$

3. TF-IDF (Engl.: IDF - inverse document frequency) ist das Produkt der Termhäufigkeit und der inversen Dokumenthäufigkeit.

$$\begin{aligned} idf(t_i, \mathbb{D}) &= \log \frac{|\mathbb{D}|}{|(d_i \supset t_i)|} \\ \omega_i = tfidf(t_i, d, \mathbb{D}) &= tf(t_i, d) \times idf(t_i, \mathbb{D}) \end{aligned} \tag{4.2}$$

wobei $|\mathbb{D}|$ die Anzahl der Dokumente im Korpus und $|(d_i \supset t_i)|$ die Anzahl der Dokumente, in denen t_i vorkommt, bezeichnen.

Diese Formel repräsentiert die Beobachtung, dass die wichtigen Begriffe eines Dokuments höhere Häufigkeit in diesem Dokument und niedrige Häufigkeit im gesamten Korpus haben.

4.2.3 Bestimmung der Qualität des Klassifikators

Um die Klassifikationsalgorithmen zu vergleichen, ist es notwendig, ein Maß für die Qualität ihrer Leistung einzuführen.

Dazu wird ein Trainingsset mit bekannten Klassen benötigt. Nach der Ausführung des trainierten Klassifikators auf der Teststichprobe werden für jede Klasse folgende Werte berechnet [Kohavi und Provost (1998)]:

- TP - die Anzahl der wirklich positiven Ergebnisse;
- TN - die Anzahl der wirklich negativen Ergebnisse;
- FP - die Anzahl der falsch klassifizierten positiven Ergebnisse;
- FN - die Anzahl der falsch klassifizierten negativen Ergebnisse;

Anhand dieser Werte werden die Genauigkeit (Engl.: precision) und die Vollständigkeit (Engl.: recall) berücksichtigt. Wir interpretieren diese Begriffe für die Bestimmung der Tonalität von Dokumenten folgendermaßen:

	klassifiziert als +	klassifiziert als -
Klasse +	TP	FN
Klasse -	FP	TN

Tabelle 4.1: Ergebnisse der binären Klassifikation

Sei N_p Dokumente in einer Sammlung von N Dokumenten mit einer positiven emotionalen Färbung (sie gehören zu der Klasse $+$) und N_n Dokumente mit einer negativen emotionalen Färbung (sie gehören zu der Klasse $-$) gegeben. Als Ergebnis der Klassifizierung dieser Dokumente werden TP Dokumente der Klasse $+$ korrekt zugewiesen und FP verhältnismäßig falsch zugewiesen. Ähnlich sind TN Dokumente der Klasse $-$ korrekt zugewiesen und FN entsprechend falsch zugewiesen. In Bezug auf die Klasse $+$ werden jetzt die Begriffe eingeführt:

Die **Genauigkeit** ist das Verhältnis der Anzahl der korrekt zur Klasse $+$ zugeordneten Dokumente zur Anzahl aller als $+$ klassifizierten Dokumente:

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

Die **Vollständigkeit** ist das Verhältnis der Anzahl der korrekt zur Klasse $+$ zugeordneten Dokumente zur Anzahl der Dokumente der Klasse $+$ in der Sammlung:

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

Ihre Bedeutung kann so formuliert werden: die Genauigkeit ist der Anteil der Ergebnisse, der wirklich zu einer bestimmten Klasse gehört, und die Vollständigkeit ist der Prozentsatz der gefundenen Ergebnisse von ihrer Gesamtanzahl.

Wenn einer dieser Werte gegen Null tendiert, verringert sich auch der Aussagewert des Klassifikators. Daher wird ein F_1-Maß für die Mittelung beider Werte definiert und zwar als der harmonische Mittelwert der Genauigkeit und der Vollständigkeit:

$$F_1 = \frac{2 \times (Precision + Recall)}{Precision \times Recall} \tag{4.5}$$

Im Gegensatz zum arithmetischen Mittel wird der Durchschnitt in diesem Fall 0 anstatt $\frac{1}{2}$, wenn $Precision = 0$ und $Recall = 1$ sind (oder umgekehrt).

Es kann auch ein Problem des *Überschreitens* (Engl.: overfitting) entstehen, wenn das Modell auf der Teststichprobe ausgezeichnet funktioniert, aber für echte Daten unzuverlässig ist. Deswegen wird eine *Kreuzvalidierungsmethode* (Engl.: cross-validation) verwendet, um das Modell zu validieren. Dafür wird die gesamte Trainingsprobe in k-Teile unterteilt, dann wird der Klassifikationsalgorithmus für alle $i \in \{1, 2, \ldots, k\}$ mit Ausnahme des Teils i auf der gesamten Trainingsprobe trainiert und im i-ten Teil anschließend getestet. Das Ergebnis des Verfahrens ist das arithmetische Mittel aller Durchläufe.

4.2.4 Die Auswahl des Klassifikationsalgorithmus

In einer Reihe von Studien zur Bestimmung der Tonalität des Textes haben die Lernmethoden mit dem Lehrer eine hohe Effizienz gezeigt. Diese Methoden werden sowohl in früheren Werken verwendet, um die durchschnittliche Tonalität des Dokuments zu bestimmen, als auch in modernen Werken, in denen die Sätze und kurze Textnachrichten analysiert werden. Um die eigentliche Aufgabe zu lösen, wurden zwei Algorithmen ausgewählt, die sich für die Bestimmung der Tonalität kurzer Textnachrichten als die effektivsten erwiesen haben. [Thelwall u. a. (2010), Thelwall u. a. (2011)]

Der naive Bayes-Klassifikator wird normalerweise als ein grundsätzliches und einfacheres Modell verwendet und die Methode der Stützvektoren wird als ein komplexeres Modell verwendet, da es als effizienter betrachtet wird. [Bermingham und Smeaton (2010),

Wang und Manning (2012)] zeigten jedoch, dass der naive Bayes-Klassifikator eine bessere Leistung im Vergleich zu der Methode der Stützvektoren in kürzeren Texten erzielt.

4.2.4.1 Die Methode der Stützvektoren

Die Methode der Stützvektoren gehört zur Familie der linearen Klassifikatoren.

Die Grundidee der Methode der Stützvektoren besteht darin, nach einer trennenden Hyperebene zu suchen, die so weit wie möglich von den nähesten Punkten im Merkmalsraum entfernt ist. [Tong und Koller (2002)]

Die Hyperebene ist ein Raum, dessen Dimension um eins kleiner ist, als die Dimension des ursprünglichen Raums. Die Vektoren, die der trennenden Hyperebene am nächsten sind, werden die Stützvektoren genannt.

Angenommen, es gibt eine Stichprobe:

$$(x_1, y_1), \ldots, (x_m, y_m), x_i \in \mathbb{R}^n, y_i \in \{-1, 1\} \tag{4.6}$$

Die Methode der Stützvektoren konstruiert die Klassifizierungsfunktion F in der Form $F(x) = sign(\langle w, x\rangle + b)$, wobei $\langle, \rangle$ das Skalarprodukt, w der Normalvektor zur trennenden Hyperebene und b ein Hilfsparameter ist. Jene Objekte, für die $F(x)$ gleich 1 ist, fallen in eine Klasse, und Objekte mit $F(x) = -1$ fallen in die andere. Die Wahl einer solchen Funktion ist nicht zufällig: jede Hyperebene kann in der Form $\langle w, x\rangle + b$ für bestimmte w und b dargestellt werden.

Als nächstes wollen wir w und b so wählen, dass der Abstand zu jeder Klasse maximiert wird. Die gegebene Entfernung ist $\frac{1}{\|w\|}$. Davon das Maximum zu finden, entspricht dem Problem, das Minimum von $\|w\|^2$ zu finden.

Im Falle eines linear trennbaren Satzes an Daten kann die Suche nach einer Hyperebene als Optimierungsproblem geschrieben werden:

$$\begin{aligned} &\|w\|^2 \to \min_{w,b} \\ &y_i(\langle w, x_i \rangle + b) \geq 1, \quad i = 1, \ldots, m \end{aligned} \tag{4.7}$$

Für einen allgemeineren Fall einer linear untrennbaren Stichprobe kann der Algorithmus den Fehler bei Lehrobjekten erzeugen. Ein neues Optimierungsproblem beinhaltet die Anforderung, den Fehler zu minimieren:

$$\begin{aligned} &\|w\|^2 + C \sum_{i=1}^{k} e_i \to \min_{w,b} \\ &y_i(\langle w, x_i \rangle + b) \geq 1 - e_i, \quad i = 1, \ldots, m \\ &e_i \geq 0, \quad i = 1, \ldots, m \end{aligned} \tag{4.8}$$

Die Variablen e_i kennzeichnen die Größe des Fehlers. Die Konstante C lässt einen Kompromiss zwischen der zu maximierenden Lücke und dem zu minimierenden Gesamtfehler in der Trainingsprobe finden.

4.2.4.2 Der naive Bayes-Klassifikator

Der naive Bayes-Klassifikator basiert auf dem Bayes-Theorem [Manning und Schütze (1999)]:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{4.9}$$

wobei

- $P(c|d)$ ist die bedingte Wahrscheinlichkeit, dass das Dokument d zur Klasse c gehört.

- $P(d|c)$ ist die bedingte Wahrscheinlichkeit, ein Dokument d unter allen Dokumenten der Klasse c zu treffen.

- $P(c)$ ist die a-priori-Wahrscheinlichkeit, ein Dokument der Klasse c unter allen Dokumenten zu treffen.

- $P(d)$ ist die a-priori-Wahrscheinlichkeit des Dokuments d im Korpus.

Als Ergebnis der Klassifikation wird eine Klasse mit einem maximalen Wahrscheinlichkeitswert ausgewählt:

$$c = \arg\max_{c \in \mathbb{C}} P(c|d) \tag{4.10}$$

Da die Verteilungsdichten meistens unbekannt sind, werden sie von der Trainingsstichprobe geschätzt. In diesem Fall ist die Bewertung der Wahrscheinlichkeit des Dokuments in der Stichprobe $P(d) = const$ und beeinflusst die Klassifikation daher nicht.

$$c = \arg\max_{c \in \mathbb{C}} P(d|c)P(c) \tag{4.11}$$

Der naive Bayes-Klassifikator basiert auf der zusätzlichen Annahme, dass die Klassifikationsmerkmale unabhängig sind, also für die bedingte Wahrscheinlichkeit des Dokuments $P(d|c) \approx P(w_1|c)P(w_2|c) \dots P(w_{|\mathbb{V}_d|}|c)$ gilt, wobei für jeden Term $w_i \in d$ gilt. Sei $\mathbb{V}_d$ der gesamte Wortschatz des Dokuments d. Dann gilt:

$$c = \arg\max_{c \in \mathbb{C}} P(c) \prod_{i=1}^{|\mathbb{V}_d|} P(w_i|c) \tag{4.12}$$

Um einen Überlauf von unten zu vermeiden, wird das Produkt normalerweise logarithmiert (dies ändert die numerischen Werte der probabilistischen Schätzungen, aber der Wert, bei dem das Maximum erreicht wird, bleibt erhalten).

$$c = \arg\max_{c \in \mathbb{C}} \left[\log P(c) + \sum_{i=1}^{|\mathbb{V}_d|} \log P(w_i|c) \right] \tag{4.13}$$

Die probabilistischen Schätzungen sind auf Basis einer Stichprobe $\mathbb{D}$ wie folgt definiert:

$$\begin{aligned} P(c) &= \frac{|\mathbb{D}_c|}{|\mathbb{D}|} \\ P(w_i|c) &= \frac{W_{ic}}{\sum_{j=1}^{|\mathbb{V}|} W_{jc}} \end{aligned} \tag{4.14}$$

wobei

- $|\mathbb{D}_c|$ die Anzahl der Dokumente $d \in \mathbb{D}$ der Klasse c ist,
- W_{ic} die Häufigkeit, mit der der i-te Term in Dokumenten der Klasse c vorkommt, und
- $\mathbb{V}$ der gesamte Wortschatz der Stichprobe $\mathbb{D}$ ist.

