Universität Zürich[UZH]

Bachelorarbeit
Frühlingssemester 2016

Sentimentanalyse für Twitter

Verfasser: Lukas Fischer
Matrikel-Nr: 12-730-289

Referent: Martin Volk
Betreuer: Manfred Klenner
Institut für Computerlinguistik

Abgabe: Mai 2016

Inhaltsverzeichnis

1 Einleitung

In den letzten Jahren wurden Microblogging-Plattformen wie Twitter immer populärer. Dies hat zur Folge, dass viele Firmen daran interessiert sind, auf diesen Plattformen gepostete Informationen zu nutzen, um herauszufinden, wie ihre Produkte bei den Twitter-Usern ankommen [Kouloumpis et al., 2011]. Sentimentanalyse Tools für Twitter sind darum äusserst gefragt, und werden bereits vertrieben (z.B. tweetfeel[1]).

Die Sentimentanalyse, oft auch Opinion Mining genannt, befasst sich mit der Extraktion von Meinungen und Einstellungen zu bestimmten Themen aus Texten. Sentiment ist zwar ein vielschichtiger Begriff aus der Linguistik, wird aber oft einfach als *positive oder negative Emotion* definiert [Pang and Lee, 2002]. Während die Einteilung in zwei Klassen praktisch und einfach ist, gibt es natürlich unzählige Abstufungen wie *ganz OK, fürchterlich, ziemlich gut*, etc. Polaritätslexika werden diesem Umstand bereits gerecht, da sie für positive Einträge Werte über Null verwenden und für negative Werte unter Null. Je grösser der Wert, desto stärker ist die jeweilige Emotion. Für Maschinelles Lernen ist es aber einfacher sich auf zwei Klassen zu beschränken, was je nach Ziel des Systems auch schon genügt. Sentimentanalyse für Twitter erlaubt es auch, die Meinung der User in Echtzeit zu analysieren, wie [Illecker, 2015] es in seinem Projekt tut.

Während Sentimentanalyse für Reviews oder Zeitungsartikel bereits ausgiebig erforscht wurde, bietet das Microblogging-Format neue Herausforderungen. Einerseits fallen Tweets durch informelle Sprache und Rechtschreibfehler auf. Ausserdem sind Tweets auf 140 Zeichen beschränkt, ein durchschnittlicher Tweet ist gemäss [Go et al., 2009] 14 Wörter oder 78 Zeichen lang. Diese Eigenschaften verkomplizieren Sentimentanalyse, da Wörter trotz Fehlern (und auch Interjektionen und Abkürzungen wie *yolo, lol* oder *omfg*) erkannt werden müssen, damit aus 14 Wörter überhaupt noch ein Sentiment abgeleitet werden kann. Der Vorteil an Twitter ist aber, dass Daten in grosser Anzahl zur Verfügung stehen und öffentlich sind.

In dieser Bachelorarbeit trainiere ich selbst einen Classifier, der Sentimentanalyse für Twitter in englischer Sprache betreibt. Dabei lege ich besonderen Wert darauf, Features zu verwenden, die spezifische Merkmale von Twitter, wie Emoticons und Hashtags, abbilden. Um das informelle Vokabular abzubilden, kommen Lexika zum Einsatz, die aus Social Media Posts kompiliert wurden. Ich verwende zwei verschiedene Korpora mit jeweils leicht unterschiedlichem Featureset. Ausserdem vegleiche ich die Performanz bei zwei Klassen (*positiv/negativ*) und drei Klassen (*positiv/negativ/neutral*) sowie bei verschieden grossen Trainingssets.

[1] `http://www.tweetfeel.com/`

2 Literaturdiskussion

Im Bereich Sentimentanalyse für Twitter wurde bereits reichlich Forschung betrieben. Im folgenden Abschnitt werde ich einige Ansätze vorstellen.

Da in Tweets grosses Rauschen vorhanden ist, werden sie in der Regel vorverarbeitet, was das Problem von *Sparse Data* minimiert. [Castellucci et al., 2013] und [Go et al., 2009] ersetzen in ihren Experimenten beispielsweise Usernamen, Hashtags und URL's durch je einen Platzhalter, entfernen wiederholte gleiche Zeichen (z.B. *noooooo!!!!!!* wird zu *no!!*) und setzen gänzlich grossgeschriebene Wörter durch ihre kleingeschriebe Version.

Häufig wird dann ein supervisiertes Lernverfahren angewendet (siehe u.a. [Mohammad et al., 2013], [Illecker, 2015], [Spencer and Uchyigit, 2012]), bei dem unterschiedliche Classifier angewendet werden können. [Nakov et al., 2013] stellten fest, dass Naive Bayes und Support Vector Machines (SMV) zu den beliebtesten Classifiern zählen. Naive Bayes schätzt aus der *a priori* Wahrscheinlichkeit eines Features die Wahrscheinlichkeit der Klasse und nimmt dabei die Unabhängigkeit zwischen Features an. SVM vergleichen Featurevektoren miteinander, wobei jedes Feature auf den Vektoren abgebildet ist. Der Classifier separiert dann (wenn möglich) die Vektoren räumlich in ihre Klassen.

Die Features werden aus den Tweets generiert. Beispiele für Features sind Wort- und Zeichen-N-Gramme, Satzzeichen, Länge des Tweets und Polaritätswerte, die aus Sentimentlexika gelesen werden [Mohammad et al., 2013]. Das semi-supervisierte System UNITOR von [Castellucci et al., 2013] kommt aber auch ohne Sentiment-Lexikon aus: Zusätzlich zu annotierten Trainingsdaten werden auch semantische Ähnlichkeiten aus einem nicht annotierten Korpus extrahiert. Der Vorteil ist hier einerseits, dass die Arbeit wegfällt, ein semantisches Lexikon zu erstellen und andererseits, dass aus dem unannotierten Korpus für Menschen nicht greifbare Strukturen ermittelt werden können. Allerdings schneiden solche Verfahren tendenziell schlechter ab. UNITOR erreichte beim SemEval 2013 Workshop einen F-Score von 58.27%, während das supervisierte Verfahren von [Mohammad et al., 2013], welches Polaritätslexika verwendete, einen Score von 69.02% erzielte.

Ein weiterer Ansatz ist die Verwendung Neuronaler Netze, die für supervisierte oder unsupervisierte Lernverfahren eingesetzt werden können und vom biologischen Aufbau des menschlichen Gehirns inspiriert sind. [dos Santos and Gatti, 2014] trainierten mit dieser Methode ein System, welches einen F-Score von 86.4% erreichte (Trainingskorpus: 80'000 Tweets).

Es existieren noch viele weitere innovative Möglichkeiten, Sentimentanalyse für Twitter zu betreiben. Da sich supervisierte Ansätze mit Naive Bayes (und Polaritätslexikon) bewährt haben, werde ich in meiner Bachelorarbeit diese Methode verwenden.

2.1 Korpora und Lexika

Supervisiertes Maschinelles Lernen setzt grosse Mengen an annotierten Daten voraus. Da bereits einige Forschungsarbeiten zum Thema Sentimentanalyse für Twitter gemacht wurden, zum Beispiel [Illecker, 2015], [Spencer and Uchyigit, 2012], [Kouloumpis et al., 2011], [Go et al., 2009] und [Nakov et al., 2013], sind mehrere annotierte Tweet-Korpora verfügbar. Ich habe mir die Korpora von SemEval13 und Sentiment140 ausgesucht, die unterschiedlich zusammengestellt und annotiert wurden, und werde sie in diesem Experiment miteinander vergleichen.
Eine weitere wichtige Komponente für Sentimentanalyse sind auch die sogenannten Emotions-Lexika, in denen für jedes Wort die Polarität in Form einer Zahl angegeben ist. Die Polaritätswerte einzelner Wörter (je grösser der Wert, desto stärker die Polarität), können wertvolle Informationen über die Polarität des gesamten Tweets geben. Ich bediene mich in dieser Arbeit zweier Lexika, die ich in diesem Kapitel ebenfalls kurz vorstelle.

2.1.1 SemEval 2013

SemEval ist ein Workshop, bei dem computerbasierte Semantische Evaluationssysteme miteinander verglichen werden. Die Workshops werden etwa jährlich (mit Unterbrüchen) gehalten [2]. Im Jahr 2013 bestand eine Teilaufgabe darin, die Polarität von Tweets zu bestimmen [Nakov et al., 2013]. Die Organisatoren stellten den Teilnehmern ein annotiertes Korpus zum Trainieren und Testen ihrer Classifier zur Verfügung, welches auch nach dem Projekt noch verfügbar ist.

[Nakov et al., 2013] downloadeten die Tweets über die Twitter API und gruppierten sie nach beliebten Themen, die von [Ritter et al., 2012] folgendermassen definiert werden: *"those named entities that are frequently mentioned in association with a specific date"*. Annotiert wurden die Tweets über die Crowd-Sourcing Plattform Amazon Mechanical Turk (AMT). Über diese Plattform können Annotationsprojekte an eine Crowd verteilt werden, die die Arbeit in viel kürzerer Zeit und auch äusserst preisgünstig erledigen [Fort et al., 2011]. Die Turker, wie die AMT-Worker genannt werden, annotierten die gesamte Polarität der Tweets, identifizierten aber auch Teilkonstrukte. Pro Annotation wurden sie mit drei Cent (US$) Lohn vergütet. Um die Qualität der Annotationen zu gewährleisten, musste jeder Turker in einer Testphase mit mindestens 95% korrekten Klassifikationen bestehen, bevor er das eigentliche Testset annotieren konnte. Ausserdem wurde jeder einzelne Tweet von 5 Turkern annotiert und nur bei Übereinstimmung ins Korpus übernommen.

