

Seminararbeit

# Erkennen von Emotionen in Tweets

Adrian Schmid - schmiad1@students.zhaw.ch

Zürich, 16.06.2014

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>2</b>
1.1	Thema, Motivation und Ziel der Arbeit . . . . .	2
1.2	Planung . . . . .	3
<b>2</b>	<b>Grundlagen</b>	<b>4</b>
2.1	Algorithmen zur Erkennung der Gefühlslage in Texten . . . . .	4
2.2	Twitter Search API . . . . .	8
<b>3</b>	<b>Umsetzung</b>	<b>10</b>
3.1	Benutzerführung . . . . .	10
3.2	Twitterdaten Sammeln . . . . .	11
3.3	Algorithmen zur Erkennung der Gefühlslage in Texten . . . . .	12
3.4	Parallelisierung . . . . .	16
<b>4</b>	<b>Analyse</b>	<b>18</b>
4.1	Vergleich der Algorithmen . . . . .	18
4.2	Parallelisierung . . . . .	21
<b>5</b>	<b>Fazit</b>	<b>22</b>

# 1 Einleitung

## 1.1 Thema, Motivation und Ziel der Arbeit

Als Basis für viele soziale Netzwerkdatenanalysen ist es wichtig erkennen zu können welche Gefühle mit bestimmten Textfragmenten verknüpft werden. Es gibt verschiedene Ansätze und Algorithmen um dies zu erreichen. Im Rahmen dieser Seminararbeit möchte ich mich mit verschiedenen Algorithmen und Ansätzen zum Erkennen einer negativen oder positiven Grundhaltung beziehungsweise Grundstimmung in kurzen Textfragmenten festzustellen auseinandersetzen.

Ziel der Arbeit ist es zuerst verschiedene Verfahren zu untersuchen und kennenzulernen um danach entscheiden zu können welche Verfahren im Rahmen dieser Arbeit implementiert werden könnten. Schlussendlich sollen die umsetzbaren Verfahren implementiert werden.

Die Implementation soll dabei folgende Funktionalität aufweisen:

1. In einem ersten Schritt wird ein Thema erfasst. Ein Thema beinhaltet einen „Titel“, „Stichwörter“ und einem Zeitrahmen. Nach diesen Attributen sollten Tweets auf Twitter gefunden werden können.
2. Das Programm soll mit diesen Daten relevante Tweets herauslesen.
3. Nun sollen Parallel verschieden Algorithmen angestossen werden welche entscheiden ob die definierten Tweets eher positiv oder eher negativ sind.
4. Dem Benutzer wird ausgegeben welche Algorithmen wieviele Tweets aus der zuvor zusammengestellten Menge „positiv“ oder „negativ“ bewerten.

## 1.2 Planung

Informationen beschaffen & Einarbeitung in Twitter API und Algorithmen	13.03.2014 – 20.03.2014
Erstellen eines Konzepts, Festlegen welche Algorithmen implementiert werden sollen bzw können.	20.03.2014 – 02.04.2014
<b>Milestone 1: Konzept erstellt</b>	<b>02.04.2014</b>
Analyse und Design der Software	03.04.2014 – 16.04.2014
Implementation	17.04.2014 – 07.05.2014
Testing / Dokumentation	08.05.2014 – 14.05.2014
<b>Milestone 2: Implementation abgeschlossen</b>	<b>14.05.2014</b>
Vergleich der Ergebnisse	15.05.2014 – 21.05.2014
Dokumentation	22.05.2014 – 28.05.2014
Endkorrektur / Abschluss	29.05.2014 – 16.06.2014
<b>Milestone 3: Arbeit fertig</b>	<b>16.06.2014</b>
Abgabe der Arbeit	16.06.2014
Präsentation	19.06.2014

Tabelle 1: Projektplanung

## 2 Grundlagen

### 2.1 Algorithmen zur Erkennung der Gefühlslage in Texten

Es wird grundsätzlich zwischen zwei Ansätzen zur Analyse der Gefühlslage in Texten unterschieden. Der erste Ansatz verwendet machine-learning Technologien der zweite basiert auf der lexikalischen Analyse der Texte. Für den ersten Ansatz wird ein Korpus mit gelabelten Trainingsdaten benötigt. [15] Der Hauptvorteil von machine-learning basierten Methoden ist, dass sie sich an neue Gegebenheiten anpassen können. Der grosse Nachteil ist, dass keine gelabelten Trainingsdaten für neue Themen existieren und dass das erstellen eines neuen Datenkorpus sehr aufwändig und teuer ist. Die lexikalischen Methoden haben eine vordefinierte Liste von Wörtern welche mit bestimmten Gefühlen verknüpft sind. Um gute Resultate zu erhalten, müssen diese Wortlisten der entsprechenden Domäne angepasst werden. So werden zum Beispiel im Web-Slang andere Wörter mit anderen Gefühlen verknüpft als in einem formalen Text. [13]

Im Rahmen dieser Arbeit werden sowohl machine-learning basierte als auch lexikalische Algorithmen zur Erkennung der Gefühlslage in Texten untersucht und - wenn in Python umsetzbar - auf Tweets zu verschiedenen Themen angewandt.