Die Formel 4.14 beschreibt das sogenannte *Multinomial*-Modell. [Manning u. a. (2008)]

Um das Problem mit den Nullwerten zu vermeiden, wenn ein noch unbekannter Term dem Klassifikator beim Anwenden vorliegt, wird eine Laplace-Glättung verwendet:

$$P(w_i|c) = \frac{W_{ic} + 1}{\sum_{j=1}^{|\mathbb{V}|}(W_{jc} + 1)} \tag{4.15}$$

Die resultierende Klassifizierungsformel lautet:

$$c = \arg\max_{c \in \mathbb{C}} \left[\log \frac{|\mathbb{D}_c|}{|\mathbb{D}|} + \sum_{i=1}^{|\mathbb{V}_d|} \log \frac{W_{ic} + 1}{\sum_{j=1}^{|\mathbb{V}|}(W_{jc} + 1)} \right] \tag{4.16}$$

In diesem Verfahren stellt ein Dokument einen $|d|$-dimensionalen Vektor von Termen dar, die genau in ihrer ursprünglichen Reihenfolge im Dokument vorkommen.

Es gibt auch ein anderes, sogenanntes *Bernoulli*-Modell [Manning u. a. (2008)] für die Berechnung von $P(w_i|c)$:

$$P(w_i|c) = \frac{|\mathbb{D}_{cw_i}| + 1}{|\mathbb{D}_c| + 2} \tag{4.17}$$

wobei

- $|\mathbb{D}_{cw_i}|$ die Anzahl der Dokumente $d \in \mathbb{D}$ der Klasse c, in denen der Term w_i vorkommt, und
- $|\mathbb{D}_c|$ die Anzahl der Dokumente $d \in \mathbb{D}$ der Klasse c ist.

In diesem Verfahren stellt ein Dokument einen $|\mathbb{V}|$-dimensionalen Vektor binärer Attribute dar, die angeben, ob ein Term aus dem gesamten Wortschatz $\mathbb{V}$ im Dokument gefunden wurde.

Das Bernoulli-Modell bringt das beste Ergebnis mit einem kleinen Vokabular und das Multinomial-Modell mit einem größeren Vokabular. Das Multinomial-Modell erreicht eine höhere Genauigkeit insgesamt. [McCallum und Nigam (1998)]

Trotz der Tatsache, dass die Annahme der Unabhängigkeit von Klassifikationsmerkmalen in natürlicher Sprache nicht wahr ist (die Bedeutung des Wortes hängt vom Kontext ab), zeigt der naive Bayes-Klassifikator häufig gute Ergebnisse in der Textklassifikation. [Pang u. a. (2002)]

4.3 Die Bewertung der Qualität von Algorithmen

Die Auswertung von Algorithmen wurde mit Hilfe von Tests an zwei markierten englischsprachigen Korpussen durchgeführt: [Wilson u. a. (2013)] und [Nakov u. a. (2016)]. Die neutralen Tweets wurden während der Vorbereitungsphase davon entfernt. Die beiden Korpusse umfassen die Nachrichten aus unterschiedlichen Bereichen.

Korpus	positiv	negativ	allgemein
SemEval13	3523	1337	4860
SemEval16	8485	3281	11766

Tabelle 4.2: Numerische Merkmale von Korpussen

4.3.1 Die Normalisierung von Daten

Bevor die Algorithmen bewertet werden, müssen die Daten in Terme segmentiert werden. Anschließend werden sie normalisiert, um die Dimension und das „Rauschen“ der Daten zu reduzieren.

Dafür wurden folgende Methoden verwendet:

1. da in modernen englischen Wörtern keine Wörter verwendet werden, in denen drei oder mehr identische Symbole aufeinander folgen, wurden alle solche Vorkommen durch zwei Symbole ersetzt („aaah“ wird in „aah“ transformiert);

2. alle Buchstaben wurden in Kleinbuchstaben transformiert;

3. alle Erwähnungen von Benutzern wurden entfernt;

4. alle Hyperlinks und E-Mail Adressen wurden entfernt;

5. alle gleichen aufeinander folgenden Terme wurden auf einen Term reduziert (eine Liste von Termen ['a', 'a', 'a'] wird in ['a'] transformiert;

6. alle Emoticons[6] und ihre äquivalenten Emoji[7] Zeichen wurden mit dem Term „p_emotion“ oder „n_emotion“ ersetzt, abhängig von der Tonalität (siehe Tabelle 4.3).

7. alle Hashtags, Satz- und Sonderzeichen und Terme mit Satz- oder Sonderzeichen am Anfang wurden entfernt (Ausnahme: Frage- und Ausrufezeichen);

[6] https://en.wikipedia.org/wiki/List_of_emoticons (09.04.2018)
[7] https://en.wikipedia.org/wiki/Emoji (09.04.2018)

Tonalität	Emoticons
positive Emoticons	:-) :) :-] :] :-3 :3 :-> :> 8-) 8) :-} :} :o) :c) :^) =] =) :-D :D 8-D 8D x-D xD X-D XD =D =3 B^D :'-) :') :-* :* :x ;-) ;) *-) *) ;-] ;] ;^) :-, ;D :-P :P X-P XP x-p xp :-p :p :-Þ :Þ :-þ :þ :-b :b d: =p >:P O:-) O:) 0:-3 0:3 0:-) 0:) 0;^) |;-) <3 \o/ ^5
negative Emoticons	:-(:(:-c :c :-< :< :-[:[:-|| >:[:{ :@ >:(:'-(:'(D-': D:< D: D8 D; D= DX :-O :O :-o :o :-0 8-0 >:O O_O o-o O_o o_O o_o O-O :-/ :/ :-. >:\ >:/ :\ =/ =\ :L =L :S :-| :| :$:-X :X :-# :# :-& :& >:-) >:) }:-) }:) 3:-) 3:) >;) |-O %-) %) :# :-# ',:-| ',:-l </3 <\3 >.<

Tabelle 4.3: Emotionale Färbung von Emoticons

8. alle Terme mit den gleichen aufeinander folgenden Zeichen wurden auf Vorhandensein im WordNet-Lexikon (siehe 3.4.2) überprüft. Wenn für solchen Term kein Treffer gefunden wurde, wurden die duplizierten Zeichen auf nur ein Zeichen reduziert;

9. Wenn der Term auf „'s“ endet, wurde solche Endung entfernt;

10. alle Terme, die entweder mit Zahlen oder mit „n“, „nr“, „no“, „x“ + Zahl beginnen, wurden entfernt;

11. alle Terme, die ausschließlich aus einem Buchstaben bestehen, wurden entfernt;

12. alle Terme wurden auf Vorhandensein in einer manuell erweiterten Liste von Stoppwörtern[8] überprüft. Die zusammenstimmenden Terme wurden entfernt.

In dieser Weise haben wir die Anzahl der Merkmale für die Umsetzung von Algorithmen deutlich verringert.

Korpus	vor Normalisierung		nach Normalisierung	
	Monogramme	Bigramme	Monogramme	Bigramme
SemEval13	18692	67695	10452	34748
SemEval16	30850	132784	15088	73525

Tabelle 4.4: Anzahl der N-Gramme nach Normalisierung

Zusätzlich werden auch zwei Methoden als alternative Vorgehensweise betrachtet:

1. Bei dem ersten *Stammformreduktion*-Verfahren werden die Terme nach den oben genannten Schritten regelbasiert auf ihre Wortstämme zurückgeführt. Das heißt, die Suffixe werden somit verändert und verkürzt. [Porter (2006)]

2. Das andere ähnliche *Lemmatisierung*-Verfahren bedeutet die Reduktion der Terme auf ihre Grundformen, also Lemmata, mithilfe von einer lexikalischen Datenbank. [Princeton (2010)]

Die beiden Verfahren dienen dem gleichen Zweck, die Dimension der Daten zu reduzieren. Wenn die gleichartigen Wörter mit unterschiedlichen Endungen im Text gefunden werden, können sie auf die gleiche Form zurückgeführt werden.

Die Genauigkeit der Stimmungsanalyse kann aber in gewissen Fällen negativ beeinflusst werden. Die Grundformen von Termen verlieren die morphologische Information, die für die Analyse der Tonalität nützlich sein kann. Die Tonalität von „ich möchte ein Haus kaufen“ ist höchstwahrscheinlich positiv, weil hier die Hoffnung ausgedrückt wurde. Die Tonalität des Satzes in Vergangenheit kann dagegen negativ sein, weil somit Bedauern ausgedrückt wird.

[8] https://www.ranks.nl/stopwords (09.04.2018)

Korpus	Norm. zzgl. Stammformreduktion		Norm. zzgl. Lemmatisierung	
	Monogramme	Bigramme	Monogramme	Bigramme
SemEval13	8509	34102	9038	34207
SemEval16	11445	71092	12598	71466

Tabelle 4.5: Anzahl der N-Gramme nach Normalisierung zzgl. Stammformreduktion oder Lemmatisierung

4.3.2 Die Tests

Die Tests wurden mit Hilfe einer 10-fachen Kreuzvalidierungsmethode durchgeführt. Zunächst wurden die Ergebnisse der Iterationen gemittelt und anschließend für die Klassen gemittelt.

Die Testverfahren untersuchten verschiedene Kombinationen von Gewichtungsmethoden (boolesche Gewichte, Termhäufigkeit, TF-IDF), Merkmalen (Monogramme, Bigramme) und Algorithmen (Bernoulli und Multinomial Naive Bayes Klassifikatoren; LinearSVC als eine Implementation der Methode der Stützvektoren). Zum Vergleich wurde auch die lexikonbasierte Methode verwendet [Caren (2012)].

Der Übersichtlichkeit und Nutzerfreundlichkeit halber wurden alle Tests im *Jupyter Notebook*[9] durchgeführt. Dafür wurden zusätzliche Bibliotheken *pandas*,[10] *NumPy*,[11] *NLTK*[12] und *scikit-learn*[13] verwendet.

Das *Jupyter Notebook* ist eine populäre Open-Source-Webanwendung, die häufig zur Datenbereinigung, Datentransformation und Datenvisualisierung verwendet wird.

Die Abbildung 4.2 repräsentiert das Arbeitsfenster von Jupyter Notebook im Laufe der Datenverarbeitung.

[9] http://jupyter.org (09.04.2018)
[10] https://pandas.pydata.org (09.04.2018)
[11] http://www.numpy.org (09.04.2018)
[12] http://www.nltk.org (09.04.2018)
[13] http://scikit-learn.org (09.04.2018)

Abbildung 4.2: Jupyter Notebook

Alle Analysestrategien wurden in mehrere Dateien aufgeteilt, die sowohl den Code-, als auch die Textbeschreibung aller Schritte zusammenfassen.