Die annotierten Trainingsdaten (insgesamt 9451 Tweets) zum SemEval13 Twitter Task 2 stehen zum Download zur Verfügung[3]. Allerdings verbietet Twitter die Weitergabe von gros-

[2] `https://en.wikipedia.org/wiki/SemEval`
[3] `http://alt.qcri.org/semeval2015/task10/index.php?id=data-and-tools`

sen Mengen an Tweets [4], weshalb ich die Tweets selbst herunterladen musste. Nachfolgend ein kleiner Auszug aus der SemEval13 Korpus-Datei, wie sie zur Weitergabe verfügbar ist:

```
264183816548130816 15140428 13 13 positive
263405084770172928 591166521 3 4 negative
262163168678248449 35266263 1 1 neutral
```

In der ersten Spalte befindet sich der Identifier für den annotierten Tweet, mit dem der Tweet über ein Skript heruntergeladen werden kann. In der letzen Spalte befindet sich die Annotation. Die übrigen Metadaten sind hier nicht relevant. Glücklicherweise steht auf der SemEval-Homepage ein Skript zur Verfügung, mit dem die Tweets automatisch heruntergeladen werden können. Allerdings erlaubt Twitter dem Skript nur eine begrenzte Anzahl an Zugriffen pro Stunde, weshalb der Download des gesamten Korpus einen halben Tag dauerte.

Nach dem Download sichtete ich das Korpus und musste feststellen, dass auf viele Tweets nicht mehr zugegriffen werden konnte. Insgesamt gingen auf diese Weise 2083 Tweets verloren. Tabelle 1 gibt einen Überblick über die Grösse des heruntergeladenen SemEval-Korpus:

Klasse	**annotierte Tweets**	**verfügbare Tweets**
Positiv	3640	2841
Negativ	1458	1068
Neutral	4586	3692
Total	9684	7601

Tabelle 1: Anzahl verfügbare Tweets im SemEval13 Korpus nach Klasse.

Da ich in meinem anderen Korpus (beschrieben im nächsten Abschnitt) jeweils gleich grosse Klassen habe, verwende ich in diesem Experiment nur jeweils 1068 Tweets pro Klasse, damit die beiden Experimente vergleichbar sind. Die zu verwendenden Tweets aus den Klassen positiv und neutral habe ich mittels eines Python-Skripts zufällig ausgewählt und dann die Klassen wieder in einem einzelnen File durchmischt.

2.1.2 Sentiment140

Sentiment140 ist eine Twitter API entwickelt von [Go et al., 2009]. Sie ist über ein Web-Interface[5] verfügbar und erlaubt Sentimentanalyse für Tweets mit bestimmten Schlagworten in Echtzeit (allerdings nur in Spanisch und Englisch). Ich habe beispielsweise für das Suchwort *hawaii* den Output in Abbildung 1 erhalten. Zusätzlich zu dieser Grafik werden alle annotierten Tweets ebenfalls angezeigt, zusammen mit der Angabe, wann sie veröffentlicht wurden. Die

[4] `https://dev.twitter.com/overview/terms/policy`
[5] `http://www.sentiment140.com/`

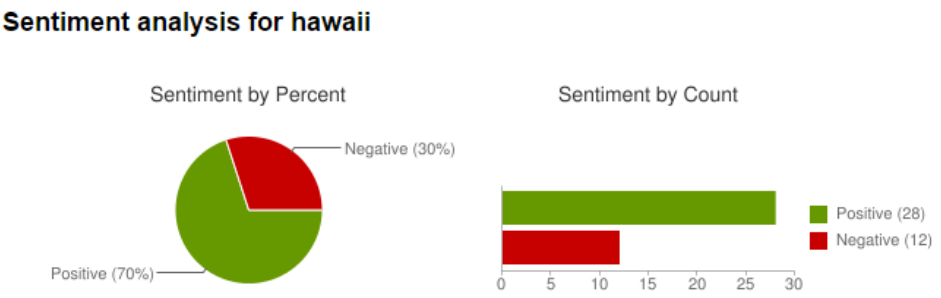

Abbildung 1: Output des Sentiment140-Tools am 21.04.2016 um 16:05 Uhr.

40 Tweets, die meine Abfrage gebracht hat, wurden alle in den 20 vorhergehenden Minuten veröffentlicht.

Die Klassifikation der Tweets liegt einem Korpus zugrunde, das [Go et al., 2009] selbst kompiliert haben. Über zwei Monate sammelten sie Tweets, indem sie Queries mit Emoticons aus Tabelle 2 an die Twitter API schickten. Die Tweets wurden aufgrund der Emoticons als positiv oder negativ annotiert, die Emoticons selbst aber wurden dann entfernt. [Go et al., 2009] begründen ihr Vorgehen damit, dass es oft positive Tweets mit negativen Emoticons gebe, und umgekehrt. Solche Tweets würden dem Classifier nur schaden. Aus demselben Grund wurden alle Tweets wieder gelöscht, die sowohl positive als auch negative Emoticons beinhalteten. Auch sogenannte Retweets, die nur eine Kopie eines Tweets in einem anderen Account sind, wurden nicht verwendet.

Positive Emoticons	`:)` `:-)` `: )` `:D` `=)`
Negative Emoticons	`:(` `:-(` `: (`

Tabelle 2: In den Queries verwendete Emoticons. [Go et al., 2009]

Das daraus resultierende Sentiment140 Korpus umfasst 8 Millionen positive und 8 Millionen negative Tweets. Um einen Vergleich mit dem SemEval13 Korpus zu erlauben, habe ich je 1'000 Tweets von Sentiment140 ausgewählt (Subset 1), allerdings auch noch ein grösseres Set (2) mit jeweils 10'000 Tweets zusammengestellt, um auch Auswirkungen der Performanz in Abhängigkeit der Grösse des Korpus zu untersuchen (siehe Tabelle 3).

Das Korpus ist auf der Homepage von Sentiment140 frei verfügbar[6]. Die Tweets stehen wie folgt in der Datei (Zeilenumbrüche erfolgen eigentlich nur nach Ende eines Tweets):

```
0 1468055791 Mon Apr 06 23:28:49 PDT 2009 NO_QUERY ecjc
@kisluvkis oh that is very sad, poor boy.
```

[6] http://help.sentiment140.com/for-students

```
 4 1467822272 Mon Apr 06 22:22:45 PDT 2009 NO_QUERY ersle
I LOVE @Health4UandPets u guys r the best!!
```

In der ersten Spalte steht jeweils die Polarität der Tweets (0 entspricht negativ, 4 positiv). Darauf folgen Tweet-ID, Datum des Tweets, die Query, mit der der Tweet gesucht wurde, der Username und schliesslich der Tweet selbst. Für mich relevant waren hier also nur die erste Spalte mit der Annotation und die letze mit dem Tweet. Die Metadaten wären nützlich, um z.B. eine Diskursanalyse zu machen. Die Informationen zu User und Zeit wären zusammen mit den @-Anschriften und Retweet-Kennzeichnungen hilfreich, um Konversationen zu rekonstruieren.

Klasse	**Gesamtes Korpus**	**Subset 1**	**Subset 2**
Positiv	8'000'000	1'000	10'000
Negativ	8'000'000	1'000	10'000
Total	16'000'000	2'000	20'000

Tabelle 3: Anzahl Tweets im Sentiment140 Korpus und die im Experiment verwendeten Sets nach Klasse.

2.1.3 VADER-Lexikon und Wörterlisten

VADER-Lexikon: [Hutto and Gilbert, 2014] kompilierten ein Polaritätslexikon für ihr regelbasiertes Sentimentanalyse Tool für Social Media. Das Lexikon umfasst 7517 Einträge, sowohl Wörter und Interjektionen, als auch Emoticons, die von Social Media Plattformen kompiliert wurden. Alle Einträge wurden mit einem von ihnen berechneten Polaritätswert versehen: Positive Werte erhielten eine Zahl über Null, negative eine Zahl kleiner als Null. Je grösser der Betrag einer Zahl, desto stärker die Polarität. Aufgrund der Einträge ist das Lexikon bestens geeignet um Tweets zu analysieren. Einige Beispiele:

`):` -2.1 | `(-:` 1.6 | `lol` 2.9 | `confusing` -0.9 | `freeway` 0.2 | `haha` 2.0 | `hurt` -2.4

Wörterlisten: [Liu et al., 2005] haben für ihre Arbeit zum Thema Meinungsforschung im Internet Wörterlisten von positiven und negativen Wörtern erstellt, die sie angetroffen haben. Insgesamt umfasst die Liste positiver Wörter 2006 Einträge (*a+, excite, ultra-crisp,* etc.), die der negativen 4783 (*abominable, sloooow, yawn,* etc.). Auffällig ist, dass einige Ausdrücke sehr informell und auch falsch geschrieben sind, was für Sentimentanalyse auf Twitter ideal ist.

3 NLP-Pipeline

In diesem Kapitel erläutere ich den Aufbau meines Experiments. Abbildung 2 vermittelt eine Übersicht der einzelnen Schritte, die in den folgenden Abschnitten beleuchtet werden. Die annotierten Korpora werden erst in eine standardisierte Form gebracht, tokenisiert und getaggt. Aus den getaggten Tweets werden dann die fürs Maschinelle Lernen benötigten Features generiert. Diese Liste an Features wird in zwei Teile gespalten, das Trainingsset, welches 90% der Tweets umfasst, und das kleinere Testset (10%). Mit den Features des Trainingssets wird dann der Classifier trainiert. Zum Schluss wird der Classifier auf das Testset angewendet und die Qualität der Klassifikation kann evaluiert werden.