#### 2.1.1 Emoticons

Der einfachste Ansatz zum erkennen der Grundhaltung des Authors gegenüber eines Themas ist die Untersuchung der Emoticons die es beinhaltet. Emoticons sind meist Kombinationen von ASCII Zeichen die einen Gesichtsausdruck darstellen. Dieser kann Gefühle unter anderem gefühle wie glücklich oder traurig representieren. Für diesen Algorithmus wurde eine Menge von gängigen Emoticons aus dem Web [1][9][4][6] zusammengesucht und manuell in «positiv», «neutral», und «negativ»klassifiziert. Mithilfe solcher Listen kann die Anzahl Emoticons der entsprechenden Kategorie in einem Text zählen und gegeneinander Abwägen. [13] Die Umsetzung eines solchen Algorithmus in Python stellt keine Probleme dar.

#### 2.1.2 LIWC - Linguistic Inquiry and Word Count

LIWC ist ein Textanalysetool welche emotionale, kognitive und strukturelle Komponenten von Texten mithilfe eines Wörterbuchs klassifiziert. Die LIWC Software wird kommerziell vertrieben und kann - bei Bedarf - um eigene Wörterbücher ergänzt werden. LIWC liest Text ein und generiert daraus ein Tab-Delimited File welches zum Beispiel in einem Python Programm oder mithilfe von SPSS weiter verarbeitet werden könnte.

Es existieren Versionen für Windows und Macintosh Computer. [13][5] Die fehlende Verfügbarkeit für Linux, die komplizierte Anbindung an ein Python Programm und ökonomische Gründe haben dazu geführt, dass im Rahmen dieser Arbeit keine LIWC-Python Anbindung implementiert wurde.

### 2.1.3 SentiStrength

SentiStrength ist ein machine-learning basiertes Textanalyse Tool welches spezialisiert ist auf die Analyse von Texten aus dem Social-Web. Die Freie Version ist auf der Basis des .Net Technologiestacks implementiert und nur für Windows verfügbar. Eine kommerzielle Version des Tools wurde in Java geschrieben. Die kommerzielle Java Version ist für Forschungszwecke frei verfügbar und kann per Email angefordert werden. Diese Version liesse sich wie im Listing 1 aufgezeigt an Python Programme anbinden. [7][13]

```
1 #Alec Larsen – University of the Witwatersrand, South Africa, 2012 import
  shlex, subprocess
2
3 def RateSentiment(sentiString):
4     #open a subprocess using shlex to get the command line string into the
      correct args list format
5     p = subprocess.Popen(shlex.split("java -jar SentiStrength.jar stdin
      sentidata C:/SentiStrength_Data/"), stdin=subprocess.PIPE, stdout=
      subprocess.PIPE, stderr=subprocess.PIPE)
6     #communicate via stdin the string to be rated. Note that all spaces
      are replaced with +
7     stdout_text, stderr_text = p.communicate(sentiString.replace(" ", "+"))
8     #remove the tab spacing between the positive and negative ratings. e.g
      . 1-5 -> 1-5
9     stdout_text = stdout_text.rstrip().replace("\t", "")
10    return stdout_text
```

Listing 1: Python Anbindung an SentiStrength (JAVA)

Der entsprechende Email-Kontakt hat nicht innert nützlicher Frist geantwortet, weshalb dieser Algorithmus aus dieser Arbeit ausgeklammert werden musste.

### 2.1.4 SentiWordNet

Das SentiWordNet ist eine offen verfügbare lexikalische Ressource (Wörterbuch) auf der Basis von WordNet[10] mithilfe welchem ein Text auf folgendes untersucht werden kann:

- Subjektivität-Objektivität: Besteht ein gegebener Text primär aus Fakten, oder haften ihm Emotionen an?

- Positiv-Negativ: Sind die Emotionen die im Text ausgedrückt werden primär positiv oder primär negativ?
- Stärke der Emotion: Wie stark ist die positive oder negative Emotion in einem Text?