Die Abbildung 4.3 repräsentiert ein überwachtes Lernverfahren im Gebrauch:

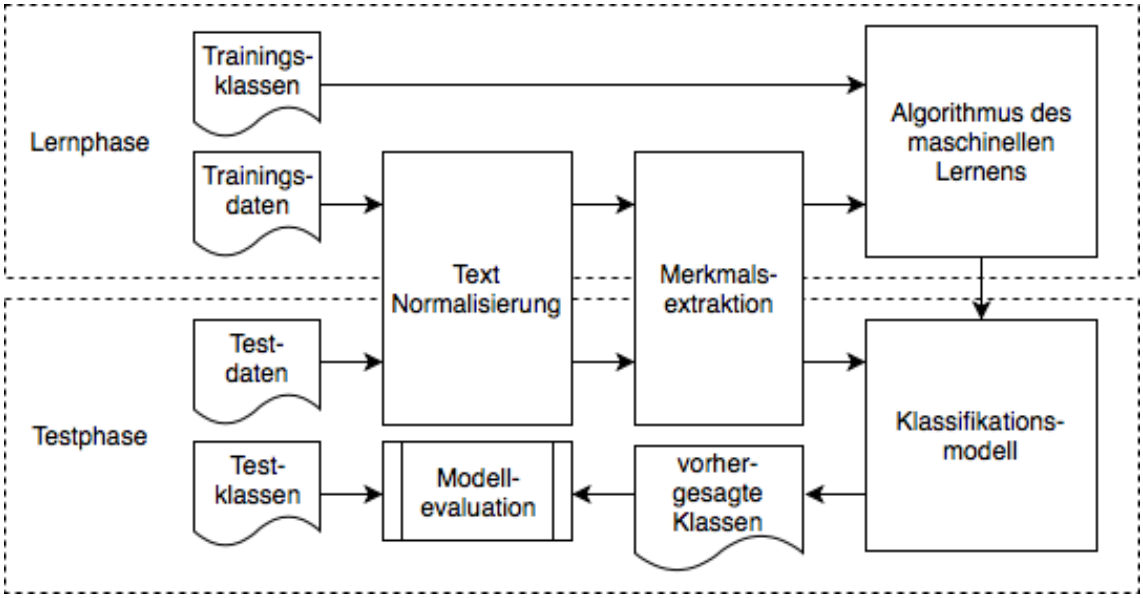

Abbildung 4.3: überwachtes Lernverfahren

Die Ergebnisse der Experimente sind den Tabellen 4.6, 4.7 und 4.8 zu entnehmen.

Als Bezeichnungen gelten:

- P entspricht der Genauigkeit, R stellt die Vollständigkeit dar und F_1 steht für F_1-Metrik (Siehe Abschnitt 4.2.3);
- *Bool* - boolesche Gewichtung;
- *TF* - Termhäufigkeit;
- *TF-IDF* - das Produkt der Termhäufigkeit und der inversen Dokumenthäufigkeit.

Die praktischen Tests zeigen auf, dass die beiden Algorithmen des maschinellen Lernens deutlich bessere Stimmungsanalyse gegenüber der lexikonbasierten Methode erzielen. Im Allgemeinen wurde die beste Bestimmung der Polarität getroffen, wenn eine Komposition

von Monogrammen und Bigrammen als Merkmale benutzt wurde. Dazu hat die Methode der Stützvektoren im Schnitt um 2-3 Prozent bessere Resultate erbracht.

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
BernoulliNB	1	Bool	0,73	0,74	**0,74**	0,73	0,74	**0,73**
	2	Bool	0,61	0,63	0,61	0,63	0,65	0,63
	1+2	Bool	0,7	0,73	0,71	0,7	0,73	0,71
MultinomialNB	1	TF	0,75	0,74	0,75	0,73	0,73	**0,73**
	2	TF	0,63	0,65	0,55	0,62	0,65	0,58
	1+2	TF	0,73	0,77	**0,75**	0,7	0,73	0,71
MultinomialNB	1	TF-IDF	0,74	0,68	0,69	0,75	0,72	**0,73**
	2	TF-IDF	0,64	0,67	0,63	0,63	0,66	0,63
	1+2	TF-IDF	0,74	0,72	**0,73**	0,73	0,73	0,73
LinearSVC	1	Bool	0,77	0,78	0,77	0,73	0,76	0,74
	2	Bool	0,73	0,55	0,53	0,61	0,57	0,57
	1+2	Bool	0,79	0,77	**0,78**	0,75	0,75	**0,75**
	1	TF	0,77	0,77	0,77	0,73	0,76	0,74
	2	TF	0,73	0,55	0,53	0,61	0,57	0,57
	1+2	TF	0,79	0,76	**0,77**	0,74	0,75	**0,74**
	1	TF-IDF	0,76	0,79	0,77	0,73	0,77	0,74
	2	TF-IDF	0,74	0,55	0,53	0,61	0,57	0,58
	1+2	TF-IDF	0,78	0,78	**0,78**	0,74	0,77	**0,75**
Lexikon	-	-	0,73	0,49	0,58	0,71	0,46	0,56

Tabelle 4.6: Ergebnisse von Algorithmen nach Normalisierung

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
BernoulliNB	1	Bool	0,74	0,75	**0,75**	0,73	0,74	**0,73**
	2	Bool	0,61	0,64	0,61	0,64	0,66	0,64
	1+2	Bool	0,71	0,74	0,72	0,70	0,73	0,71
MultinomialNB	1	TF	0,77	0,76	**0,76**	0,74	0,74	**0,74**
	2	TF	0,63	0,66	0,57	0,63	0,66	0,59
	1+2	TF	0,75	0,78	0,76	0,71	0,74	0,72
MultinomialNB	1	TF-IDF	0,75	0,69	0,70	0,75	0,72	0,73
	2	TF-IDF	0,64	0,68	0,63	0,64	0,67	0,64
	1+2	TF-IDF	0,75	0,73	**0,74**	0,74	0,73	**0,74**
LinearSVC	1	Bool	0,77	0,79	0,78	0,74	0,77	0,75
	2	Bool	0,73	0,56	0,54	0,62	0,58	0,59
	1+2	Bool	0,80	0,78	**0,79**	0,76	0,76	**0,76**
	1	TF	0,77	0,79	0,78	0,73	0,76	0,75
	2	TF	0,73	0,56	0,54	0,62	0,58	0,58
	1+2	TF	0,80	0,78	**0,79**	0,76	0,76	**0,76**
	1	TF-IDF	0,77	0,80	0,78	0,75	0,78	0,75
	2	TF-IDF	0,74	0,55	0,53	0,62	0,59	0,59
	1+2	TF-IDF	0,78	0,79	**0,79**	0,75	0,78	**0,76**

Tabelle 4.7: Ergebnisse von Algorithmen nach Normalisierung zzgl. Stammreduktion

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
BernoulliNB	1	Bool	0,74	0,75	**0,75**	0,73	0,74	**0,73**
	2	Bool	0,61	0,63	0,61	0,64	0,66	0,64
	1+2	Bool	0,70	0,73	0,71	0,70	0,73	0,71
MultinomialNB	1	TF	0,76	0,75	**0,76**	0,73	0,74	**0,74**
	2	TF	0,63	0,65	0,56	0,63	0,66	0,59
	1+2	TF	0,74	0,78	0,76	0,71	0,73	0,72
MultinomialNB	1	TF-IDF	0,74	0,69	0,70	0,75	0,72	**0,73**
	2	TF-IDF	0,64	0,68	0,63	0,64	0,67	0,64
	1+2	TF-IDF	0,74	0,72	**0,73**	0,74	0,73	0,73
LinearSVC	1	Bool	0,77	0,78	0,78	0,74	0,76	0,75
	2	Bool	0,73	0,56	0,54	0,62	0,58	0,59
	1+2	Bool	0,80	0,77	**0,78**	0,76	0,76	**0,76**
	1	TF	0,77	0,78	0,77	0,73	0,76	0,75
	2	TF	0,73	0,56	0,54	0,62	0,58	0,59
	1+2	TF	0,80	0,77	**0,79**	0,75	0,76	**0,75**
	1	TF-IDF	0,77	0,79	0,78	0,74	0,78	0,75
	2	TF-IDF	0,73	0,55	0,53	0,62	0,59	0,59
	1+2	TF-IDF	0,78	0,78	**0,78**	0,75	0,77	**0,76**

Tabelle 4.8: Ergebnisse von Algorithmen nach Normalisierung zzgl. Lemmatisierung

4.4 Verbesserung der Genauigkeit der Algorithmen

Es gibt mehrere Möglichkeiten, die Qualität von maschinellen Lernalgorithmen zu verbessern. Hier folgen einige häufig verwendete Verfahren:

4.4.1 Auswahl der Merkmale

Die Auswahl der Merkmale wird hauptsächlich verwendet, um die Dimensionalität des Merkmalsraums zu reduzieren. Da es aber ein gewisses „Rauschen“ in den Daten gibt (sehr seltene oder sehr häufige Wörter, die in allen Klassen vorkommen), kann eine Verringerung der Dimension die Qualität und die Geschwindigkeit der Algorithmen verbessern.

Die Merkmale können mit Hilfe von folgenden Methoden ausgewählt werden: [Ikonomakis u. a. (2005)]

1. Basierend auf der Dokumenthäufigkeit. Die Idee liegt darin, dass wir selten verwendete Terme aus dem Merkmalsraum ausschließen. Wenn die Anzahl der Dokumente, in denen der Term vorkommt, niedriger als m (oder Prozentsatz von dem gesamten Korpus) ist, wird solcher Term nicht mehr bearbeitet. Die Zahl m wird experimentell ermittelt.

   Analog können auch die Terme ausgeschlossen werden, die sehr häufig in Dokumenten vorkommen.

2. Basierend auf gegenseitiger Information (Engl.: mutual information) [Manning u. a. (2008)]. Der Wert der gegenseitigen Information zeigt die Beziehung zwischen einem Merkmal und einer Klasse. Für jede Klasse werden m Merkmale mit den höchsten Werten genommen.

   Um den Wert der gegenseitigen Information zu berechnen, müssen folgende Werte für jeden Term t und für jede Klasse c berechnet werden:

- $N_{1,1}$ - wie oft tritt der Term t in Dokumenten der Klasse c auf.
- $N_{0,1}$ - wie oft werden andere Terme in Dokumenten der Klasse c gefunden.
- $N_{1,0}$ - wie oft tritt der Term t in Dokumenten anderer Klassen auf.
- $N_{0,0}$ - wie oft werden andere Terme in Dokumenten anderer Klassen gefunden.
- $N = N_{1,1} + N_{0,1} + N_{1,0} + N_{0,0}$

Dann wird der Wert der gegenseitigen Information nach der Formel berechnet:

$$M_{t,c} = \frac{N_{1,1}}{N} \log \frac{NN_{1,1}}{N_{1,.}N_{.,1}} + \frac{N_{0,1}}{N} \log \frac{NN_{0,1}}{N_{0,.}N_{.,1}} + \frac{N_{1,0}}{N} \log \frac{NN_{1,0}}{N_{1,.}N_{.,0}} + \frac{N_{0,0}}{N} \log \frac{NN_{0,0}}{N_{0,.}N_{.,0}} \tag{4.18}$$

Ein Index mit dem Punkt bedeutet, dass die Summe aus den Werten 0 und 1 an dieser Stelle $(N_{1,.} = N_{1,1} + N_{1,0})$ gebildet wird.

3. Basierend auf der Delta-TF-IDF-Metrik[14] [Martineau und Finin (2009)]. Die Idee liegt darin, dass wir die Bedeutung von Termen verstärken, die eine nicht-neutrale Tonalität haben. Je ungleichmäßiger ein Term zwischen den positiven und negativen Klassen verteilt ist, desto höherer Wert hat er. Die gleichmäßig verteilten Terme werden im Gegensatz gar nicht berücksichtigt.

   Der Wert von Delta-TF-IDF wird wie folgt berechnet:

$$w_{t,d} = tf(t,d) \times \left(\log \frac{|P|}{P_t} - \log \frac{|N|}{N_t} \right) \tag{4.19}$$

   dabei ist:

   - $tf(t,d)$ - die Anzahl der Fälle, in denen der Begriff t im Dokument d vorkommt,
   - P - eine Menge von Dokumenten aus der Trainingsstichprobe, deren Klasse positiv ist,

[14] verwendet https://github.com/r-m-n/sklearn-deltatfidf (09.04.2018)

- P_t - die Anzahl der Dokumente aus P, in denen der Term t vorkommt,
- N - eine Menge von Dokumenten aus der Trainingsstichprobe, deren Klasse negativ ist,
- N_t - die Anzahl der Dokumente aus N, in denen der Term t vorkommt.