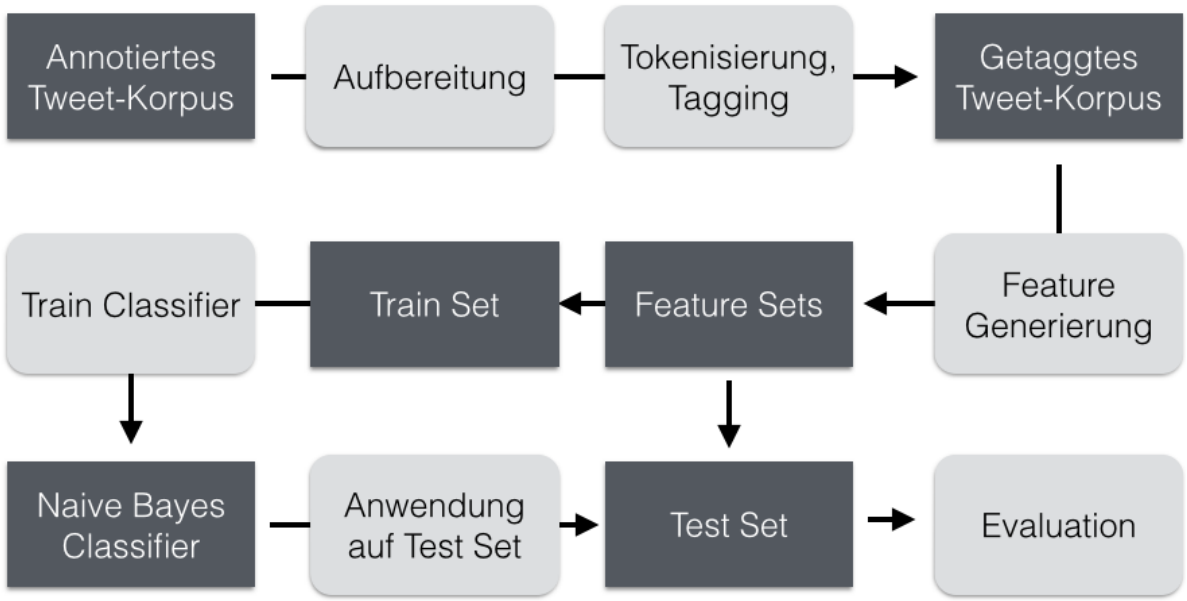

Abbildung 2: Architektur der NLP-Pipeline.

3.1 Aufbereitung der Daten

In einem ersten Schritt habe ich die Files der beiden Korpora eingesehen und die Tweets nach Kategorien sortiert. Ausserdem habe ich alle nicht relevanten Spalten, wie z.B. Informationen zum Zeitpunkt der Veröffentlichung des Tweets, gelöscht. In dieser standardisierten Version des Korpus sind nur noch zwei Spalten, nämlich der Tweet in seiner Rohtextform und die Annotation vorhanden, sodass die weiteren Verarbeitungsschritte für beide Korpora identisch sind.

3.1.1 ARK Tagger

Zur Tokenisierung und zum Taggen der Tweets habe ich den ARK Tagger von [Gimpel et al., 2011] verwendet, die ein spezielles Tagset für das Medium Twitter definiert haben (siehe Tabelle 4). Neben gebräuchlichen Tags wie *Nomen, Präposition* und *Pronomen* haben sie auch twitterspezifische Tags definiert, nämlich *Hashtag, At-Anschrift, URL/Email-Adresse* und *Emoticon*. Der Tagger benutzt einen Conditional Random Fields Classifier. Ihr Featureset beinhaltet unter anderem Reguläre Ausdrücke, zum Erkennen von *Hashtags, URL's* und *At*, traditionelle Tag-Lexika und phonetische Normalisierung nach [Philips, 1990], welche zum Beispiel die verschiedenen angetroffenen Schreibweisen von *thanks* (`thangs thanks thanksss thanx thinks thnx`) alle zu `0NKS` mappen.

Tag	Beschreibung	Tag	Beschreibung
N	Nomen	P	Präposition
O	Pronomen	&	Konjunktion
^	Eigennamen	#	Hashtag
V	Verb	@	At-Anschrift
A	Adjektv	U	URL oder Email-Adresse
R	Adverb	E	Emoticon
!	Interjektion	$	Zahlen
D	Determiner	,	Punktuation

Tabelle 4: Auszug aus dem Tagset des ARK-Taggers gemäss Definition von [Gimpel et al., 2011]

Input Tweet	Tokenisiert und getagged
@iFlak @mozidesigner Haha. <3 Can't wait for BL2, and Gears3 remains excellent. Hope you both enjoy ME3! :)	@iFlak/@ @mozidesigner/@ Haha/! ./, <3/E Can't/V wait/V fo/Pr BL2/^ ,/, and/& Gears3/^ remains/V excellent/A ./, Hope/V you/O both/D enjoy/V ME3/^ !/, :)/E

Tabelle 5: Input und Output des ARK-Taggers.

In ihrem Versuch trainierten sie den Classifier mit 1'000 annotierten Tweets (14'542 Token) und evaluierten mit weiteren 500 Tweets (7124 Token) [Gimpel et al., 2011]. Sie erreichten dabei eine Accuracy von 89.37% korrekt annotierten Tweets. Der Tagger ist frei auf ihrer Website verfügbar[7]. Er liest ein `.txt` File mit je einem Tweet pro Zeile ein und gibt ein TSV-File ()Tabulator-Separated-Values) mit vier Spalten zurück: Dem originalen Tweet, dem tokenisierten Tweet, den POS-Tags für den tokenisierten Tweet und die Sicherheit (ein Wert zwischen 0 und 1), mit der der Classifier jedes einzelne Token getaggt hat. In Tabelle 5 findet sich ein Beispieltweet, der tokenisiert und getaggt wurde.

[7] `http://www.cs.cmu.edu/~ark/TweetNLP/`

3.2 Supervisiertes Maschinelles Lernen

Das Ziel von supervisiertem Maschinellen Lernen ist die automatische Klassifikation von Daten [Domingos, 2012]. Das System benötigt üblicherweise ein Feature-Set und gibt dann für die jeweilige Einheit die Klasse zurück. Ein einfaches Beispiel ist ein Spam-Filter, der Emails in die Klassen Spam und kein Spam einteilt. Ein Feature-Set könnte in diesem Fall folgendermassen aussehen:

```
x = (AbsenderBekannt = True, AbsenderAufBlacklist = False, ...
```

Die Featuresets werden beim supervisierten Lernen automatisch aus den Trainingsdaten erzeugt. Für den Erfolg eines solchen Classifiers muss das Featureset die Trainingsdaten möglichst optimal repräsentieren [Domingos, 2012], damit fundierte Entscheidungen getroffen werden können. Allerdings müssen auch die Daten selbst repräsentativ und in möglichst grosser Menge vorhanden sein, da sonst die Problematik von Overfitting droht: Der Classifier ist in diesem Fall zu sehr auf das Trainingsset angepasst und führt auf neuen Daten angewendet zu schlechten Resultaten.

Evaluiert werden solche Classifier, indem man sie ein bereits von menschlichen Experten annotiertes Test-Set klassifizieren lässt. Aus dem Vergleich von Annotation und Klassifikation kann dann der Classifier bewertet werden.

3.3 Feature Generation

Ich verwende in meinem Experiment den NaiveBayes Classifier von NLTK [Bird et al., 2009]. Um den Classifier zu trainieren und zu evaluieren, müssen die Daten korrekt formatiert sein, nämlich in einer Liste aus Tupeln, die jeweils an erster Stelle ein Dictionary mit den Features beinhalten und an zweiter Stelle die Klasse. Ein Eintrag aus dieser Liste sieht z.B. folgendermassen aus:

```
({'and': 1, 'Arcade': 1, '2nd': 1, 'Much': 1, 'White': 1,
'Foo': 1, 'esp': 1, 'Fire': 1, ',': 1, '/': 1, '.': 1,
'amazing': 1, 'Mick': 1, 'too': 1, 'pos_words': 2, 'was': 1,
'Jack': 1, 'I': 1, 'season': 1, 'repetitions': 0, 'neg_words':
0, 'Tho': 1, '@USER': 1, 'half': 1, 'with': 1, 'has @': 1,
'loved': 1, 'last': 1, 'of': 1, 'Jagger': 1, 'v-polarity': 4.1,
'Fighters': 1}, 'pos')
```

Tweet 1: @the2scoops Much of last season, too, esp 2nd half. Tho I loved Jack White, and Mick Jagger with Arcade Fire / Foo Fighters was amazing.