Das Wörterbuch wird als Tabulator-getrenntes Textfile unter <http://sentiwordnet.isti.cnr.it/download.php> zu Verfügung gestellt. Das Wörterbuch enthält folgende Spalten:

- die Wortart ('a' = Adjektiv, 'n' = Nomen, ...)
- eine ID (z.B. 00004980)
- einen Wert der die Stärke der positiven Emotionen anzeigt (Zwischen 0 und 1, folgend auch *posScore* genannt)
- einen Wert der die Stärke der negativen Emotionen anzeigt (Zwischen 0 und 1, folgend auch *negScore* genannt)
- das Synset (Wörter mit der selben Bedeutung und jeweils einem «Rank»)
- eine Liste von Synonymen
- eine Beschreibung des Wortes

Den Wert für die Objektivität (*objScore*) lässt sich wie folgt berechnen:

$$objScore = 1 - (negScore + posScore) \quad (1)$$

Zum besseren Verständnis sind im Listing 2 drei Zeilen aus dem aktuellen SentiWordNet Wörterbuch abgebildet.

```
1 a_00004980__0_0_unabridged#1__(used of texts) not shortened; "an
   unabridged novel"
2 a_00005107__0.5_0_uncut#7 full-length#2_complete; "the full-length play"
3 a_00007813__0_0.5_nonabsorptive#1 nonabsorbent#1_not capable of absorbing
   or soaking up (liquids)
```

Listing 2: SentiWordNet Zeile)

Die Gute Dokumentation und die freie Verfügbarkeit aller Ressourcen lassen eine Python Implementation des SentiWordNet Wörterbuches zu.

### 2.1.5 SenticNet

SenticNet ist eine weitere lexikalische Resource welche im Opinion Mining verwendet werden kann. Das SenticNet kann unter <http://sentic.net/downloads/> heruntergeladen werden. Das Wörterbuch ist in diesem Falle als XML abgespeichert. Ein Wort-Eintrag enthält neben dem Wort selbst vor allem vier Werte für pleasantness, attention, sensitivity und aptitude. Aus diesen Werten lässt sich mit folgender Formel die allgemeine Polarität (positiv oder negativ) berechnen:[11]

$$p = \sum_{i=1}^N \frac{Plsnt(c_i) + |Attnt(c_i)| - |Snst(c_i)| + Aptit(c_i)}{3N} \quad (2)$$

In Version 2 des SenticNet wird dieser Wert aus praktischen Gründen jeweils pro Wort direkt im Wörterbuch mitgeliefert. Um die Polarität eines ganzen Textes zu berechnen wird jedoch weiterhin die oben genannte Formel empfohlen. [11] Zum einfacheren Verständnis befindet sich unter Listing 3 Beispielhaft ein Eintrag eines Wortes.

```

1 <rdf:Description rdf:about="http://sentic.net/api/en/concept/worship">
2   <rdf:type rdf:resource="http://sentic.net/api/concept"/>
3   <text xmlns="http://sentic.net/api/">worship</text>
4   <semantics xmlns="http://sentic.net/api/" rdf:resource="http://sentic.
5     net/api/en/concept/hope"/>
6   <semantics xmlns="http://sentic.net/api/" rdf:resource="http://sentic.
7     net/api/en/concept/religious_purpose"/>
8   <semantics xmlns="http://sentic.net/api/" rdf:resource="http://sentic.
9     net/api/en/concept/trust"/>
10  <semantics xmlns="http://sentic.net/api/" rdf:resource="http://sentic.
11    net/api/en/concept/devotion"/>
12  <semantics xmlns="http://sentic.net/api/" rdf:resource="http://sentic.
13    net/api/en/concept/religious"/>
14  <pleasantness xmlns="http://sentic.net/api/" rdf:datatype="http://www.w3
    .org/2001/XMLSchema#float">+0.265</pleasantness>
15  <attention xmlns="http://sentic.net/api/" rdf:datatype="http://www.w3.
    org/2001/XMLSchema#float">+0.601</attention>
16  <sensitivity xmlns="http://sentic.net/api/" rdf:datatype="http://www.w3.
    org/2001/XMLSchema#float">-0.207</sensitivity>
17  <aptitude xmlns="http://sentic.net/api/" rdf:datatype="http://www.w3.org
    /2001/XMLSchema#float">+0.373</aptitude>
18  <polarity xmlns="http://sentic.net/api/" rdf:datatype="http://www.w3.org
    /2001/XMLSchema#float">+0.344</polarity>
19 </rdf:Description>

```

Listing 3: SenticNet Wort

Alle Grundlagen und das Wörterbuch des SenticNet sind frei zugänglich, weshalb sich auch hier eine entsprechende Python Implementation anbot.