Die oben genannten Verfahren für die Merkmalsauswahl wurden weiteren Tests unterzogen. Die Tests wurden unter den im Abschnitt 4.3.2 genannten Umständen durchgeführt.

Die Testverfahren untersuchten verschiedene Kombinationen von Gewichtungsmethoden analog zum Abschnitt 4.3.2.

Die Ergebnisse sind den Tabellen 4.9, 4.10, 4.11, 4.12, 4.13 und 4.14 zu entnehmen.

Als zusätzliche zu den im Abschnitt 4.3.2 erwähnten Bezeichnungen gelten:

- *DeltaTF-IDF* - Delta-TF-IDF-Metrik;
- *DFL* - Beschränkung der Dokumenthäufigkeit;
- *MI* - gegenseitige Information.

Im Laufe der Experimente wurde die größte Genauigkeitssteigerung durch den Einsatz der gegenseitigen Information erzielt. Die beiden Algorithmen des maschinellen Lernens haben stark davon profitiert. Im Fall des naiven Bayes-Klassifikators wurde die Genauigkeit um mehr als 9 Prozent gestiegen. Die effektivste Konfiguration in Bezug auf den F_1-Wert ergab sich aus dem Bernoulli-Modell, das auf Monogramme trainiert wurde. Das resultierende trainierte Modell wurde anschließend zur späteren Verwendung gespeichert.

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
BernoulliNB	1	Bool + DFL	0,75	0,74	**0,75**	0,73	0,74	**0,74**
	2	Bool + DFL	0,63	0,61	0,61	0,59	0,59	0,59
	1+2	Bool + DFL	0,74	0,75	0,74	0,70	0,73	0,70
MultinomialNB	1	TF + DFL	0,76	0,76	**0,76**	0,73	0,75	**0,73**
	2	TF + DFL	0,63	0,60	0,61	0,58	0,59	0,59
	1+2	TF + DFL	0,75	0,77	0,76	0,70	0,73	0,70
BernoulliNB	1	Bool + MI	0,91	0,87	**0,89**	0,81	0,82	0,82
	2	Bool + MI	0,88	0,70	0,73	0,75	0,71	0,72
	1+2	Bool + MI	0,91	0,86	0,88	0,81	0,83	**0,82**
MultinomialNB	1	TF + MI	0,89	0,85	0,87	0,80	0,81	0,80
	2	TF + MI	0,82	0,59	0,58	0,75	0,70	0,72
	1+2	TF + MI	0,88	0,87	**0,87**	0,80	0,82	**0,81**
MultinomialNB	1	TF-IDF + MI	0,88	0,82	0,84	0,81	0,80	0,80
	2	TF-IDF + MI	0,85	0,63	0,64	0,75	0,69	0,71
	1+2	TF-IDF + MI	0,88	0,84	**0,86**	0,82	0,83	**0,82**
MultinomialNB	1	DeltaTF-IDF	0,76	0,69	0,71	0,75	0,72	**0,73**
	2	DeltaTF-IDF	0,63	0,62	0,63	0,60	0,61	0,60
	1+2	DeltaTF-IDF	0,75	0,71	**0,73**	0,72	0,73	0,72

Tabelle 4.9: Ergebnisse von Algorithmen nach Normalisierung zzgl. Merkmalsauswahl

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
LinearSVC	1	Bool + DFL	0,77	0,78	0,77	0,73	0,76	0,73
	2	Bool + DFL	0,65	0,68	0,63	0,60	0,62	0,57
	1+2	Bool + DFL	0,77	0,78	**0,78**	0,73	0,76	**0,74**
	1	TF + DFL	0,76	0,78	0,76	0,72	0,75	0,73
	2	TF + DFL	0,65	0,68	0,63	0,60	0,62	0,57
	1+2	TF + DFL	0,77	0,78	**0,77**	0,73	0,76	**0,74**
	1	Bool + MI	0,83	0,83	0,84	0,77	0,81	0,78
	2	Bool + MI	0,82	0,62	0,63	0,68	0,66	0,67
	1+2	Bool + MI	0,84	0,85	**0,84**	0,78	0,81	**0,79**
	1	TF + MI	0,83	0,80	0,81	0,77	0,80	0,78
	2	TF + MI	0,82	0,59	0,59	0,67	0,65	0,65
	1+2	TF + MI	0,83	0,81	**0,82**	0,78	0,80	**0,79**
	1	TF-IDF + MI	0,82	0,85	0,83	0,77	0,81	0,78
	2	TF-IDF + MI	0,80	0,59	0,59	0,70	0,67	0,68
	1+2	TF-IDF + MI	0,83	0,86	**0,84**	0,77	0,82	**0,79**
	1	DeltaTF-IDF	0,76	0,79	0,77	0,73	0,76	0,73
	2	DeltaTF-IDF	0,65	0,68	0,63	0,59	0,61	0,59
	1+2	DeltaTF-IDF	0,77	0,79	**0,77**	0,72	0,76	**0,74**

Tabelle 4.10: Ergebnisse von Algorithmen nach Normalisierung zzgl. Merkmalsauswahl

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
BernoulliNB	1	Bool + DFL	0,77	0,76	**0,76**	0,73	0,75	**0,74**
	2	Bool + DFL	0,63	0,61	0,62	0,59	0,61	0,59
	1+2	Bool + DFL	0,75	0,76	0,76	0,70	0,74	0,71
MultinomialNB	1	TF + DFL	0,77	0,77	0,77	0,73	0,75	**0,74**
	2	TF + DFL	0,63	0,61	0,61	0,59	0,60	0,60
	1+2	TF + DFL	0,77	0,78	**0,77**	0,70	0,73	0,71
BernoulliNB	1	Bool + MI	0,90	0,87	0,88	0,81	0,81	**0,81**
	2	Bool + MI	0,88	0,71	0,74	0,76	0,72	0,74
	1+2	Bool + MI	0,92	0,87	**0,89**	0,80	0,82	0,81
MultinomialNB	1	TF + MI	0,88	0,87	0,87	0,79	0,81	0,80
	2	TF + MI	0,83	0,61	0,62	0,76	0,72	0,73
	1+2	TF + MI	0,89	0,88	**0,88**	0,79	0,82	**0,80**
MultinomialNB	1	TF-IDF + MI	0,86	0,82	0,84	0,80	0,78	0,79
	2	TF-IDF + MI	0,86	0,64	0,66	0,77	0,72	0,74
	1+2	TF-IDF + MI	0,89	0,84	**0,86**	0,82	0,82	**0,82**
MultinomialNB	1	DeltaTF-IDF	0,79	0,71	0,73	0,75	0,72	**0,74**
	2	DeltaTF-IDF	0,63	0,62	0,63	0,61	0,62	0,62
	1+2	DeltaTF-IDF	0,78	0,73	**0,75**	0,72	0,72	0,72

Tabelle 4.11: Egebnisse von Algorithmen nach Normalisierung zzgl. Stammreduktion und Merkmalsauswahl

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
LinearSVC	1	Bool + DFL	0,77	0,79	0,78	0,73	0,77	0,75
	2	Bool + DFL	0,65	0,69	0,64	0,60	0,62	0,58
	1+2	Bool + DFL	0,78	0,80	**0,79**	0,74	0,77	**0,75**
	1	TF + DFL	0,77	0,79	0,78	0,73	0,76	0,74
	2	TF + DFL	0,65	0,69	0,64	0,60	0,62	0,58
	1+2	TF + DFL	0,78	0,79	**0,79**	0,74	0,77	**0,75**
	1	Bool + MI	0,83	0,84	0,83	0,77	0,80	0,79
	2	Bool + MI	0,82	0,63	0,64	0,69	0,68	0,69
	1+2	Bool + MI	0,84	0,85	**0,85**	0,79	0,81	**0,79**
	1	TF + MI	0,83	0,82	0,83	0,77	0,79	0,77
	2	TF + MI	0,82	0,60	0,60	0,69	0,66	0,68
	1+2	TF + MI	0,85	0,83	**0,83**	0,78	0,80	**0,79**
	1	TF-IDF + MI	0,83	0,86	0,84	0,77	0,81	0,78
	2	TF-IDF + MI	0,80	0,60	0,60	0,71	0,69	0,69
	1+2	TF-IDF + MI	0,84	0,87	**0,85**	0,78	0,82	**0,79**
	1	DeltaTF-IDF	0,77	0,80	0,78	0,73	0,77	0,74
	2	DeltaTF-IDF	0,65	0,69	0,65	0,60	0,61	0,60
	1+2	DeltaTF-IDF	0,78	0,80	**0,79**	0,74	0,77	**0,75**

Tabelle 4.12: Egebnisse von Algorithmen nach Normalisierung zzgl. Stammreduktion und Merkmalsauswahl

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
BernoulliNB	1	Bool + DFL	0,76	0,75	**0,76**	0,73	0,75	**0,73**
	2	Bool + DFL	0,63	0,61	0,62	0,59	0,60	0,60
	1+2	Bool + DFL	0,75	0,76	0,75	0,70	0,73	0,71
MultinomialNB	1	TF + DFL	0,77	0,76	**0,76**	0,73	0,75	**0,74**
	2	TF + DFL	0,63	0,61	0,61	0,59	0,60	0,60
	1+2	TF + DFL	0,76	0,77	0,76	0,70	0,73	0,70
BernoulliNB	1	Bool + MI	0,90	0,87	0,88	0,81	0,82	0,81
	2	Bool + MI	0,88	0,71	0,74	0,76	0,72	0,74
	1+2	Bool + MI	0,91	0,86	**0,88**	0,81	0,82	**0,81**
MultinomialNB	1	TF + MI	0,87	0,86	0,86	0,80	0,81	**0,80**
	2	TF + MI	0,83	0,61	0,62	0,76	0,72	0,73
	1+2	TF + MI	0,88	0,86	**0,87**	0,79	0,82	0,80
MultinomialNB	1	TF-IDF + MI	0,85	0,81	0,82	0,80	0,79	0,79
	2	TF-IDF + MI	0,86	0,64	0,66	0,77	0,72	0,73
	1+2	TF-IDF + MI	0,88	0,84	**0,86**	0,81	0,82	**0,82**
MultinomialNB	1	DeltaTF-IDF	0,78	0,70	0,72	0,75	0,72	**0,73**
	2	DeltaTF-IDF	0,63	0,62	0,63	0,61	0,61	0,61
	1+2	DeltaTF-IDF	0,77	0,73	**0,74**	0,73	0,72	0,72

Tabelle 4.13: Egebnisse von Algorithmen nach Normalisierung zzgl. Lemmatisierung und Merkmalsauswahl