Dies ist das effektive Featureset für Tweet 1 (aus dem SemEval-Korpus), das in meinem Versuch generiert wird. Die einzelnen Features werden später in diesem Kapitel erläutert. Der Naive Bayes Classifier errechnet nun die bedingten Wahrscheinlichkeiten für die Klassen *positiv*, *negativ* und *neutral* für jedes gegebene Feature. Nehmen wir beispielsweise an, 70 Tweets haben den Wert 2 für das Feature `pos_words`. Davon sind 50 Tweets positiv, während es nur 5 negative und 15 neutrale Tweets gibt. Die bedingte Wahrscheinlichkeit, dass ein Tweet mit `'pos_words': 2` positiv ist, wird mit der Baye'schen Formel berechnet und beträgt

$$P(pos|pos\_words:2) = \frac{P(pos \cap pos\_words:2)}{P(pos\_words:2)} = \frac{50}{70} \approx 0.714 \tag{1}$$

Analog berechnet sich auch die Wahrscheinlichkeit für die anderen beiden Klassen:

$$P(neg|pos\_words:2) = \frac{5}{70} \approx 0.071 \tag{2}$$

$$P(neut|pos\_words:2) = \frac{15}{70} \approx 0.214 \tag{3}$$

Natürlich werden für jedes Feature die bedingten Wahrscheinlichkeiten für die drei Klassen errechnet. Dann werden pro Klasse die einzelnen Wahrscheinlichkeiten multipliziert und damit die Wahrscheinlichkeit, mit der der Tweet jeder Klasse entspricht, berechnet:

$$P(pos|Tweet1) = P(pos|pos\_words:2) * P(pos|and:1) * P(pos|polarity:4.1) * ... \tag{4}$$

$$P(neg|Tweet1) = P(neg|pos\_words:2) * P(neg|and:1) * P(neg|polarity:4.1) * ... \tag{5}$$

$$P(neut|Tweet1) = P(neut|pos\_words:2) * P(neut|and:1) * P(neut|polarity:4.1) * ... \tag{6}$$

Schliesslich werden die drei Wahrscheinlichkeiten verglichen und dem Tweet wird die Klasse mit der höchsten Wahrscheinlichkeit zugewiesen. Das Verfahren trägt den Namen *Naive* Bayes, weil wir annehmen, dass die einzelnen Features voneinander unabhängig sind. In Wahrheit müssen wir aber davon ausgehen, dass z.B. die Anzahl positiver Wörter im Tweet (`pos_words`) und die Polarität (`polarity`) in einer kausalen Beziehung zueinander stehen. Folglich müssten wir auch die Wahrscheinlichkeit für die Klassen schätzen, wenn beide Features präsent sind. Dies ist oft nicht möglich, da enorm grosse Datenmengen dafür notwendig wären. Darum wird

in der Computerlinguistik oft darauf verzichtet und die einfachere, *naive* Formel verwendet. Glücklicherweise können wir mit dieser vereinfachten Vorgehensweise aber dennoch sehr gute Resultate erzielen.

Ich habe viele verschiedene Features ausprobiert, einige erwiesen sich effektiver als andere. Im folgenden präsentiere ich diejenigen Features, welche ich bis in die finale Version meines Projekts beibehalten habe.

3.3.1 Unigram

Eines der denkbar simpelsten Features – und doch sehr wertvoll für Machine Learning – ist das Unigram-Feature. Jedes Token, das im Tweet vorkommt, wird als Feature übernommen. Der Naive Bayes Algorithmus errechnet dann für das Token (ob Wort oder Emoticon ist egal), wie oft es in einem positiven, negativen und neutralen Tweet vorkommt. Falls das Token aber ein Username (beginnend mit @) oder eine URL ist, wird nicht der Name oder die URL als Feature verwendet, sondern lediglich ein Platzhalter (*@USER* oder *URL*).

Token	***pos***	***neg***	***neut***	$P(pos\|token)$	$P(neg\|token)$	$P(neg\|token)$
`:)`	88	1	13	0.86	0.01	0.13
`:(`	1	37	4	0.02	0.88	0.1
`love`	67	3	4	0.91	0.04	0.05
`hate`	1	16	1	0.05	0.9	0.05
`have`	108	113	105	0.33	0.35	0.32

Tabelle 6: Anzahl Vorkommen der Token in positiven, negativen und neutralen Tweets des SemEval-Korpus, sowie die daraus berechneten bedingten Wahrscheinlichkeiten für die jeweilige Klasse gegeben der Token.

Das Emoticon `:)`, welches ein lächelndes Gesicht darstellt und darum mit positiven Emotionen konnotiert wird, kommt im SemEval13-Korpus beispielsweise 13 mal in einem neutralen Kontext vor, 88 mal in einem positiven und nur gerade einmal in einem negativen (siehe auch Tabelle 6). Das Vorkommen von `:)` in einem Tweet deutet also stark darauf hin, dass der Tweet positiv zu klassifizieren ist. Dies bestätigt auch der Wert für die Wahrscheinlichkeit, dass ein Tweet positiv ist, wenn er `:)` enthält. Die sogenannte bedingte Wahrscheinlichkeit wird mit Formel 7 berechnet. Dabei steht B für das Vorkommen von `:)` und A für die Klasse "positiv".

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{7}$$

Neutrale Wörter wie zum Beispiel `have`, sind regelmässig über die Klassen verteilt und fallen folglich auch nicht ins Gewicht bei der Klassifizierung, wie in der letzten Zeile von Tabelle 6 zu erkennen ist. In den linken Spalten der Tabelle befinden sich die absoluten Zahlen und

die drei rechten Spalten enthalten die bedingten Wahrscheinlichkeiten. Aus diesen Werten lässt sich erkennen, dass das Vorkommen von `:),` `:(,` `love` und `hate` einen grossen Einfluss auf die Klassifizierung hat, während die bedingte Wahrscheinlichkeit bei `have` für alle drei Klassen $\approx \frac{1}{3}$ beträgt und somit in der Rechnung nicht ins Gewicht fällt.

3.3.2 Bigram

Das Bigram-Feature ist eine Erweiterung des Unigram-Features. Wie der Name schon sagt, werden hier jeweils zwei Token als Feature übernommen. Dies hat den Vorteil, dass Ausdrücke, die aus zwei Wörtern bestehen und eine andere Bedeutung als ihre Einzelteile haben, berücksichtigt werden. Negationen wie `don't love` werden so als negatives Feature erkannt, während bei den Unigrammen `love` unabhängig vom Kontext als positives Feature wirkt. Aus einem Tweet werden die Bigram-Features wie folgt generiert:

Tweet: How can you hate on YOLO
Features: `How can, can you, you hate, hate on, on YOLO`

Die Bigram-Features haben aber verglichen mit den Unigram-Features den Nachteil, dass es viel seltener vorkommt, dass dieselben zwei Wörter hintereinander stehen. Daraus folgt, dass Bigramme nur dann nützlich sind, wenn die Menge an Tweets gross genug ist. Ansonsten droht die Problematik von Overfitting.

3.3.3 Polarität

Für das Polaritäts-Feature habe ich mir das Emotionslexikon des VADER-Projekts zunutze gemacht. Es umfasst über 7'500 Einträge, welche mit den in den Tweets enthaltenen Token abgeglichen werden. Die Werte der Polarität, die negativ sind für negative und positiv für positive Tweets, werden addiert. Die Summe wird dann als Feature verwendet. Falls kein Token des Tweets auch im Lexikon vorkommt, wird Null als Polaritätswert gesetzt. In Tabelle 7 wird anhand der folgenden drei Beispieltweets aus dem SemEval-Korpus demonstriert, wie die Summe berechnet wird.

Tweet 1: @annie_cutler Good luck tomorrow night. You'll be great. Also, Samoas and Thin Mints = great way to warm up the crowd. #Mmm #SOCOMX

Tweet 2: @PrincessMassey lmao i sat here for five minutes like what the fuck did i do to courtney???? Ha damn.....

Tweet 3: Badly judged joke from Herring. Joke may not "be about rape" but reference to Rohypnol always evocative of such, no matter the intent.

Tweet 1	Good	luck	great	great	warm	**Summe**
Polarität	1.9	2.0	3.1	3.1	0.9	11
Tweet 2	lmao	like	fuck	Ha	damn	**Summe**
Polarität	2.9	1.5	-2.5	1.4	-1.7	1.6
Tweet 3	Badly	joke	Joke	rape	no	**Summe**
Polarität	-2.1	1.2	1.2	-3.7	-1.2	-4.6

Tabelle 7: Alle Token aus den Beispieltweets, die im Lexikon vorkommen und ihre Polaritätswerte.

Der erste Tweet wurde von den Annotatoren als positiv eingestuft, der zweite und dritte Tweet als negativ. Im Falle des ersten und dritten Tweets sind das gute Neuigkeiten, da das Vorzeichen der Zahl mit der Klassifikation korreliert. Der berechnete Wert des zweiten Tweets liegt aber über 0 und sollte also anhand dieser Eigenschaft als positiv eingestuft werden. Hier zeigt sich ein allgemeines Problem des Annotierens, denn die Annotation des Tweets als negativ lässt sich anfechten. Der User zeigt sich zwar offenbar genervt über einen Vorfall mit Courtney, die Eröffnung *lmfao* (=laughing my fucking ass of, ich lach mich kaputt) relativiert aber den Ärger und lässt offen, wie der Tweet nun wirklich gemeint ist. Diese Ambiguität wird von der Polaritätssumme ziemlich gut eingefangen, da die negativen und positiven Summanden sich gegenseitig aufheben, sodass ein (vergleichsweise) kleiner Wert übrigbleibt.