### 2.1.6 SASA - SailAil Sentiment Analyzer

Der SailAil Sentiment Analyzer ist eine machine-learning basierte Library die direkt für Python verfügbar ist. In dieser Arbeit wurde die Library in der Version 0.1.3 verwendet (<https://pypi.python.org/pypi/sasa/0.1.3>)

### 2.1.7 Klassifizierung mit NLTK und naivem Bayes

Bei der Recherche zu dieser Arbeit bin ich über einen Artikel von Jacob Perkins [16] gestolpert. Er beschreibt wie man mithilfe des Natural Language Toolkits (NLTK) [12] einen einfachen Sentiment Analyzer implementieren kann. Er trainiert einen `nlk.classify.NaiveBayesClassifier` mithilfe des NLTK eigenen Film-Review Korpus (`nlk.corpus.movie_reviews`).

## 2.2 Twitter Search API

Das Twitter Search API[2] ist ein REST API welches es erlaubt Tweets zu bestimmten Themen zu suchen. Christian Koepp hat eine Python Anbindung an dieses API implementiert[14]. Mithilfe dieser Anbindung lässt sich in wenigen Zeilen (Siehe Listing 4) eine Twitter Anbindung realisieren.

```
1 from TwitterSearch import *
2 try:
3     tso = TwitterSearchOrder()
4     tso.setKeywords(['Worldcup', 'Brazil'])
5     tso.setLanguage('en')
6     tso.setIncludeEntities(False)
7
8     ts = TwitterSearch(
9         consumer_key = 'aaabbb',
10        consumer_secret = 'cccdde',
11        access_token = '111222',
12        access_token_secret = '333444'
13    )
14
15    for tweet in ts.searchTweetsIterable(tso):
16        print( '@%s tweeted: %s' % ( tweet['user']['screen_name'], tweet['text'] ) )
17 except TwitterSearchException as e:
18    print(e)
```

Listing 4: TwitterSearch Python-Twitter Anbindung

Was das Twitter Search API nicht bietet, ist die Möglichkeit nach Tweets in bestimmten Zeiträumen zu suchen. Man sucht jeweils im aktuellen SearchIndex welcher sowohl aus populären als auch sehr aktuellen Tweets besteht. Für die in der Aufgabenstellung beschriebene Zeitraum-Auswahl-Funktion könnten Daten von einem kommerziellen Anbieter bezogen werden oder man könnte sich selber ein Twitter Archiv mithilfe der Streaming API [8] anlegen. Im Rahmen dieser Arbeit wurde darauf verzichtet. Das heisst, in der im Rahmen dieser Arbeit erstellten Applikation kann «nur» nach aktuellen oder populären Tweets mithilfe von Stichwörtern gesucht werden.

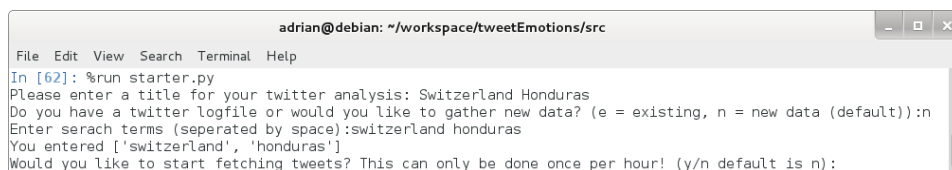
## 3 Umsetzung

### 3.1 Benutzerführung

Die Applikation wurde so als Konsolenapplikation angelegt, dass alle Module auch einzeln verwendet werden können. Für Enduser wurde jedoch im `starter.py` ein Konsolengesteuerter Workflow implementiert.

1. Titel für die Suche muss eingegeben werden
2. Der Benutzer wird gefragt, ob er bereits vorhandene Twitterdaten analysieren will oder ob er neue Twitterdaten über das API herunterladen möchte.
  - a) Wenn der Benutzer neue Twitterdaten laden möchte wird er nach Schlüsselwörtern gefragt
  - b) Dem Benutzer werden die eingegebenen Schlüsselwörter nochmals angezeigt und er muss bestätigen damit das API angestossen wird
  - c) Die Daten werden heruntergeladen und im File `keyword1_keyword2_...keywordn_raw_search_time.txt` abgelegt. Der Filepfad wird zurückgegeben.
3. Wenn der Benutzer vorhandene Twitterdaten analysieren möchte, muss er den Pfad zum Twitterdatenfile angeben
4. Der Benutzer bestätigt, dass er die Twitter Datenanalyse für die gegebenen Tweets starten möchte
5. Das Programm analysiert die Daten, erstellt die Plots und ein Logfile mit Resultaten