Klassifikator			Korpus					
Algorithm	N-Gramme	Gewicht	SemEval13			SemEval16		
			P	R	F_1	P	R	F_1
LinearSVC	1	Bool + DFL	0,77	0,78	0,77	0,73	0,76	0,74
	2	Bool + DFL	0,65	0,69	0,64	0,60	0,62	0,58
	1+2	Bool + DFL	0,78	0,79	**0,78**	0,74	0,77	**0,75**
	1	TF + DFL	0,77	0,78	0,77	0,73	0,76	0,74
	2	TF + DFL	0,65	0,69	0,64	0,60	0,62	0,58
	1+2	TF + DFL	0,78	0,79	**0,78**	0,74	0,77	**0,75**
	1	Bool + MI	0,83	0,84	0,83	0,78	0,81	0,79
	2	Bool + MI	0,82	0,63	0,64	0,69	0,67	0,68
	1+2	Bool + MI	0,84	0,85	**0,84**	0,79	0,81	**0,79**
	1	TF + MI	0,83	0,81	0,81	0,77	0,79	0,78
	2	TF + MI	0,82	0,60	0,60	0,69	0,66	0,67
	1+2	TF + MI	0,84	0,82	**0,83**	0,78	0,81	**0,79**
	1	TF-IDF + MI	0,82	0,85	0,83	0,77	0,81	0,78
	2	TF-IDF + MI	0,80	0,60	0,60	0,71	0,69	0,69
	1+2	TF-IDF + MI	0,83	0,86	**0,84**	0,78	0,82	**0,79**
	1	DeltaTF-IDF	0,76	0,79	0,78	0,73	0,77	0,74
	2	DeltaTF-IDF	0,65	0,68	0,64	0,60	0,61	0,61
	1+2	DeltaTF-IDF	0,77	0,79	**0,78**	0,74	0,77	**0,75**

Tabelle 4.14: Egebnisse von Algorithmen nach Normalisierung zzgl. Lemmatisierung und Merkmalsauswahl

4.4.2 Verwendung von kombinierten Modellen

Die Idee der Verwendung kombinierter Modelle (eine Zusammensetzung der Klassifikatoren) besteht darin, dass mehrere einfachere Modelle genauere Ergebnisse liefern können als ein komplexes Modell. [Bell u. a. (2009), Koren (2009), Piotte und Chabbert (2009), Töscher u. a. (2009)]

Eine der beliebtesten Methoden (Engl.: bagging) besteht darin, dass viele Modelle an den Teilen der Trainingsstichprobe trainiert werden. Wenn die resultierenden Modelle danach zur Klassifizierung verwendet werden, werden die Ergebnisse gemittelt oder das beliebteste Ergebnis erzielt. Bei der Mittelwertbildung können unterschiedliche Gewichte für Klassifikatoren verwendet werden.

4.4.3 Verwendung zusätzlicher Merkmale

Als Klassifizierungsmerkmale können auch andere Konstrukte gebraucht werden:

- Informationen zu den Wortarten. Zum Beispiel in [Thelwall u. a. (2011)] wird ein Wörterbuch der Adjektive und Adverbien zusammengesetzt, weil diese Begriffe die Emotionen bezeichnen.

- Andere Sprachmodelle (nicht N-Gramme). [Bengio (2008), Cambria und Hussain (2012)]

- Informationen aus den Hashtags. [Davidov u. a. (2010a)]

- Semantik von Wörtern. Beispielsweise kann das Wort „weiß“ durch „hell“ ersetz werden, da sie ähnliche Bedeutung haben. Solche semantisch ähnliche Wörter umfassen zum Beispiel Synonyme, Hyponyme, Hyperonyme usw. Somit würden wir jedes Wort durch die Nummer seiner semantischen Gruppe ersetzen. Am Ende gäbe es ein „Bag-of-Words“-Modell, aber von einer tieferen Bedeutung. Dazu kann die Word2Vec-Technologie[15] verwendet werden.

[15] https://code.google.com/archive/p/word2vec/ (09.04.2018)

5 Softwareimplementierung

Für den praktischen Einsatz des im Abschnitt 4.4.1 erwähnten Algorithmus wurde eine Webanwendung entwickelt. Deren Ziel besteht darin, dass die Interessierenden nach bestimmten Themen in Twitter suchen können. Darüber hinaus werden die Suchergebnisse analysiert, und anschließend werden deren Stimmungen extrahiert und automatisiert bewertet.

5.1 Eingesetzte Technologien

Das gesamte Projekt wurde in den Programmiersprachen Python[16] und PHP[17] implementiert. Diese beiden Sprachen wurden aus folgenden Gründen ausgewählt:

1. Die zu lösende Aufgabe erfordert keine hohen Leistungsanforderungen.
2. Die beiden Programmiersprachen sind plattformübergreifend und objektorientiert.
3. PHP ist die meist verbreitete Programmiersprache für die Webanwendungen.
4. Der Algorithmus zur automatischen Klassifikation von Texten in positive und negative Meinungen wurde im letzten Kapitel schon in Python implementiert.

[16] https://www.python.org (09.04.2018)
[17] http://www.php.net (09.04.2018)

5.2 Applikationsarchitektur

Die Software-Implementierung besteht aus drei Komponenten:

1. Das Paket „Klassifikator“ dient der Stimmungsanalyse von Tweets.

2. Das Twitter Wrapper Paket dient vor allem dem Herunterladen von Tweets. Darüber hinaus gehört die Datenübergabe an „Klassifikator“ und Webanwendung zu deren Verantwortung.

3. Die Webanwendung „Sentiment“ ermöglicht die clientseitige Interaktion übers Internet. Zu deren Aufgaben gehört die Verarbeitung der Client-Eingaben und die Ergebnisdarstellung.

5.2.1 Klassifikator

Das Paket „Klassifikator“ ist für die Vorverarbeitung der Eingangsdaten, deren Bereinigung, Merkmalsauswahl und letztendliche Klassifizierung von Texten in positive und negative Meinungen zuständig.

Das Paket wurde in Python implementiert und enthält eine „Preprocessor“-Klasse (siehe Appendix A) und ein im Voraus trainiertes Modell (siehe Appendix B).

Die „Preprocessor“-Klasse führt die Normalisierung der Daten durch und extrahiert die Merkmale zur Stimmungsanalyse. Das Modell wurde auf den beiden Korpussen (siehe Abschnitt 4.3) gemeinsam trainiert. Solches Verfahren ermöglicht die sofortige Bestimmung der Tonalität von Eingabedaten.

Des Weiteren werden die Eingabedaten genau so verarbeitet, wie die Korpusse während der Normalisierung (siehe Abschnitt 4.3.1). Die Vorverarbeitung von Tweets wurde mit Hilfe von *NLTK*-Bibliothek implementiert. Um das Problem der Stimmungsanalyse zu lösen, wurde die Bibliothek *scikit-learn* benutzt. Der Verlauf und jeweilige Referenzen sind dem Abschnitt 4.3.2 zu entnehmen.

5.2.2 Twitter API

Das Twitter API[18] Wrapper Paket enthält eine Wrapper-Klasse für den „Twitter“ API Zugriff (siehe Appendix C).

An dieser Stelle findet die Authentifizierung bei Twitter statt. Danach werden die Tweets zum eingegebenen Thema abgefragt und deren Treffer an „Klassifikator“ Komponente zur weiteren Analyse weitergeleitet. Anschließend wird jeder Tweet als positiv, negativ oder neutral klassifiziert.

Das Paket wurde in Python implementiert.

5.2.3 Webanwendung

Der Kern der entwickelten Anwendung ist eine Client-Server-Architektur, in der die Information zwischen Diensten (sogenannten Servern) und Kunden von Diensten (als Clients bezeichnet) verteilt wird. Das Internet wird als Schnittstelle zwischen dem Client und dem Server verwendet. Serverseitig werden die Webseiten nach Anfrage dynamisch generiert und an den Client zurückgeschickt. Clientseitig kommt ein Web-Browser zum Einsatz.

Die Vorteile der Client-Server-Architektur sind folgende:

- Die Webanwendungen können plattformübergreifend eingesetzt werden, da die Ressourcen eines einzelnen Servers von Clients mit unterschiedlichen Betriebssystemen und Hardwareplattformen verwendet werden können.
- Keine clientseitige Softwareinstallation notwendig.
- Die Anwendung wird an einer zentralen Stelle (Server) angepasst und erweitert.

[18] Application Programming Interface; wird über Tweepy Bibliothek (http://www.tweepy.org (09.04.2018)) zugegriffen

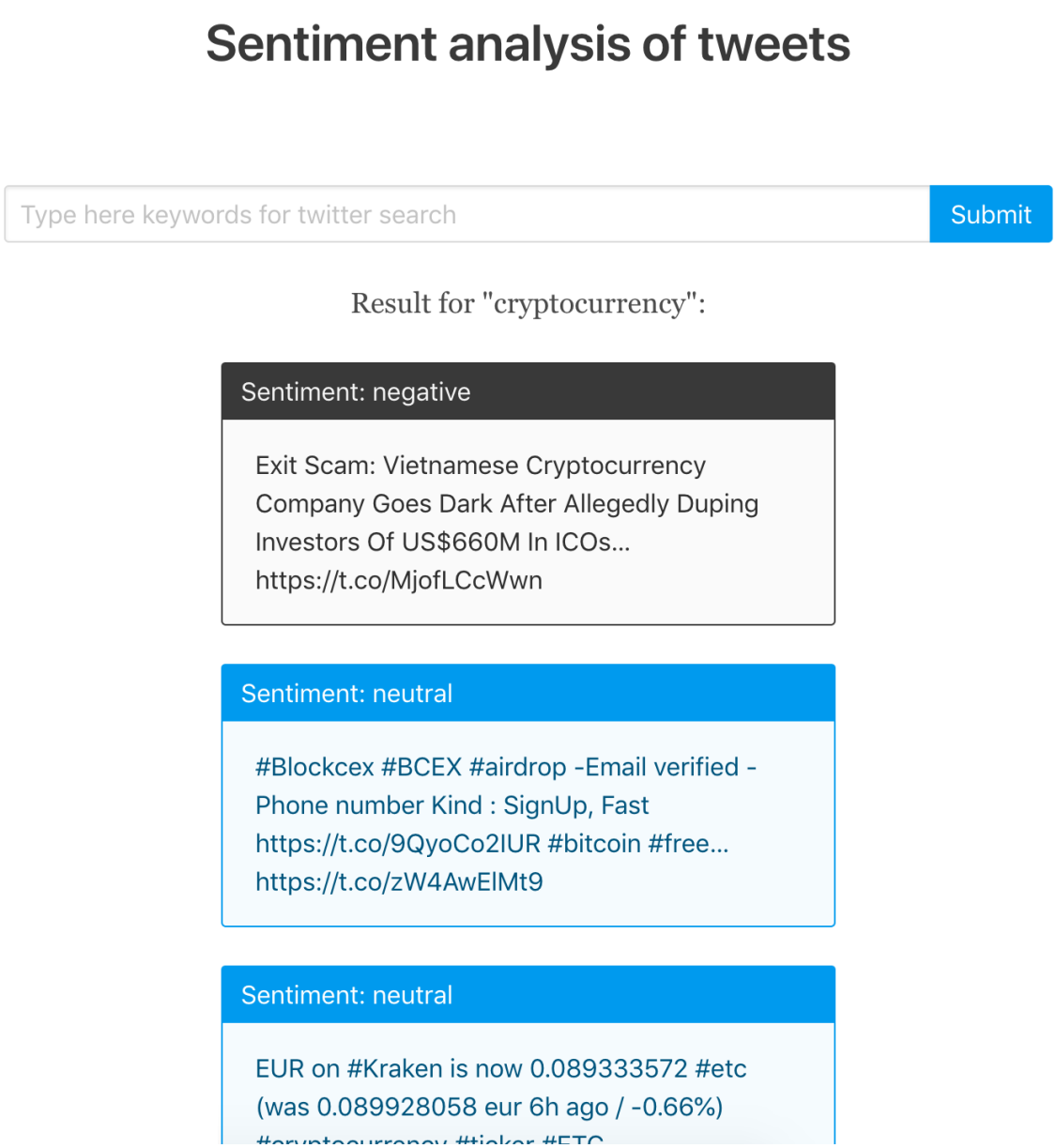

Abbildung 5.1: Webanwendung

Die Abbildung 5.1 zeigt einige typische Ergebnisse beim Ablaufen einer Suchanfrage für das Wort „cryptocurrency“.