3.3.4 Negative/Positive Wörter

Die Wörterlisten funktionieren ganz ähnlich wie das Polaritätslexikon. Insgesamt 2006 positive und 4783 negative Wörter sind darauf vermerkt. Die Token in den Tweets werden mit den Wörterlisten abgeglichen und für jeden Tweet wird die Anzahl positiver und negativer Wörter gezählt. Da keine abgestuften Polaritätswerte vorhanden sind, werden also nur die Anzahl positiver und negativer Wörter im Tweet als je ein einzelnes Feature generiert. Anzumerken ist, dass in den Wörterlisten keine Emoticons und für Twitter typische Abkürzungen wie *lol* vorkommen. So werden z.B. folgende Wörter gematched:

Tweet 1: @sandraa0407 @Stillsanna Dongwoo is like**/pos** a big baby, sooo cute**/pos** :3 I just wanna hug**/pos** him... Is Dongwoo becoming my 3rd Infinite bias**/neg**? *0*
Features: `pos = 3, neg = 1`

Tweet 2: Spain summons Argentine ambassador in oil dispute**/neg** - Spain warned**/neg** Argentina on Friday it risks**/neg** becoming an... URL
Features: `pos = 0, neg = 3`

3.3.5 URL, Hashtag, At

URL's, Hashtags und At-Zeichen kommen in fast jedem Tweet vor. Über die URL's werden Inhalte auf anderen Seiten verlinkt. Das At-Zeichen, welches ursprünglich für E-Mail-Adressen eingeführt wurde, dient als Markierung für den Empfänger eines Tweets. So richtet sich zum Beispiel Tweet 1 an den User namens `LaCombe_865`. Über das At-Zeichen können sich Twitter User gegenseitig Tweets schicken, die aber, wie für die Plattform üblich, öffentlich sichtbar sind. Die Hashtags dienen als Schlagworte, über die der Tweet aufgerufen werden kann. Tweet 2 kann also gefunden werden, wenn nach `#StarWars` gesucht wird. Über die POS-Tags lässt sich schnell ermitteln, ob ein Tweet URL, Hashtag und At enthält. Aus diesem Wissen werden drei weitere Features (mit den boolschen Werten `True` oder `False`) generiert, wie unter den Tweets Beispiel angegeben.

Tweet 1: @LaCombe_865 I'm gonna be in Monroe like all day Sat ! Haha
Features: `URL = False, Hashtag = False, At = True`

Tweet 2: Didn't get a chance to post this the other day. Goodbye Ralph McQuarrie. May you return as a Force ghost to guide young artists... #StarWars
Features: `URL = False, Hashtag = True, At = False`

3.3.6 Repetitionen

In der sogenannten Computer Mediated Communication kommt es oft vor, dass User Zeichen öfters wiederholen, um zum Beispiel einem Wort besonderen Nachdruck zu verleihen. Im untenstehenden Tweet 1 wird das Wort *help* (hilfe!) durch das dreifache vorkommen von *p* betont. Die drei darauffolgenden Ausrufezeichen erfüllen dieselbe Funktion.

Tweet 1: MORNING**!!!** Good im bloody knackered**!!!** Work is not for me today HEL**PPP**!!!!

Tweet 2: so effing tired of my throat hurting**.... ooooo**hh... i just got a crazy craving for a pina colada/banana slushie**!!!!**

Um die Repetitionen von Buchstaben und Zeichen und die damit verbundene, ausgedrückte Emotionalität ebenfalls in die Klassifikation einfliessen zu lassen, habe ich ein Repetitions-Feature generiert. Für jedes Token wird mittels folgendem regulären Ausdruck ermittelt, ob darin 3 oder mehr gleiche, aufeinanderfolgende Zeichen vorkommen:

```
([a-zA-Z])\1{2,}
```

Die Anzahl Token innerhalb des Tweets, die eine Repetition von 3 oder mehr Zeichen beinhalten, wird dann als Feature übernommen. Das Feature wäre also sowohl für Tweet 1 als auch für Tweet 2 jeweils `Repetitionen = 3`.

3.3.7 Ausrufezeichen

Wie im obigen Abschnitt bereits gesehen, sind wiederholte Ausrufezeichen auf Twitter keine Seltenheit. Zusätzlich zu den Repetitionen habe ich ein Ausrufezeichen-Feature implementiert, welches zählt, wieviele Ausrufezeichen pro Tweet vorkommen. Damit möchte ich besonders emotionsgeladene Tweets identifizieren, wie z.B. den folgenden:

Tweet: UNC!!! NCAA Champs!! Franklin St.: I WAS THERE!! WILD AND CRAZY!!!!!! Nothing like it...EVER
Feature: `Anzahl Ausrufezeichen = 13`

3.4 Classifier Trainieren und Anwenden

Wie bereits erwähnt, werden die generierten Features in ein Test- und Trainingsset eingeteilt. Das Trainingsset wird an das NLTK-Modul `NaiveBayesClassifier` weitergegeben, welches daraus den Classifier generiert. Um herauszufinden, wie wichtig die einzelnen Features sind, habe ich mehrere Featuresets generiert und immer ein Feature weggelassen. Dadurch lässt sich der Einfluss der verschiedenen Features auf die Performanz des Systems leicht ablesen.

Sind die verschiedenen Classifier erst einmal trainiert, können sie auf das Testset angewendet werden. Diese Arbeit übernimmt ebenfalls ein Modul von NLTK. Da ich jedoch relativ kleine Datenmengen habe, wollte ich das Ergebnis meines Versuchs nicht dem Zufall überlassen, und habe deshalb 10-fache Kreuzvalidation verwendet. Das bedeutet also, dass ich das Featureset jeweils in 10 gleiche Teile geteilt habe, und den gesamten Prozess des Classifier trainieren und evaluieren 10 mal wiederholt habe. Dabei war jeder der Zehntel des Featureset einmal das Trainingsset, während die jeweils anderen 90% als Trainingsset gebraucht wurden. Die 10 verschiedenen Messwerte wurden dann gemittelt. Dadurch können starke Schwankungen im Resultat, die durch günstige oder ungünstige Unterteilung des Featuresets entstehen, abgeschwächt werden.

4 Evaluation

4.1 Vorkommen der Features

Tabelle 8 fasst die verwendeten Features nochmal zusammen. Zudem sind jeweils die Anzahl Token angegeben, in denen dieses Feature vorkommt. Damit die Werte verglichen werden können, habe ich neben dem absoluten Vorkommen jeweils angegeben, in wievielen Prozent der Token die Features vorkommen. Für die Bigramme ist diese Information nicht relevant und fehlt deshalb in der Tabelle. Ausserdem ist der Wert für Unigramme natürlich immer 100%.

Es ist ersichtlich, dass die Features in allen 3 Korpora ungefähr gleich stark vertreten sind. Mit 7 bis knapp 10% ist das Polaritäts-Feature am häufigsten vertreten (abgesehen von den Unigrammen). Die Werte für die positiven und negativen Wörter, für das At-Zeichen und die Ausrufezeichen bewegen sich zwischen 2 und 4%. Am Ende der Liste stehen Hashtag, URL und Repetitionen mit Werten zwischen 0.1 und 1.1%. Ein Feature ist aussagekräftiger, je öfters es in den Trainingsdaten vorkommt. Es ist also wahrscheinlich, dass die Features mit höherem Vorkommen relevanter sind.

Feature	**SemEval13** (n=3'204)		**Sentiment140** (n=2'000)		**Sentiment140** (n=20'000)		**Beschreibung**
Unigram	71'293	100%	30'225	100%	301'668	100%	Anzahl Token
Bigram	n/a	n/a	28'227	-	281'670	-	Anzahl Bigramme
Polarität	4'991	7%	2'958	9.8%	29'111	9.7%	Anzahl Token, die Polaritätslexikon matchen
Wörter_p	1'985	2.8%	1'201	4%	12'744	4.2%	Anzahl Token, die pos. Wörterliste matchen
Wörter_n	1'460	2%	1'040	3.5%	9'476	3.1%	Anzahl Token, die neg. Wörterliste matchen
At	2'183	3%	1'053	3.5%	9'668	3.2%	Anzahl Token, die mit @ getaggt wurden.
Hashtag	673	0.9%	25	0.1%	215	0.1%	Anzahl Token, die mit # getaggt wurden.
URL	438	0.6%	94	0.3%	1'047	0.3%	Anzahl Token, die mit *U* getaggt wurden.
Rep.	756	1.1%	n/a	n/a	n/a	n/a	Anzahl Token mit mindestens dreifacher Repetition
!-Zeichen	n/a	n/a	781	2.6%	7'346	2.4%	Anzahl Token mit mindestens 1 Ausrufezeichen

Tabelle 8: Übersicht der Features und Vorkommen in den Korpora. Links in der Spalte steht jeweils das absolute Vorkommen und rechts das Vorkommen im Verhältnis zur Gesamtmenge der Token. n gibt die Anzahl Tweets pro Korpus an. n/a steht für Features, die im jeweiligen Korpus nicht benutzt wurden.