Live sieht dies wie in Abbildung 1 aus.



```
adrian@debian: ~/workspace/tweetEmotions/src
File Edit View Search Terminal Help
In [62]: %run starter.py
Please enter a title for your twitter analysis: Switzerland Honduras
Do you have a twitter logfile or would you like to gather new data? (e = existing, n = new data (default)):n
Enter search terms (seperated by space):switzerland honduras
You entered ['switzerland', 'honduras']
Would you like to start fetching tweets? This can only be done once per hour! (y/n default is n):
```

Abbildung 1: Benutzerführung

### 3.1.1 Verwendete bzw. Eingebundene Ressourcen

starter.py	Enthält den Code der Benutzerführung
gather_twitterdata.py	Enthält sowohl die Benutzerführung als auch die Methoden zum herunterladen der Tweets
analyse_twitterdata.py	Wird mit MPI angestossen um die Daten zu analysieren

Tabelle 2: Verwendete Ressourcen: Benutzerführung

## 3.2 Twitterdaten Sammeln

Wie bereits im Kapitel 2.2 beschrieben wurde zum Herunterladen der Tweets das Package TwitterSearch von Christian Koeppe[14] verwendet.

Im File gather\_twitterdata.py werden - geführt - Twitterdaten heruntergeladen. Das heisst, der Benutzer wird nach Schlüsselwörtern gefragt, darauf aufmerksam gemacht, dass nach einem Twitter API Call das Interface für eine weile blockiert sein wird, er muss bestätigen, dass er den Interface Call wirklich mit den von ihm eingegebenen Parametern anstossen will und zum Schluss wird ihm der Dateiname des Twitter Logs angezeigt.

Das schreiben der Tweets läuft wie folgt ab:

1. Tweet und Benutzer auslesen
2. Zeilenumbrüche im Tweet durch Leerschläge ersetzen
3. String im Format user \t tweet \n zusammensetzen
4. String ins File schreiben

Wie im Listing 5 ersichtlich, führt dieser Schreibvorgang zu Files bei welchen auf jeder Zeile ein Benutzername und ein Tweet steht. Innerhalb jeder Zeile sind Benutzernamen und Tweets mit einem Tabulator voneinander getrennt.

```
1 UrbanRadio254_Do you think Sepp Blatter is too old to lead FIFA? #  
  UrbanKickOff  
2 AlanMcneel_I am thinking of running as head of FIFA. I'm going tae be the  
  new Sepp Blatter. 'Sepp Bladder.'  
3 blunt_waves_Sepp #Blatter is as corrupt as most of the rulers we have in #  
  Africa. #FIFA.
```

Listing 5: Twitter Logfile Format)

### 3.2.1 Verwendete bzw. Eingebundene Ressourcen

gather_twitterdata.py	Enthält sowohl die Benutzerführung als auch die Methoden zum Herunterladen der Tweets
TwitterSearch	Library zum Ansteuern der Twitter API

Tabelle 3: Verwendete Ressourcen: Twitterdaten Sammeln

## 3.3 Algorithmen zur Erkennung der Gefühlslage in Texten

### 3.3.1 Emoticons

Als Grundlage für den Emoticon Algorithmus wurden als erstes, wie im Listing 6 ersichtlich, Emoticons von den verschiedenen Quellen [1][9][4][6] zusammengetragen und in die drei Gruppen positiv, negativ und neutral aufgeteilt.

```

1 positive = [u'\u263B', u'\u263A', u':)', u':D', u':-D', u':|', u':}', u':o',
  u':o|', u':o}', u':-|', u':-)', u':-}', u':=)', u':=|', u':=}', u':^|',
  u':^)', u':^}', u':B', u':-D', u':-B', u':^D', u':^B', u':=B', u':^B',
  u':^D', u':\')', u':\']', u':\'}', u':<3', u'^.^', u'^-^', u'^_^', u'^^^',
  u':*', u':*', u':-*', u':;)', u':;|', u':;}', u':-p', u':-P', u':-b', u':
  :^p', u':^P', u':^b', u':=P', u':=p', u':/o/', u':P', u':p', u':b', u':b',
  u':=p', u':=P', u':^b', u':\o/']
2 negative = [u'\u2639', u'D:', u'D=', u'D-', u'D^:', u'D^=', u':(', u':[',
  u':{', u':o(', u':o|', u':^(', u':^[', u':^{', u':=(', u':=^', u':>=(',
  u':>=|', u':>={', u':>=(', u':>=-{', u':>=-|', u':>=-(', u':>=^|', u':>=-(', u':
  :-|', u':-(', u':=(', u':=[', u':={', u':=^[', u':>:-=(', u':>=|', u':>=^(', u'
  '=\\', u':\\', u':/', u':$', u'o.O', u'O_o', u'Oo', u':$:-{', u':>:-{',
  u':>=^', u':o{']
3 neutral = [u':|', u':=|', u':-|', u':>.<', u':<.>', u':>_<', u':o', u':0', u'=O',
  u':@', u'=@', u':^o', u':^@', u':-.-', u':-_-', u':x', u'=X', u':-x',
  u':-@', u':-#', u':^x']