Um die Webanwendung übersichtlich zu entwickeln, die auch gut wartbar und erweiterbar ist, wurde das „MVC“ (Engl.: Model, View, Controller) Entwurfsmuster verwendet. „MVC“ bezeichnet die Trennung von Software in die drei Komponenten: Datenmodell, Präsentation und Programmsteuerung. [Gamma u. a. (2001)]

Die Webanwendung nimmt die Suchanfragen von Clients entgegen und leitet sie weiter an Twitter API Wrapper Paket. Anschließend werden die gefundenen und klassifizierten Tweets in eine Tabelle auf der Webseite zusammengebracht.

Diese Komponente wurde in PHP implementiert. Dazu wurden Composer Abhängigkeitsmanager[19], Slim Framework[20] und Twig Template-Engine[21] benutzt.

5.3 Arbeitsablauf

Der gesamte Arbeitsprozess ist im Diagramm 5.2 abgebildet. (Siehe nächste Seite)

Das Workflow enthält die folgenden Schritte:

1. Der Client geht auf die Webseite der Anwendung, gibt ein oder mehrere Stichwörter ein und schickt die Suchanfrage an den Server.

2. Der Server (vor allem der PHP-Prozess) bekommt die Anfrage, prüft, ob die Eingabedaten valide sind, und leitet sie weiter an Python Prozess.

3. Der Python Prozess beziehungsweise die Twitter API Wrapper Komponente lädt das gespeicherte Modell (die gelernte Funktion) und stellt die Verbindung zum Twitter API her.

4. Nach der Authentifizierung werden zehn aktuelle Tweets runtergeladen, die die eingegebenen Stichwörter enthalten. Aus den Tweets werden die Inhalte extrahiert. Die Metadaten wie Zeitstempel oder Identifikationsnummer werden nicht betrach-

[19] https://getcomposer.org (09.04.2018)
[20] https://www.slimframework.com (09.04.2018)
[21] https://twig.symfony.com (09.04.2018)

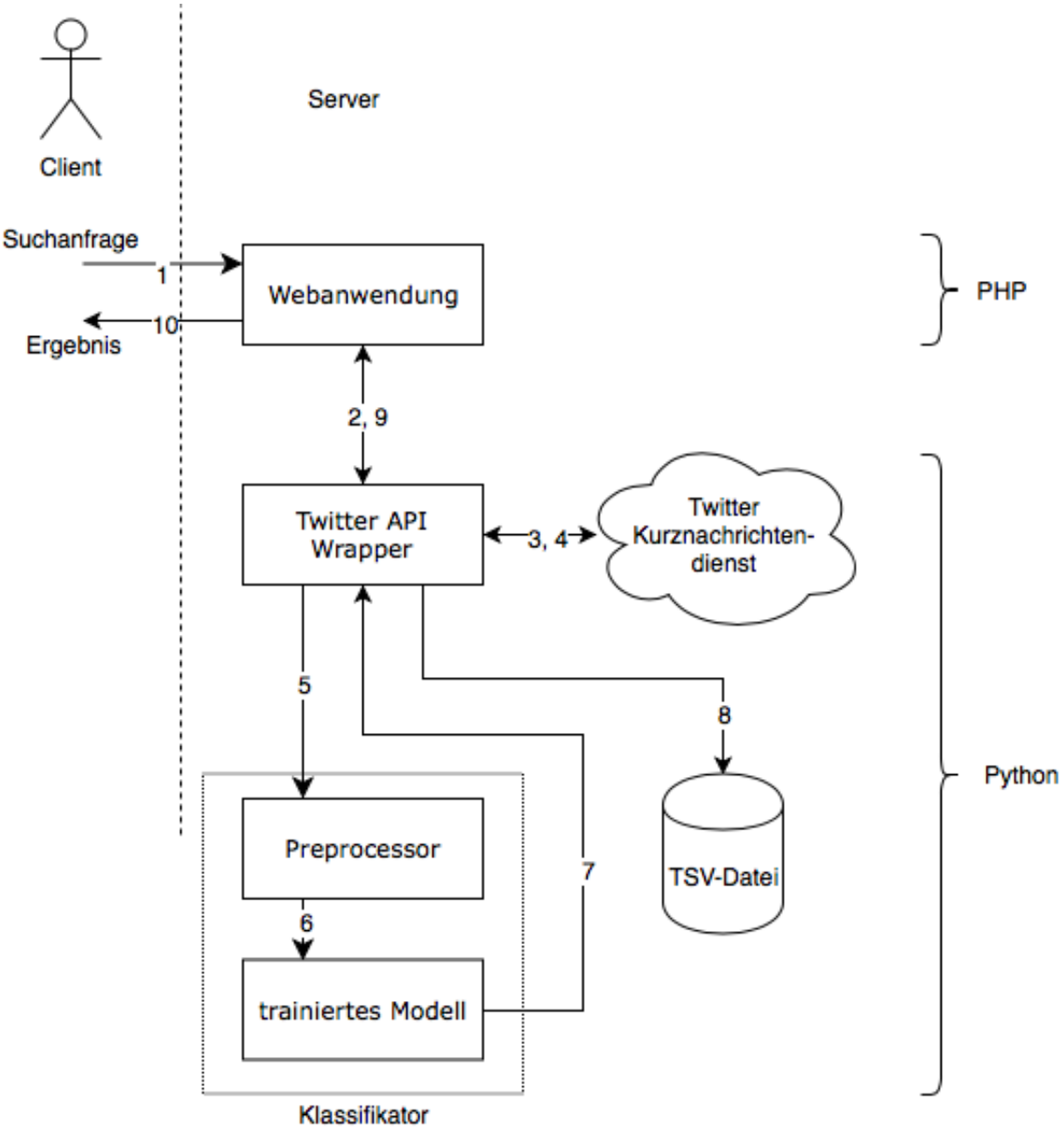

Abbildung 5.2: Arbeitsablauf

tet. Anschließend werden die Tweets zur Stimmungsanalyse an „Klassifikator“ Komponente weitergeleitet.

5. Die Tweets werden bereinigt und verarbeitet laut dem Plan aus dem Abschnitt 4.3.1. Somit wird jedes Tweet in Terme zerlegt. Diese Repräsentation wird weiter an eine Stimmungsanalyse-Pipeline übergeben.

6. Als nächster Schritt werden die Listen von Termen vektorisiert, also durch die Listen von booleschen Werten ersetzt. Diese booleschen Werte entsprechen dem vom Modell erlernten Wortschatz und bedeuten das Vorhandensein jeweiliger Wörter. Anschließend werden die Vektoren ans Modell übergeben.

7. Das Modell bewertet den Vektor (berechnet die Wahrscheinlichkeiten, dass der Vektor zu einer bestimmten Klasse gehört) und schickt das Ergebnis an Twitter API Wrapper Komponente zurück. Anschließend werden die Wahrscheinlichkeitswerte von dieser Komponente beurteilt. Falls nicht eindeutig, wird die Meinung des Tweets als neutral betrachtet.

8. Die Ergebnisse werden als CSV Datei gespeichert. Falls die Datei schon existiert, werden die neuen Daten am Ende hinzugefügt. Sonst wird die Datei automatisch erzeugt. Die Ergebnisse fassen die ursprünglichen Tweet-Inhalte und ihre Tonalität um. Diese Protokollierung ermöglicht die nachträgliche manuelle Kontrolle der Bewertungen. Zum späteren Zeitpunkt könnte die resultierende Datei als eigenständiges oder zusätzliches Korpus für weitere Klassifikationsmodelle verwendet werden.

9. Die Ergebnisse werden über die Standardausgabe in einer JSON-Repräsentation[22] ausgegeben. Diese Repräsentation wird zum Resultat der Client-Suchanfrage im PHP-Prozess.

10. Im letzten Schritt werden die Tweets zum eingegebenen Thema und ihre Tonalität in übersichtlicher Form auf der Webseite dargestellt.

[22] JavaScript Object Notation

5.4 Technische Voraussetzungen

5.4.1 Client-Anforderungen

Da es sich um eine Webanwendung handelt, kann sie auf den meisten Betriebssystemen über einen Webbrowser ausgeführt werden. Im Laufe der Vorbereitung dieser Arbeit wurden Microsoft Windows 10,[23] Apple macOS 10.13,[24] Google Chrome 65[25] und Microsoft Edge 41[26] ohne erkannten Fehler getestet.

5.4.2 Server-Anforderungen

Die folgenden Technologien werden für den Server benötigt:

- Apache HTTP Server[27] (oder analog andere);
- PHP 7-Interpreter mit Composer Abhängigkeitsmanager;
- Python 3.5-Interpreter mit zusätzlichen *NLTK*, *scikit-learn* [die Referenzen auf die jeweiligen Webseiten wurden im Abschnitt 4.3.2 gegeben], *Tweepy* und *unicodecsv*[28] Modulen.

Die Lizenzen der oben genannten Serverdienste und Bibliotheken erlauben ihre freie Verwendung in wissenschaftlichen Projekten.

[23] https://www.microsoft.com/de-de/windows (09.04.2018)
[24] https://www.apple.com/de/macos/high-sierra/ (09.04.2018)
[25] https://www.google.de/chrome/ (09.04.2018)
[26] https://www.microsoft.com/de-de/windows/microsoft-edge (09.04.2018)
[27] https://httpd.apache.org (09.04.2018)
[28] https://github.com/jdunck/python-unicodecsv (09.04.2018)

6 Zusammenfassung

In dem folgenden Abschnitt wird das Fazit der gesamten Arbeit gezogen. Anschließend wird ein Ausblick auf mögliche weiterführende Untersuchungen gegeben, welche sich aus den Resultaten dieser Arbeit ergeben.

6.1 Fazit

Im Rahmen der vorliegenden Arbeit wurden die gestellten Aufgaben gelöst, und es wurden somit folgende Ergebnisse erzielt:

- Der Themenbereich der Stimmungsanalyse wurde untersucht. Ein Überblick über die aktuelle Forschung und die Aufgaben dieses Gebiets wurde gegeben. Die bestehenden Lösungsmethoden dienten anschließend als Ansatzpunkt für eigenes Projekt.

- Es wurden die Besonderheiten der in den Mikroblogs geäußerten Meinungen analysiert. Die erkannten Merkmale haben eine Verbesserung von Klassifikationsmethoden ermöglicht.

- Es wurden die Methoden zur automatischen Analyse von emotional gefärbten Nachrichten im sozialen Netzwerk Twitter entwickelt.

- Eine experimentelle Bewertung der Wirksamkeit vom überwachten Lernen wurde durchgeführt. Zu den Methoden des überwachten Lernens gehören ein naiver Bayes-Klassifikator und die Stützvektormethode. Die beiden Klassifikatoren haben

die genaueren Ergebnisse der Stimmungsanalyse erzielt, als die lexikonbasierte Methode.

- Es wurden die Möglichkeiten zur Verbesserung der Genauigkeit der Tonalitätsanalyse berücksichtigt. Diesbezüglich wurden auch neue Tests mit deren Verwendung durchgeführt. Infolge der Experimente wurde ein optimaler Ansatz für die Stimmungsanalyse gewählt. Der resultierende Analyseablauf basiert auf einem naiven Bayes'schen Klassifikator, Monogrammen und Extraktion der gegenseitigen Information.

- Abschließend wurde eine Web-Anwendung entwickelt, um Meinungen zu einem bestimmten Thema im sozialen Netzwerk Twitter zu suchen und ihre Tonalität zu analysieren. Diese Anwendung kann beispielsweise dazu verwendet werden, soziologische oder Marktforschung durchzuführen.