4.2 Messwerte

Zur Evaluation meines Systems verwende ich folgende Standardmessungen[8]:

- **Accuracy:**
  Die Accuracy beschreibt die Genauigkeit des Systems. Sie misst, wieviele Tweets korrekt klassifiziert wurden. Der NLTK-Classifier verfügt über eine Methode, mit der dieses Mass automatisch berechnet werden kann. Die Accuracy lässt sich wie in Formel (8) beschrieben berechnen. Das Resultat ist immer ein Wert zwischen 0 und 1, wobei 1 einer perfekten Klassifikation entspricht und 0 einer komplett falschen.

$$Accuracy = \frac{|KorrektKlassifiziert|}{|KorrektKlassifiziert| + |FalschKlassifiziert|} \tag{8}$$

- **Precision:**
  Die Precision misst, wie korrekt die Klassifikation innerhalb einer Klasse ist. Wir berechnen also für alle Klassen i die Precision separat. Wie in Formel (9) dargestellt, werden alle korrekt der Klasse i zugewiesenen Tweets (True Positives = TP) durch die gesamte Anzahl der Tweets (auch den False Positives = FP), die Klasse i zugewiesen wurden, geteilt. Auch hier erhalten wir einen Wert zwischen 0 und 1.

$$Precision_i = \frac{TP_i}{TP_i + FP_i} \tag{9}$$

- **Recall:**
  Der Recall ist das Gegenstück zur Precision und misst, wieviele Instanzen einer Klasse gefunden wurden. Wie Formel (10) zeigt, kommen hier die False Negatives (FN), also alle Tweets, die fälschlicherweise nicht der Klasse i zugeordnet wurden, zum Zug. Wie bei Accuracy und Precision ist auch beim Recall 1 der höchste, und 0 der tiefstmögliche Wert.

$$Recall_i = \frac{TP_i}{TP_i + FN_i} \tag{10}$$

- **F-Mass:**
  Das F-Mass vereint die Information aus Precision und Recall zu einem harmonischen Mittel. Ich werde im folgenden die Precision und den Recall nicht aufführen, sondern mich auf die F-Masse F_{pos}, F_{neg} und F_{neut} beschränken.

$$F_i = 2 * \frac{Precision_i * Recall_i}{Precision_i + Recall_i} \tag{11}$$

[8] `https://en.wikipedia.org/wiki/Precision\_and\_recall`

- **McNemar's Test:**
  Um zu ermitteln, ob Unterschiede in der Accuracy der verschiedenen Classifier (alle Features vs. ein weggelassenes Feature) signifikant sind, wende ich McNemar's Test an. Dafür muss für jeden Tweet gespeichert werden, welcher Classifier ihn richtig annotiert hat, und welcher nicht. Mit dieser Information können wir dann eine 2x2 Tabelle erstellen, die zeigt, wieviele Features von beiden Classifiern richtig oder falsch annotiert wurden (a und d) und wieviele Features jeweils von einem falsch, und vom andern richtig annotiert wurden (b und c):

	korrekt C1	falsch C1
korrekt C2	a	b
falsch C2	c	d

	korrekt C1	falsch C1
korrekt C2	14'049	1'023
falsch C2	1'151	3'775

Tabelle 9: Beispiel für 2x2 Tabellen der Classifier C1 und C2.

Wenn zwischen den Classifiern kein (signifikanter) Unterschied besteht, gilt $a+b \approx a+c$ und $d+b \approx d+c$. Aus diesen Gleichungen können wir jeweils a und d kürzen. Aus den Wahrscheinlichkeiten des Auftretens (p) der unterschiedlich klassifizierten Mengen b und c ergeben sich Null- und Alternativhypothese: $H_0 : p_b = p_c$ und $H_1 : p_b \neq p_c$. Mit Formel (12) berechnen wir nun den χ^2 Wert. Da b und c recht klein sein können, habe ich die Edwards-Korrektur verwendet, um Approximationsfehler zu vermeiden, wie es auch von [Dietterich, 1998] empfohlen wird.

$$\chi^2 = \frac{(|b-c|-1)^2}{b+c} = \frac{(|1023-1151|-1)^2}{1023+1151} \approx 7.42 \tag{12}$$

Der berechnete Wert kann nun mit einer χ^2-Tabelle abgeglichen werden. In diesem Beispiel: $7.88 \geq$**7.42** ≥ 6.63 (Tabelle 10). Daraus folgt, dass die Nullhypothese H_0 bei einem Signifikanzniveau von 1% verworfen werden muss. Die Alternativhypothese H_1 muss also angenommen werden, die beiden Classifier C1 und C2 sind signifikant unterschiedlich.

p	0.9	0.95	0.975	0.99	0.995
χ^2	2.71	3.84	5.02	6.63	7.88

Tabelle 10: Auszug aus der Tabelle der χ^2-Verteilung[9]bei einem Freiheitsgrad von 1.

[9] https://de.wikibooks.org/wiki/Statistik:_Tabelle_der_Chi-Quadrat-Verteilung

4.3 SemEval13

Für das SemEval13 Korpus habe ich zwei Versuche durchgeführt.

In Tabelle 11 und Abbildung 3 (alle Abbildungen befinden sich in Sektion 4.5) sind die Messwerte für den Versuch mit nur zwei Klassen, *positiv* und *negativ* aufgeführt. Mit einer Accuracy von 83.81% erzielt der Versuch (alle Features) das beste Machine Learning Resultat in dieser Arbeit. Aus der Abbildung ist auf den ersten Blick ersichtlich, dass das Fehlen des Unigram-Features den grössten Einbruch in den Werten der F-Masse und Accuracy zur Folge hat. Mit einem χ^2 Wert von 31.4 ist der Unterschied hoch signifikant ($p > 0.995$). Ebenfalls signifikant sind die Unterschiede, wenn die Wörterlisten ($p > 0.995$) oder die Polarität ($p > 0.95$) in den Features fehlen. Der Classifier ohne Hashtag/URL/At hat lediglich ein Signifikanzniveau von 10%, was ich als zu gering betrachte. Die Repetitionen sind ganz klar nicht signifikant, der Wert der Accuracy ändert sich hier überhaupt nicht.

Ausserdem fällt auf, dass die Werte von F_{pos} und F_{neg} immer sehr nah beieinander sind, ausser wenn die Unigramme weggelassen werden: In diese Fall sinkt der Wert von F_{neg} auf 76.81%. Offenbar lassen sich negative Tweets sehr gut mit Unigram-Features klassifizieren.

$\lvert Tweets\rvert = 2'136$	F_{pos}	F_{neg}	A	χ^2
Alle Features	83.95%	83.70%	83.81%	
-Repetition	83.92%	83.71%	83.81%	0.08
-Unigram	80.42%	76.81%	78.70%	31.4
-Polarität	82.73%	82.69%	82.68%	4.34
-Wörterlisten	82.01%	81.85%	81.93%	12.7
-Hashtag/URL/At	82.92%	83.40%	83.15%	3.13

Tabelle 11: F_i Werte und Accuracy des SemEval13-Korpus, mit den Klassen positiv und negativ. Der χ^2 Wert bezieht sich auf die Unterschiede in der Klassifikation.

$\lvert Tweets\rvert = 3'204$	F_{pos}	F_{neg}	F_{neut}	A	χ^2
Alle Features	67.93%	70.69%	58.73%	66.04%	
-Repetition	68.02%	70.73%	58.69%	66.07%	0.00
-Unigram	62.70%	65.26%	60.62%	62.64%	13.1
-Polarität	66.23%	68.66%	55.83%	64.01%	14.6
-Wörterlisten	66.69%	69.14%	56.55%	64.51%	9.48
-Hashtag/URL/At	68.17%	70.67%	57.44%	65.86%	0.34

Tabelle 12: F_i Werte und Accuracy des SemEval13-Korpus, mit den Klassen positiv, negativ und neutral. Der χ^2 Wert bezieht sich auf die Unterschiede in der Klassifikation.

In Tabelle 12 und Abbildung 4 sind die Werte für den Versuch mit den drei Klassen *positiv*, *negativ* und *neutral* enthalten. Die Einführung einer dritten Klasse hat die Accuracy für alle Features auf 66.04% gesenkt. Man muss aber auch beachten, dass die Majority Baseline, also

die Accuracy, wenn Tweets zufällig getaggt würden, für zwei Klassen bei 50% liegt, und für 3 Klassen bei 33% (bei gleich grossen Klassen). Die Verbesserung im Verhältnis zur Baseline ist also prozentual gesehen gleich gross. Ein möglicher Grund für die Verschlechterung ist aber auch, dass die neutrale Klasse in allen Anordnungen durchgehend sehr schlecht erkannt wird, wie die Werte von F_{neut} zeigen, während F_{neg} oft über 10% höher ist. Dies liegt wahrscheinlich der Tatsache zu Grunde, dass neutrale Tweets keine so markanten Features besitzen, wie positive und negative es tun. Die Absenz von emotionsreicher Sprache wird also vom Featureset nicht gut repräsentiert. Ein Feature, welches sachlichere Tweets erkennt, könnte den F_{neut}-Wert möglicherweise anheben.

Im SemEval 2013 Task 2 (ebenfalls mit 3 Klassen, unter Verwendung desselben Trainingssets) erzielte die beste Einreichung 69.02%, die zweitbeste 65.27% [Nakov et al., 2013]. Da die Teilnehmer ein separates Testset verwendeten, sind die Ergebnisse nicht direkt vergleichbar. Allerdings scheint mein System ebenfalls schon ein sehr hohes Niveau erreicht zu haben.

Wenn wir die χ^2 Werte betrachten, ergibt sich ein ähnliches Bild wie in Tabelle 11. Das Fehlen von Unigram, Polarität und den Wörterlisten ist hoch signifikant, während Differenzen in der Accuracy ausgelöst durch Repetitionen und Hashtag/URL/At vernachlässigbar sind. Durch die Einführung von neutralen Tweets wurde aber neu die Polarität zum signifikantesten Feature, obwohl die Differenz der Accuracy bei den Unigrammen höher ist. Der Unterschied der beiden χ^2 Werte ist zwar nicht von grosser Bedeutung, aber offenbar resultiert das Weglassen der Polarität als Feature darin, dass eine Umverteilung der Klassifikation der Tweets stattfindet. D.h. viele Tweets werden anders klassifiziert, wobei sich die Anzahl korrekter und inkorrekter Klassifikationen zu einem grossen Teil kompensiert, was natürlich stark abweichende Werte b und c in McNemar's Test verursacht.

4.4 Sentiment140

Mit dem Sentiment140 Korpus habe ich ebenfalls zwei Versuche durchgeführt, diesmal mit zwei unterschiedlich grossen Datensets, aber denselben Features.