```

Listing 6: Emoticon Arrays

Um einen Tweet zu analysieren, wird nun mithilfe der Python Methode `string.count(substring)` gezählt wie viele positive, negative und neutrale Emoticons im Tweet vorkommen. Der Rückgabewert bestimmt sich dann wie folgt:

- positive - Wenn mehr positive als negative Emoticons vorkommen
- negative - Wenn mehr negative als positive Emoticons vorkommen
- neutral - Wenn gleich viele positive und negative Emoticons vorkommen oder nur neutrale Emoticons vorhanden sind

- unsure - Wenn keine Emoticons gefunden wurden

Umgesetzt wurde das Ganze im File `emoticon_algorithm.py`.

### 3.3.2 SASA

Die Anwendung des SailAil Sentiment Analyzer war denkbar einfach:

```
1 from sasa.classifier import Classifier
2 c = Classifier()
3 classification = c.classifyFromText(tweet) #classification[0] = unicode "
      neutral", "negative", "positive", "unsure", classification[1] = score
```

Listing 7: SASA Classifier

### 3.3.3 Text Preprocessing

Für die selber implementierten lexikalischen Algorithmen und den eigenen naiven Bayes Ansatz wurden Methoden zur Textvorbereitung umgesetzt. Die Methode `preprocessTweet` ersetzt alle Satzzeichen durch Leerschläge, ersetzt mehrfache Leerschläge durch einfache, ersetzt Grossbuchstaben mit den entsprechenden Kleinen und erstellt schliesslich eine Python List mit den einzelnen Wörtern.

Die Methode `getWordCombinations` wurde bei den lexikalischen Algorithmen verwendet um möglichst viele verwendete Ausdrücke («Wort-Kombinationen») wie zum Beispiel «monday morning» zu finden. Die Methode erstellt alle Wort-Kombinationen unter Beachtung der Reihenfolge der Wörter in der Eingabe. Zur Veranschaulichung wird im Listing ?? die Funktionsweise der Preprocessing Methoden an einem Beispiel gezeigt.

```
1 In [11]: words = preprocessing.preprocessTweet("Everyone loves monday
      mornings!")
2 In [12]: print words
3 ['everyone', 'loves', 'monday', 'mornings']
4
5 In [13]: wordcombos = preprocessing.getWordCombinations(words, " ")
6
7 In [14]: print wordcombos
8 ['everyone', 'everyone loves', 'everyone loves monday', 'everyone loves
   monday mornings', 'loves', 'loves monday', 'loves monday mornings', '
   monday', 'monday mornings', 'mornings']
```

Listing 8: Text Preprocessing

### 3.3.4 SentiWordNet

Der SentiWordNet Algorithmus wurde als Python Klasse implementiert. Im Konstruktor wird das Wörterbuch aus dem File eingelesen und in ein Python Dictionary verwandelt. Die Vorgehensweise ist dabei an die Java-Beispielimplementation von Petter Törnberg [17] angelehnt. Grob funktioniert es wie folgt:

1. Das Wörterbuch (Textfile) wird eingelesen und nach Zeilen aufgeteilt die folgenden Aktionen werden für jede Zeile ausgeführt
2. Wenn ein # am Anfang der Zeile steht wird sie ignoriert
3. Die Zeile wird nach Tabulatoren aufgesplittet
4. Wenn die Zeile nicht aus 6 Elementen besteht, wird eine Fehlermeldung ausgegeben
5. Der Synset-Score der Zeile wird berechnet (*positiveScore* – *negativeScore*)
6. Das Synset wird aufgesplittet und jedes Wort des Synsets wird zusammen mit dem Rank in ein temporäres Dictionary geschrieben (`tmpDict[wort][rank] = synSetScore`)
7. Der SentiNetScore für jedes Wort entspricht dem gewichteter Durchschnitt aller Vorkommnisse dieses Wortes wird wie folgt berechnet:  $\frac{\frac{1}{2} \cdot Rank_1 + \frac{1}{3} \cdot Rank_2 + \dots + \frac{1}{n} \cdot Rank_n}{1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}}$
8. Im finalen Dictionary werden jeweils das Wort und der entsprechende SentiNetScore abgelegt

Mit dem so erstellten Dictionary kann man einfach den SentiNetScore für Wörter und Ausdrücke auslesen. (`sentiNetScore = dict[ausdruck]`). Der Methode `getTweetScore` kann ein Tweet übergeben werden. Dieser wird zuerst vorverarbeitet (Sprich es werden alle Wort-Kombinationen generiert. Siehe Kapitel 3.3.3). Nun wird für jeden Ausdruck in der generierten Liste der SentiWordNet Score gesucht und schlussendlich wird der Durchschnitt der gefunden Scores wie folgt berechnet und zurückgegeben.

$$avg = \sum_{i=1}^n \frac{sentiWordNetScore(expression_i)}{n} \quad (3)$$

Implementiert ist das Beschriebene im Python File `sentiwordnet.py`.