- Darüber hinaus wurden Erfahrungen mit Programmiersprache Python, Bibliotheken für maschinelles Lernen und Datenmanipulation sowie Twitter API gesammelt.

6.2 Ausblick

Für die weitere Arbeit in Bezug auf automatisierte Stimmungsanalyse der Tweets werden folgende Möglichkeiten identifiziert:

- Es könnten auch weitere Sprachen außer Englisch unterstützt werden. Dazu ist es notwendig, eine markierte Trainingsstichprobe in jeweiliger Sprache vorzubereiten.

- Es kann die Verwendung von Ensembles von Modellen zur Bestimmung des Nachrichtentonalität untersucht werden.

- Es kann eine Untersuchung der Erkennung von Subjektivität in Texten durchgeführt werden. Da nicht alle Nachrichten eine subjektive Bewertung enthalten, führt die binäre Klassifikation aller Nachrichten auf positive und negative zu falschen Ergebnissen.

- Die umgangssprachlichen Wörter und Abkürzungen können mit Hilfe zusätzliches Wortschatzes in normale Wörter transformiert werden. Dies könnte einen zusätzlichen Qualitätsschub liefern.

- Automatische Korrektur von Rechtschreibungsfehlern kann die Ergebnisse der Klassifizierung verbessern und die Merkmalsdimension reduzieren.

- Zusätzlich kann die Funktionalität der entwickelten Anwendung erweitert werden. Eine Möglichkeit, die Ergebnisse der Stimmungsanalyse interaktiv zu beeinflussen, wäre eine der mehreren Varianten für die Weiterentwicklung. Dies würde erlauben, den Algorithmus während des Betriebs weiter zu lehren.

Literaturverzeichnis

[Barbosa und Feng 2010] Barbosa, Luciano ; Feng, Junlan: Robust Sentiment Detection on Twitter from Biased and Noisy Data. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (2010), S. 36–44

[Bell u. a. 2009] Bell, Robert M. ; Bennett, Jim ; Koren, Yehuda ; Volinsky, Chris: The Million Dollar Programming Prize. (2009). – URL https://spectrum.ieee.org/computing/software/the-million-dollar-programming-prize. – Zugriffsdatum: 2018-04-09

[Bengio 2008] Bengio, Yoshua: Neural net language models. In: *Scholarpedia* 3 (2008), S. 3881

[Bermingham und Smeaton 2010] Bermingham, Adam ; Smeaton, Alan F.: Classifying Sentiment in Microblogs: Is Brevity an Advantage? In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10* (2010), S. 1833–1836

[Cambria und Hussain 2012] Cambria, Erik ; Hussain, Amir: *Sentic Computing: Techniques, Tools and Applications.* Springer Netherlands, 2012

[Caren 2012] Caren, Neal: Sentiment dictionary. (2012). – URL http://nealcaren.web.unc.edu/an-introduction-to-text-analysis-with-python-part-3/. – Zugriffsdatum: 2018-04-09

[Cha u. a. 2010] Cha, Meeyoung ; Haddadi, Hamed ; Benevenuto, Fabrício ; Gummadi, Krishna P.: Measuring user influence in twitter: The million follower fallacy. In: *ICWSM* (2010)

[Davidov u. a. 2010a] Davidov, Dmitry ; Tsur, Oren ; Rappoport, Ari: Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. In: *Proceedings of the 23rd International Conference on Computational Linguistics: Posters* (2010), S. 241–249

[Davidov u. a. 2010b] Davidov, Dmitry ; Tsur, Oren ; Rappoport, Ari: Semi-supervised Recognition of Sarcastic Sentences in Twitter and Amazon. In: *Proceedings of the Fourteenth Conference on Computational Natural Language Learning* (2010), S. 107–116

[Denecke 2008] Denecke, Kerstin: Using sentiwordnet for multilingual sentiment analysis. In: *IEEE 24th International Conference on Data Engineering Workshop* (2008), S. 507–512

[DimensionalResearch 2013] DimensionalResearch: Customer service and business results: a survey of customer service from mid-size companies. (2013). – URL https://d16cvnquvjw7pr.cloudfront.net/resources/whitepapers/Zendesk_WP_Customer_Service_and_Business_Results.pdf. – Zugriffsdatum: 2018-04-09

[Dudenredaktion 2017] Dudenredaktion: „Tulpe“ auf Duden online. (2017). – URL https://www.duden.de/node/641764/revisions/1658556/view. – Zugriffsdatum: 2018-04-09

[Gamma u. a. 2001] Gamma, Erich ; Helm, Richard ; Johnson, Ralph: *Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software.* Addison-Wesley, 2001

[Go u. a. 2009] Go, Alec ; Bhayani, Richa ; Huang, Lei: Twitter sentiment classification using distant supervision. In: *Stanford CS224N Project Report* (2009)

[Ikonomakis u. a. 2005] Ikonomakis, M. ; Kotsiantis, S. ; Tampakas, V.: Text Classification Using Machine Learning Techniques. In: *WSEAS Transactions on Computers* 4 (2005), S. 966–974

[Jansen u. a. 2009] Jansen, Bernard J. ; Zhang, Mimi ; Sobel, Kate ; Chowdury, Abdur: Twitter power: Tweets as electronic word of mouth. In: *Journal of the American Society for Information Science and Technology* 60 (2009), Nr. 11, S. 2169–2188

[Jindal und Liu 2006] Jindal, Nitin ; Liu, Bing: Mining comparative sentences and relations. In: *Proceedings of 21st National Conference on Artificial Intelligence, AAAI '06* (2006)

[Jurafsky und Martin 2009] Jurafsky, Daniel ; Martin, James H.: *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., 2009

[Kan 2012] Kan, Dmitry: Rule-based approach to sentiment analysis. In: *Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference Dialog 2012* (2012)

[Kennedy und Inkpen 2006] Kennedy, Alistair ; Inkpen, Diana: Sentiment classification of movie reviews using contextual valence shifters. In: *Computational Intelligence* 22 (2006), Nr. 2, S. 110–125

[Kohavi und Provost 1998] Kohavi, Ron ; Provost, Foster: Glossary of terms. In: *Machine Learning* 30 (1998), Nr. 2-3, S. 271–274

[Koren 2009] Koren, Yehuda: The BellKor Solution to the Netflix Grand Prize. (2009). – URL https://www.asc.ohio-state.edu/statistics/dmsl/GrandPrize2009_BPC_BellKor.pdf. – Zugriffsdatum: 2018-04-09

[Krikorian 2013] Krikorian, Raffi: New Tweets per second record, and how! (2013). – URL https://blog.twitter.com/engineering/en_us/a/2013/new-tweets-per-second-record-and-how.html. – Zugriffsdatum: 2018-04-09

[Lin und He 2009] Lin, Chenghua ; He, Yulan: Joint Sentiment/Topic Model for Sentiment Analysis. In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09* (2009), S. 375–384

[Liu 2010a] Liu, Bing: Sentiment analysis: A multi-faceted problem. In: *IEEE Intelligent Systems* (2010)

[Liu 2010b] Liu, Bing: Sentiment analysis and subjectivity. In: *Handbook of Natural Language Processing, Second Edition* (2010)

[Liu 2012] Liu, Bing: *Sentiment Analysis and Opinion Mining*. Morgan and Claypool Publishers, 2012

[Manning u. a. 2008] Manning, Christopher D. ; Raghavan, Prabhakar ; Schütze, Hinrich: *Introduction to Information Retrieval*. Cambridge University Press, 2008

[Manning und Schütze 1999] Manning, Christopher D. ; Schütze, Hinrich: *Foundations of Statistical Natural Language Processing*. MIT Press, 1999

[Martineau und Finin 2009] Martineau, Justin ; Finin, Tim: Delta TFIDF: An Improved Feature Space for Sentiment Analysis. In: *Proceedings of the 3rd International Conference on Weblogs and Social Media, ICWSM '09* (2009)

[McCallum und Nigam 1998] McCallum, Andrew ; Nigam, Kamal: A Comparison of Event Models for Naive Bayes Text Classification. In: *Learning for Text Categorization: Papers from AAAI Workshop* (1998), S. 41–48

[Nakov u. a. 2016] Nakov, Preslav ; Ritter, Alan ; Rosenthal, Sara ; Stoyanov, Veselin ; Sebastiani, Fabrizio: Sentiment Analysis in Twitter. (2016). – URL http://alt.qcri.org/semeval2016/task4/. – Zugriffsdatum: 2018-04-09

[Pang und Lee 2004] Pang, Bo ; Lee, Lillian: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In: *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics, ACL '04* (2004)

[Pang und Lee 2008] Pang, Bo ; Lee, Lillian: Opinion mining and sentiment analysis. In: *Foundations and Trends in Information Retrieval* 2 (2008), Nr. 1-2, S. 1–135

[Pang u. a. 2002] Pang, Bo ; Lee, Lillian ; Vaithyanathan, Shivakumar: Thumbs up?: sentiment classification using machine learning techniques. In: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing, EMNLP '02* 10 (2002), S. 79–86

[Piotte und Chabbert 2009] Piotte, Martin ; Chabbert, Martin: The Pragmatic Theory Solution to the Netflix Grand Prize. (2009). – URL

https://www.asc.ohio-state.edu/statistics/statgen/joul_aut2009/PragmaticTheory.pdf. – Zugriffsdatum: 2018-04-09

[Porter 2006] Porter, Martin: The Porter Stemming Algorithm. (2006). – URL https://tartarus.org/martin/PorterStemmer. – Zugriffsdatum: 2018-04-09

[Princeton 2010] Princeton: WordNet's morphological processing. (2010). – URL https://wordnet.princeton.edu/documentation/morphy7wn. – Zugriffsdatum: 2018-04-09

[Rosen 2017] Rosen, Aliza: Tweeting Made Easier. (2017). – URL https://blog.twitter.com/official/en_us/topics/product/2017/tweetingmadeeasier.html. – Zugriffsdatum: 2018-04-09

[Saif u. a. 2012] Saif, Hassan ; He, Yulan ; Alani, Harith: Alleviating data sparsity for Twitter sentiment analysis. In: *CEUR Workshop Proceedings (CEUR-WS.org)* (2012), S. 2–9

[Statista 2017] Statista: Anzahl der monatlich aktiven Nutzer von Twitter weltweit vom 1. Quartal 2010 bis zum 4. Quartal 2016 (in Millionen). (2017). – URL https://de.statista.com/statistik/daten/studie/232401/umfrage/monatlich-aktive-nutzer-von-twitter-weltweit-zeitreihe. – Zugriffsdatum: 2018-04-09

[Tan u. a. 2011] Tan, Chenhao ; Lee, Lillian ; Tang, Jie ; Jiang, Long ; Zhou, Ming ; Li, Ping: User-level Sentiment Analysis Incorporating Social Networks. In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2011), S. 1397–1405

[Thelwall u. a. 2011] Thelwall, Mike ; Buckley, Kevan ; Paltoglou, Georgios: Sentiment in Twitter events. In: *Journal of the American Society for Information Science and Technology* 62 (2011), Nr. 2, S. 406–418

[Thelwall u. a. 2010] Thelwall, Mike ; Buckley, Kevan ; Paltoglou, Georgios ; Cai, Di ; Kappas, Arvid: Sentiment strength detection in short informal text. In: *Journal of the American Society for Information Science and Technology* 61 (2010), Nr. 12, S. 2544–2558

[Tong und Koller 2002] Tong, Simon ; Koller, Daphne: Support Vector Machine Active Learning with Applications to Text Classification. In: *The Journal of Machine Learning Research* 2 (2002), S. 45–66

[Töscher u. a. 2009] Töscher, Andreas ; Jahrer, Michael ; Bell, Robert M.: The BigChaos Solution to the Netflix Grand Prize. (2009). – URL https://www.asc.ohio-state.edu/statistics/dmsl/GrandPrize2009_BPC_BigChaos.pdf. – Zugriffsdatum: 2018-04-09