Tabelle 13 und Abbildung 5 bieten eine Übersicht der Performanz des Systems mit n=2'000 Tweets. Die Accuracy erreicht nur knapp 70% und die F-Masse sind über alle Kategorien sehr unterschiedlich, wobei der Wert F_{neg} fast immer höher liegt als F_{pos}. Die Tweets im Sentiment140 Korpus scheinen sich also sehr stark von denen des SemEval13 Korpus zu unterscheiden. Ein möglicher Grund dafür ist, dass in Sentiment140 keine Emoticons enthalten sind, die (innerhalb des Unigram Features) stark ins Gewicht fallen und deren Polarität ebenfalls oft mit der des Tweets korreliert.

Wenn wir uns die χ^2 Werte ansehen, kommen wir auf dieselben Schlüsse wie schon bei SemEval13: Unigram, Polarität und Wörterlisten sind signifikant, die restlichen Features nicht.

Anzumerken ist insbesondere, dass die beiden neuen Features, Ausrufezeichen und Bigram zu dieser nicht signifikanten Gruppe gehören. Ich vermute, dass Ausrufezeichen ungefähr gleich oft in negativen wie in positiven Tweets verwendet werden, um Emotionen auszudrücken. Das Problem bei den Bigrammen ist, wie im letzten Kapitel bereits angesprochen, die kleine Anzahl an Tweets, die wir hier betrachten.

Im grossen Testset mit n=20'000 Tweets (Tabelle 14 und Abbildung 6) schneiden die Bigramme schon viel besser ab. Mit einem χ^2 Wert von 7.42 ist der Unterschied zu allen Features bereits hoch signifikant. Auch insgesamt verbessert sich die Accuracy um 6% im Vergleich zum kleineren Set. Neben Unigram, Polarität und Wörterlisten ist hier auch das Hashtag/URL/At Feature signifikant. Grosse Datenmengen sind also offenbar Voraussetzung für den erfolgreichen Einsatz dieses Features. Die Abbildung verrät, dass auch in diesem Versuch F_{neg} weitaus besser abschneidet als F_{pos}.

$\lvert Tweets\rvert = 2'000$	F_{pos}	F_{neg}	A	χ^2
Alle Features	68.78%	71.24%	69.97%	
-Ausrufezeichen	68.55%	70.70%	69.57%	1.44
-Unigram	68.39%	67.81%	68.02%	4.73
-Polarität	67.29%	69.72%	68.47%	8.41
-Wörterlisten	67.17%	69.78%	68.42%	10.3
-Hashtag/URL/At	67.80%	71.23%	69.52%	2.06
-Bigram	70.03%	70.38%	70.12%	0.02

Tabelle 13: F_i Werte und Accuracy des kleinen Sentiment140-Teilkorpus. Der χ^2 Wert bezieht sich auf die Unterschiede in der Klassifikation.

$\lvert Tweets\rvert = 20'000$	F_{pos}	F_{neg}	A	χ^2
Alle Features	74.92%	76.99%	76.01%	
-Ausrufezeichen	74.86%	76.89%	75.93%	0.95
-Unigram	73.46%	74.68%	74.09%	81.3
-Polarität	73.90%	76.45%	75.25%	34.4
-Wörterlisten	74.00%	76.45%	75.29%	34.2
-Hashtag/URL/At	74.06%	77.04%	75.65%	14.1
-Bigram	75.02%	75.68%	75.37%	7.42

Tabelle 14: F_i Werte und Accuracy des grossen Sentiment140-Teilkorpus. Der χ^2 Wert bezieht sich auf die Unterschiede in der Klassifikation.

4.5 Abbildungen

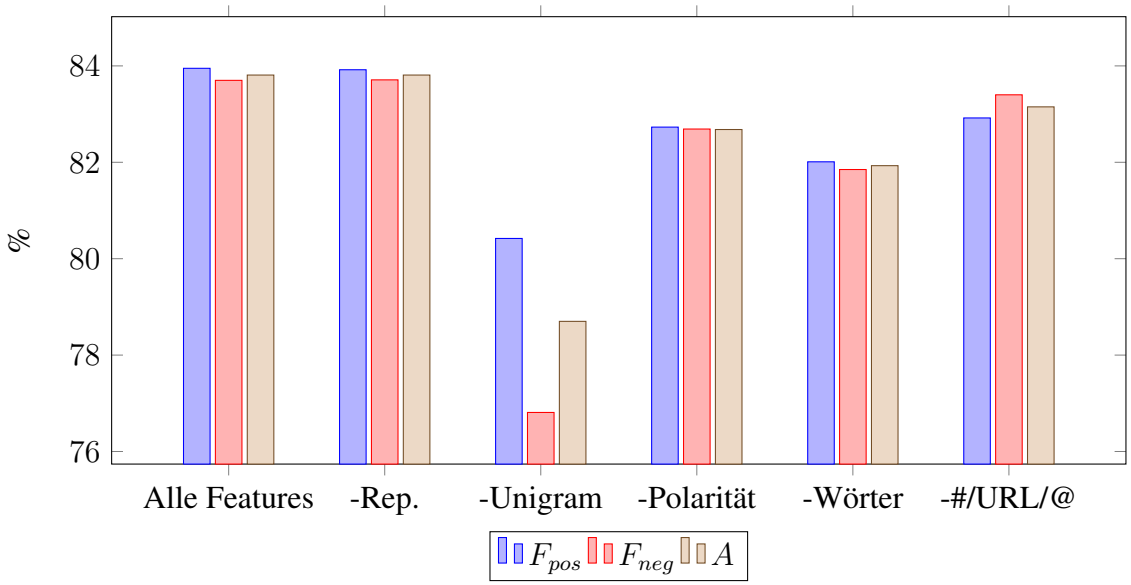

Abbildung 3: Darstellung der Werte aus Tabelle 11 als Balkendiagramm.

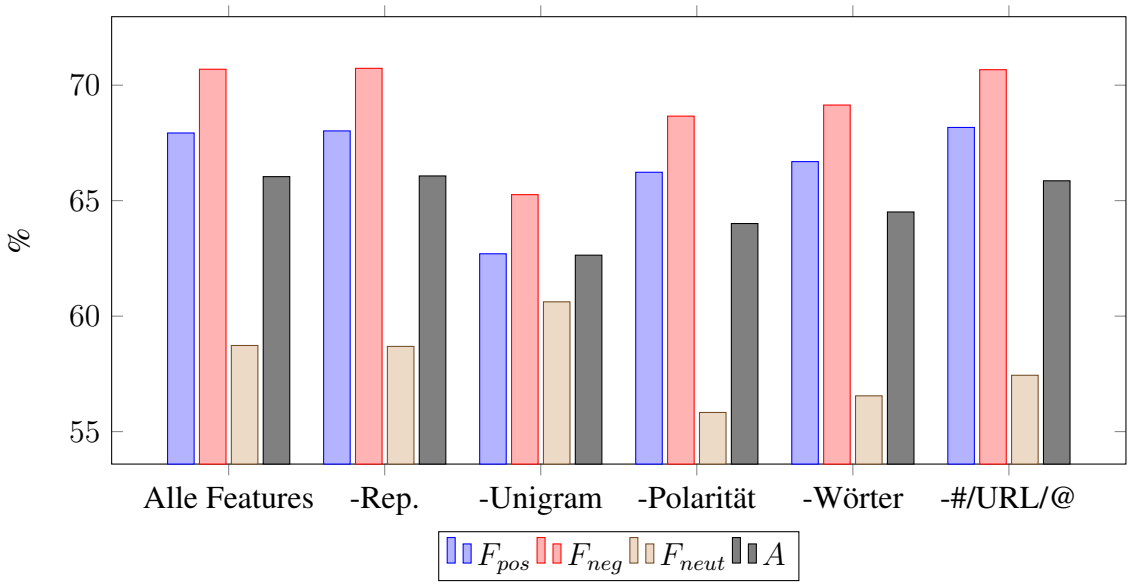

Abbildung 4: Darstellung der Werte aus Tabelle 12 als Balkendiagramm.

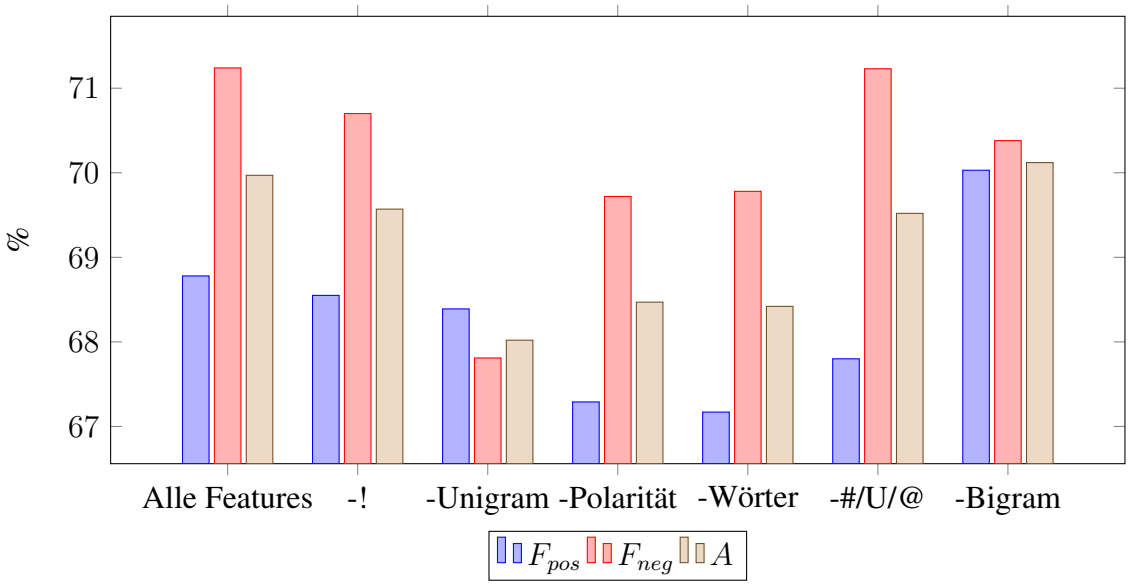

Abbildung 5: Darstellung der Werte aus Tabelle 13 als Balkendiagramm.