### 3.3.5 SenticNet

Auch der SenticNet Algorithmus wurde als Python Klasse implementiert. Im Konstruktor wird in diesem Falle das XML File geparst. Das im SenticNet alle Wörter nur einmal vorkommen, werden diese direkt in ein Dictionary eingelesen. Der SenticNet Score besteht jeweils aus 5 Werten und ist in einer kleinen Klasse SenticScore abgebildet. Diese Klasse kann unter anderem geplottet werden (`SenticScore.plot()`) und die SenticNet Polarity gemäss Formel 4 berechnen.

$$p = \sum_{i=1}^N \frac{Plsnt(c_i) + |Attnt(c_i)| - |Snst(c_i)| + Aptit(c_i)}{3N} \quad (4)$$

Das Verarbeiten der Tweets läuft dann äquivalent zur Verarbeitung von Tweets mit dem SentiWordNet Algorithmus ab. Anstelle des Floats welcher SentiWordNet Score entspricht, werden hier einfach SentiScore Objekte aufsummiert und der Durchschnitt daraus wird berechnet.

Die Implementation dieses Algorithmus befindet sich im File `senticnet.py`

### 3.3.6 NLTK naive Bayes

Der letzte Implementierte Ansatz unter Verwendung der NLTK eigenen naive Bayes Implementation ist wie bereits im Kapitel 2.1.7 Beschrieben stark an den Artikel von Jacob Perkins [16] angelehnt. Konkret wurde wiederum eine Klasse implementiert, in deren Konstruktor alles nötige initialisiert wird. In diesem Falle wird im Konstruktor der naive Bayes Algorithmus mithilfe des NLTK Movie Review Korpus trainiert.

Das sowohl das Trainieren als auch das Klassifizieren sind wie in Listing 9 denkbar unspektakulär:

```
1 from nltk.classify import NaiveBayesClassifier
2 from nltk.corpus import movie_reviews
3
4 class NLTKTweetClassifier:
5     def word_feats(self, words):
6         return dict([(word, True) for word in words])
7
8     def classifyTweet(self, tweet):
9         words = preprocessTweet(tweet)
10        return self.classifier.classify(self.word_feats(words))
11
12    def __init__(self):
13        trainfeats = getTrainFeats() #get training set
```



```
14 | self.classifier = NaiveBayesClassifier.train(trainfeats) #training
```

Listing 9: Naive Bayes Training &amp; Klassifizieren

Implementiert ist dieser Algorithmus im File `nltk_tweet_classifier.py`.

### 3.4 Parallelisierung

Um die Performanz der Berechnungen zu erhöhen, wurde das Analysieren der Tweets mithilfe von MPI parallelisiert. Der Ablauf des parallelisierten Algorithmus ist dabei wie folgt:

1. RANK 0: das Config `config/config.txt` mit den Eigabeparametern `filename` (Des Twitter Logfiles) und `title` (Der Analyse) wird eingelesen
2. RANK 0: das Twitterlogfile `filename` wird eingelesen und es wird eine Liste von Tweets erzeugt
3. RANK 0: Die Algorithmen SenticNet und SentiWordNet werden initialisiert
4. ALLE: Die Algorithmen SASA und NLTK werden initialisiert (Die Objekte können nicht über das MPI COMM transferiert werden)
5. ALLE: Die Tweet-List, die SenticNet Instanz und die SentiWordNetInstance werden per Broadcast vom RANK 0 an alle übertragen
6. ALLE: Jedem Prozess werden  $\text{len}(\text{Tweet-List})/\text{AnzahlProzesse}$  Tweets zugeteilt
7. ALLE: Jeder Prozess analysiert die ihm zugeteilten Tweets mithilfe der `analyseTweet()` Methode und schreibt das Ergebnis in eine `resultList`
8. ALLE: Die `resultList` wird an den Prozess 0 übermittelt
9. RANK 0: Die Resultat-Listen der einzelnen Prozesse werden zusammengesetzt, zusammengefasst und geplottet

Während dem ganzen Prozess wird die Zeit, welche für die einzelnen Schritte benötigt wird berechnet und sowohl ausgegeben als auch in einem Logfile abgelegt.