[Wang und Manning 2012] Wang, Sida ; Manning, Christopher D.: Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2, ACL '12* (2012), S. 90–94

[Whitelaw u. a. 2005] Whitelaw, Casey ; Garg, Navendu ; Argamon, Shlomo: Using Appraisal Groups for Sentiment Analysis. In: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management, CIKM '05* (2005), S. 625–631

[Wiegand u. a. 2010] Wiegand, Michael ; Balahur, Alexandra ; Roth, Benjamin ; Klakow, Dietrich ; Montoyo, Andrés: A Survey on the Role of Negation in Sentiment Analysis. In: *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing* (2010), S. 60–68

[Wilson u. a. 2013] Wilson, Theresa ; Kozareva, Zornitsa ; Nakov, Preslav ; Ritter, Alan ; Rosenthal, Sara ; Stoyanov, Veselin: Sentiment Analysis in Twitter. (2013). – URL https://www.cs.york.ac.uk/semeval-2013/task2. – Zugriffsdatum: 2018-04-09

Tabellenverzeichnis

Abbildungsverzeichnis

Appendix A: Die „Preprocessor“ Klasse

```
#!/usr/bin/env python3
# -*- encoding: utf-8 -*-

from itertools import groupby
from nltk.tokenize import TweetTokenizer
from nltk.corpus import wordnet
import re
import string
from sklearn.base import BaseEstimator, TransformerMixin
import os
import pickle


class Preprocessor(BaseEstimator, TransformerMixin):
    """
    Processes and prepares text data
    for machine learning algorithm
    """

    def __init__(self):
        """
        Instantiate and load dependent models
        """
        wd = os.path.dirname(__file__)
        signs_path = os.path.join(wd, 'pkl', 'emotion_signs.pkl')
```

```
26          emotion_signs = pickle.load(open(signs_path, 'rb'))
27          stop_path = os.path.join(wd, 'pkl', 'stop_words.pkl')
28          stoplist = pickle.load(open(stop_path, 'rb'))
29          self.emotion_signs = emotion_signs
30          self.tokenizer = TweetTokenizer(preserve_case=False,
31                                          strip_handles=True)
32          self.dup_chars_re = re.compile(r'(\w*)(\S)\2(\w*)')
33          self.s_ending_re = re.compile(r"\'s$")
34          try:
35              # UCS-4
36              self.unneeded_symbols = re.compile(
37                  u'([^\U00000000-\U0000d7ff\U0000e000-\U0000ffff])')
38          except re.error:
39              # UCS-2
40              self.unneeded_symbols = re.compile(
41                  u'([^\u0000-\ud7ff\ue000-\uffff])')
42          self.url_and_email_re = re.compile(r'^\S+\.\S+$')
43          punctuation = string.punctuation.replace('!', '') \
44                                          .replace('?', '')
45          self.punctuation_re = re.compile(
46              r'^[%s]' % re.escape(punctuation))
47          self.numbers_re = re.compile(
48              r'^[-]?\d[\d,]*[\.]?[\d{2}]*|^(?:n|nr|no|x)\d+')
49          self.stop_words = stoplist
50
51      def fit(self, X, y=None):
52          """
53          No action is needed
54          """
55          return self
56
57      def transform(self, X):
58          """
59          Preprocess input data
60          """
61          return [
```

```
62              list(self.pipe(string,
63                             self.remove_emotion_signs,
64                             self.tokenizer.tokenize,
65                             self.reduce_dup_tokens,
66                             self.reduce_dup_chars_and_endings,
67                             self.clean)) for string in X
68          ]
69
70      def inverse_transform(self, X):
71          """
72          No action available
73          """
74          return X
75
76      def reduce_dup_symbols(self, string):
77          """
78          removes extra characters, when
79          there are 3+ equal characters in a row.
80          only 2 equal characters will be left
81          """
82          return ''.join(''.join(s)[:2] for _, s in groupby(string))
83
84      def remove_emotion_signs(self, string):
85          """
86          replaces predefined emotion signs
87          with their meaning (positive or negative)
88          """
89          result = self.reduce_dup_symbols(string) \
90                       .replace('://', '.')
91          for sign in self.emotion_signs:
92              result = result.replace(sign[0], '_' + sign[1] + '_')
93          return result
94
95      def reduce_dup_tokens(self, list_of_tokens):
96          """
97          removes extra duplicates, when
```

```
98          there are 2+ equal tokens in a row.
99          only one such token will be left
100         """
101         return [k for k, _ in groupby(list_of_tokens)]
102
103     def reduce_dup_chars_and_endings(self, list_of_tokens):
104         """
105         reduces 2 equal chars and symbols in a row into one,
106         if there is no such word in NLTK Wordnet corpus.
107         it also deletes 's endings in the same cycle
108         """
109         def replace(before):
110             if wordnet.synsets(before):
111                 return before
112             after = self.dup_chars_re.sub(r'\1\2\3', before)
113             return replace(after) if after != before else after
114         return [re.sub(self.s_ending_re, '', replace(t))
115                 for t in list_of_tokens]
116
117     def clean(self, list_of_tokens):
118         """
119         removes irrelevant emoji,
120         irrelevant non-latin characters,
121         tokens holding a domain name (applies to urls, e-mails),
122         hashtags and other punctuation symbols,
123         number tokens,
124         single character tokens,
125         stopwords
126         """
127         return [t for t in list_of_tokens if
128                 not self.unneeded_symbols.match(t) and
129                 not re.sub(
130                   r'[^\x00-\x7f\x80-\xff]', '', t) == '' and
131                 not self.url_and_email_re.match(t) and
132                 not self.punctuation_re.match(t) and
133                 not self.numbers_re.match(t) and
```

```
134                 not re.fullmatch('[a-zA-Z]', t) and
135                 t not in self.stop_words]
136
137     def pipe(self, arg, *funcs):
138         """
139         enables pipeline for multiple actions to be applied
140         one after another
141         """
142         for f in funcs:
143             arg = f(arg)
144         return arg
145
146
147 def analyzer(list_of_tokens):
148     """
149     Forwards tokens without extra steps
150     """
151     return list_of_tokens
```

Listing 1: „Preprocessor“ Klasse

Appendix B: Modell Training

```
#!/usr/bin/env python3
# -*- encoding: utf-8 -*-

import pandas as pd
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import mutual_info_classif
from sklearn.naive_bayes import BernoulliNB
# only binary vectors supported
from sklearn.externals import joblib

# training sets are loaded into 'pandas' dataframes

trainset_2013 = pd.read_csv('semeval2013train.tsv', header=None,
                            delimiter="\t", quoting=3)
trainset_2016 = pd.read_csv('semeval2016train.tsv', header=None,
                            delimiter="\t", quoting=3)
trainset = pd.concat([trainset_2013, trainset_2016],
                     ignore_index=True)
trainset.columns = ["Id", "Sentiment", "Text"]

# There are so many rows and columns:
trainset.shape
#    (31474, 3)

# and so many tweets:
```

```
trainset.Sentiment.value_counts()

#    neutral     14848
#    positive    12008
#    negative     4618
#    Name: Sentiment, dtype: int64

# we delete the neutral ones.
trainset.drop(trainset[trainset.Sentiment == 'neutral'] /
              .index, inplace=True)

trainset.Sentiment.value_counts()

#    positive    12008
#    negative     4618
#    Name: Sentiment, dtype: int64

# here comes Preprocessor class
# ...

preprocessor = Preprocessor()
trainset['Unigrams'] = preprocessor.transform(trainset.Text)

# now we can prepare the whole pipeline
# 1. boolean weights + BernoulliNB;
# 2. mutual information with 5000 features;


def analyzer(list_of_tokens):
    """
    Forwards tokens without extra steps
    """
    return list_of_tokens

model = Pipeline([
            ('preprocessor', Preprocessor()),
```

```
64              ('vectorizer', CountVectorizer(analyzer=analyzer,
65                                             lowercase=False,
66                                             binary=True)),
67              ('mi_solver', SelectKBest(mutual_info_classif,
68                                        k=5000)),
69              ('classifier', BernoulliNB(alpha=0.1))
70          ])
71
72  model.fit(trainset.Text, trainset.Sentiment)
73
74  # now we can serialize and export our model
75  joblib.dump(model, open(os.path.join('pkl', 'model.jl'),
76                          'wb'))
```

Listing 2: Modell Training

Appendix C: Die Twitter API Wrapper Komponente

```
#!/usr/bin/env python3
# -*- encoding: utf-8 -*-

import sys
import json
import tweepy
from tweepy import OAuthHandler
from preprocess import Preprocessor
from preprocess import analyzer
import os
from sklearn.externals import joblib
import unicodecsv as csv


class TwitterSentimentClient():
    """
    Twitter Class for sentiment analysis
    """
    def __init__(self):
        """
        Class constructor or initialization method
        """
        # Twitter authentication data
        user_key = "****"
        user_secret = "****"
```

```
26          acc_token = "***"
27          acc_secret = "***"
28
29          # ML model
30          self.wd = os.path.dirname(__file__)
31          self.model = joblib.load(open(os.path.join(self.wd,
32                                                     'pkl',
33                                                     'model.jl'),
34                                        'rb'))
35
36          # attempt authentication
37          try:
38              # create OAuthHandler object
39              self.auth = OAuthHandler(user_key, user_secret)
40              # set access token and secret
41              self.auth.set_access_token(acc_token, acc_secret)
42              # create tweepy API object to fetch tweets
43              self.api = tweepy.API(self.auth)
44          except:
45              print(json.dumps({'error': 'Authentication_Failed'}))
46              sys.exit(1)
47
48      def sentiment(self, string):
49          """
50          detect text polarity.
51          if polarity detection confidence (probability) is >=0.9
52          classify as subjective, otherwise, classify as neutral
53          """
54          pos_prob = self.model.predict_proba([string])[:, 1]
55          if pos_prob >= 0.9:
56              sentiment = 'positive'
57          elif pos_prob < 0.1:
58              sentiment = 'negative'
59          else:
60              sentiment = 'neutral'
61          return sentiment
```

```
62
63      def fetch_tweets(self, query, count=10):
64          """
65          Fetch and parse tweets
66          """
67          # tweet list for output
68          tweets = []
69
70          try:
71              # fetch tweets from twitter api
72              fetched = self.api.search(q=query + '␣-filter:rt',
73                                        lang='en',
74                                        count=count,
75                                        result_type='recent')
76
77              # parse tweets and detect sentiment
78              for tweet in fetched:
79                  parsed = {}
80                  parsed['sentiment'] = self.sentiment(tweet.text)
81                  parsed['text'] = tweet.text
82                  tweets.append(parsed)
83
84              return tweets
85
86          except tweepy.TweepError as e:
87              print(json.dumps({'error': str(e)}))
88              sys.exit(1)
89
90
91  def main():
92      # create working instance
93      api = TwitterSentimentClient()
94      # search for tweets by user input
95      tweets = api.fetch_tweets(query=sys.argv[1], count=10)
96      # save results for future analysis
97      columns = tweets[0].keys()
```

```
98      out_path = os.path.join(api.wd, 'output', 'res.tsv')
99      with open(out_path, 'ab') as csv_output:
100         dict_writer = csv.DictWriter(csv_output,
101                                      columns,
102                                      delimiter='\t')
103         dict_writer.writerows(tweets)
104     # return found tweets with their sentiment
105     print(json.dumps(tweets))
106
107 if __name__ == "__main__":
108     # call main function
109     main()
```

Listing 3: Twitter API Wrapper Klasse

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 13. April 2018 Igor Arkhipov