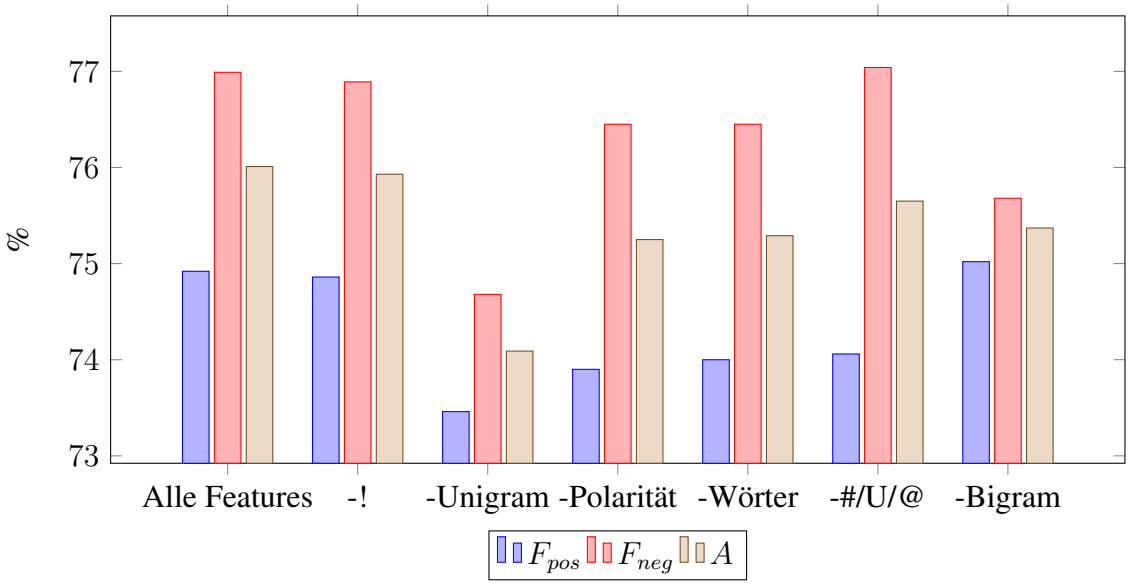

Abbildung 6: Darstellung der Werte aus Tabelle 14 als Balkendiagramm.

5 Fazit

5.1 Zusammenfassung

Die Versuche mit dem SemEval13 Korpus sind (bei zwei Klassen) massiv erfolgreicher als mit Sentiment140, was einerseits an den nicht vorhandenen Emoticons in letzterem Korpus liegt, andererseits daran, dass die Daten in SemEval13 homogener sind, da sich die Tweets auf die selben Themen beschränken. Ausserdem wurden ambige Tweets, bei denen die Annotation nicht eindeutig gemacht werden konnte, aus dem Korpus ausgeschlossen, während eine solche Selektion in Sentiment140 nicht stattgefunden hat.

Bei kleineren Datenmengen punkten insbesondere die Unigram-, Polaritäts- und Wörterlisten-Features. Wird das Datenvolumen erhöht, fallen auch das Bigram- und das Hashtag/URL/At-Feature ins Gewicht und die Accuracy erhöht sich. Dies entspricht den allgemeinen Erkenntnissen im Maschinellen Lernen [Domingos, 2012], dass mehr Daten bessere Classifier hervorbringen. Wie zu erwarten ist die Performanz ebenfalls besser, wenn statt drei nur zwei Klassen verwendet werden. Ein Feature, welches die Objektivität von neutralen Tweets besser einfängt, könnte dem Abhilfe verschaffen.

5.2 Ausblick

Sentimentanalyse für Twitter bleibt auch weiterhin ein bedeutendes Forschungsgebiet in der Computerlinguistik. Der diesjährige SemEval Workshop hat erneut einen Task dazu ausgeschrieben [Nakov et al., 2016]. Neben der Klassifikation in die drei Klassen *positiv*, *negativ* und *neutral* wird dieses Jahr aber auch geforscht, wie Tweets am besten auf einer fünfstelligen Skala von *sehr negativ* bis *sehr positiv* klassifiziert werden können, da Sentiment natürlicherweise in verschiedenen Abstufungen auftritt. Die fünfstellige Skala wurde vorgeschlagen, da im Internet z.B. Filme oder Produkte oft mit einer Anzahl von 1 bis 5 Sternen bewertet werden können.

Mit mehr vorhandenen, annotierten Daten, und raffinierteren Maschinellen Lernverfahren wird Sentimentanalyse für Twitter immer besser. Nicht nur für Meinungsforschung lässt sich Sentimentanalyse einsetzen, sondern auch um Trends vorauszusagen, die Stimmung z.B. vor politischen Wahlen abzubilden oder auf den User zugeschnittene Werbung zu schalten.

In meiner Arbeit habe ich gezeigt, dass mit relativ einfachen Mitteln bereits sehr gute Ergebnisse in der Sentimentanalyse für Twitter erzielt werden können. Dank Polaritätslexika, die auf Microblogging zugeschnitten sind und der wachsenden Anzahl an annotierten Twitter-Korpora, ist dieser Bereich der Sentimentanalyse bereits leistungsfähig und wird auch in Zukunft bedeutend sein.

6 Literatur

[Bird et al., 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media.

[Castellucci et al., 2013] Castellucci, G., Filice, S., Croce, D., and Basili, R. (2013). Unitor: Combining syntactic and semantic kernels for twitter sentiment analysis. In *Joint Conference on Lexical and Computational Semantics*.

[Dietterich, 1998] Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10.

[Domingos, 2012] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*.

[dos Santos and Gatti, 2014] dos Santos, C. N. and Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*.

[Fort et al., 2011] Fort, K., Adda, G., and Cohen, K. B. (2011). Amazon mechanical turk: Gold mine or coal mine? *Computational Linguistics*, 37.

[Gimpel et al., 2011] Gimpel, K., Schneider, N., O'Connor, B., Das, D., Mills, D., Eisenstein, J., Heilman, M., Yogatama, D., Flanigan, J., and Smith, N. A. (2011). Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of Association for Computational Linguistics 2011.*, Stroudsburg, PA, USA.

[Go et al., 2009] Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *Processing*, pages 1–6.

[Hutto and Gilbert, 2014] Hutto, C. J. and Gilbert, E. E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eigth International Conference on Weblogs and Social Media*, Ann Arbor, MI, USA.

[Illecker, 2015] Illecker, M. (2015). Real-time sentiment classification based on apache storm. Master's thesis, Leopold-Franzens-University Innsbruck.

[Kouloumpis et al., 2011] Kouloumpis, E., Wilson, T., and Moore, J. (2011). Twitter sentiment analysis: The good the bad and the omg! In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*.

[Liu et al., 2005] Liu, B., Hu, M., and Cheng, J. (2005). Opinion observer: Analyzing and comparing opinions on the web. In *Procceedings of the 14th International World Wide Web conference.*

[Mohammad et al., 2013] Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013). Nrc-canada: Building the state-of-the-art in sentiment analysis of twitter. In *Second Joint Conference on Lexical and Computational Semantics*.

[Nakov et al., 2013] Nakov, P., Kozareva, Z., Ritter, A., Rosenthal, S., Stoyanov, V., and Wilson, T. (2013). Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics, Volume 2*.

[Nakov et al., 2016] Nakov, P., Ritter, A., Rosenthal, S., Sebastiani, F., and Stoyanov, V. (2016). Semeval-2016 task 4: Sentiment analysis in twitter. Draft.

[Pang and Lee, 2002] Pang, B. and Lee, L. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing.*

[Philips, 1990] Philips, L. (1990). Hanging on the metaphone. *Computer Language*, 7.

[Ritter et al., 2012] Ritter, A., Mausam, O. E., and Clark, S. (2012). Open domain event extraction from twitter. In *Proceedings of the 18th AMC SIGKDD international conference on Knowledge discovery and data mining*, Beijing.

[Spencer and Uchyigit, 2012] Spencer, J. and Uchyigit, G. (2012). Sentimentor: Sentiment analysis of twitter data. In *CEUR Workshop Proceedings*.

7 Python Skripte

I **`sentiment_analyis.py`**

Dieses Skript generiert Features, trainiert darauf den Classifier und evaluiert ihn. Für jede einzelne Klassifikation wird in ein Output-File geschrieben, ob sie korrekt war.

Input: Tweet-Korpus (z.B. `semeval_pos_neg.txt`) und die Lexika `pos_wordlist.txt`, `neg_wordlist.txt` und `vaderlex.csv`.

Output: Accuracy und F-Scores werden in die Konsole geschrieben. In einem Output-File wird Korrektheit der einzelnen Klassifikationen (`True/False`) festgehalten.

II **`create_mcnemar_table.py`**

Das Skript vergleicht die Performanz zweier Classifier, gibt eine McNemar Tabelle und den daraus berechneten χ^2-Wert aus.

Input: Zwei der Output-Files des `sentiment_analyis.py` Skripts.

Output: McNemar Tabelle und χ^2-Wert werden in die Konsole geschrieben.