### 3.4.1 Verwendete bzw. Eingebundene Ressourcen

<code>analyse_twitterdata.py</code>	Enthält den mit MPI parallelisierten Ablauf
<code>analyse_tweet.py</code>	Enthält sowohl den Code zum analysieren eines Tweets als auch den Code zum zusammenfassen und Plotten der Resultate
<code>autolabel.py</code>	Ein Codesnippet zum Beschriften der Matplotlib Balken
<code>emoticon_algorithm.py</code>	Der verwendete Emoticon Algorithmus
<code>logger.py</code>	Ein sehr einfacher Logger der verwendet wird um die Textausgabe in ein File zu schreiben
<code>nltk_tweet_classifier.py</code>	Der verwendete NLTK Classifier
<code>sasa_tweet_classifier.py</code>	Der verwendete SASA Algorithmus
<code>senticnet.py</code>	Der verwendete SenticNet Algorithmus
<code>sentiwordnet.py</code>	Der verwendete SentiWordNet Algorithmus

Tabelle 4: Verwendete Ressourcen: Parallelisierung

## 4 Analyse

### 4.1 Vergleich der Algorithmen

Bei den Experimenten mit den verschiedenen Algorithmen ist aufgefallen, dass der Emoticon Algorithmus aufgrund der seltenen Verwendung von Emoticons in Tweets praktisch nicht funktioniert. Je ernster das Thema desto weniger Emoticons werden tendenziell verwendet. Folgend die Resultate zweier Twitter Downloads. Die Keywörter für den ersten Download waren «switzerland »und «ecuador». Die Schweiz hat kurz zuvor am WM Auftaktspiel in der 93. Minute das entscheidende 2:1 gegen Ecuador geschossen. Die Keywörter für den zweiten Download waren «ISIS»und «Iraq». ISIS ist eine militante Jihadistengruppe, die derzeit für unruhen im Nordiran sorgt. [3] Folgend werden diese Resultate beschrieben und es werden Thesen zu den Resultaten der Twitteranalysen aufgestellt. Um diese zu verifizieren oder zu falsifizieren, bräuchte man gelabelte Twitterdaten, welche im Rahmen dieser Arbeit nicht erarbeitet werden konnten.

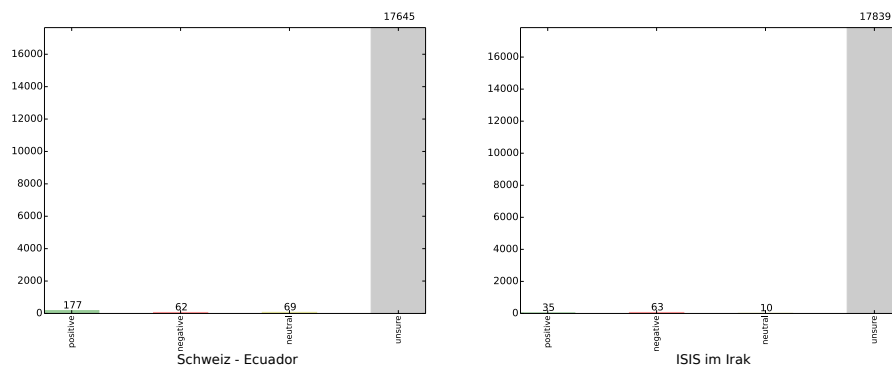


Abbildung 2: Emoticon Analyse: Schweiz - Ecuador vs. ISIS im Irak

Die Resultate der in Abbildung 2 sichtbaren Suche decken sich mit den Erkenntnissen aus vorhergehenden Suchläufen. In rund 1.7% der Schweiz-Ecuador Tweets wurden Emoticons verwendet. Erfreulicherweise waren diese mehrheitlich positiv. Die Irak Tweets hatten lediglich einen Emoticon-Anteil von 0.6% und diese waren meist negativ.

Beim der Twitteranalyse mithilfe des naiven Bayes der NLTK Library (Abbildung 3) ist ersichtlich, wie sich die Wahl des Trainingsets auf das Ergebnis auswirkt. Movie Reviews (das hier verwendete Trainingset) sind üblicherweise längere Texte die, wenn positiv, sehr sachlich geschrieben sind. Beim manuellen durchsehen der Tweets fiel auf, dass viele Tweets zu ISIS im Irak Zeitungsüberschriften entsprechen oder einfach sehr sachlich geschrieben sind. Dies wird von diesem Klassifizierungsverfahren als positiv gewertet.

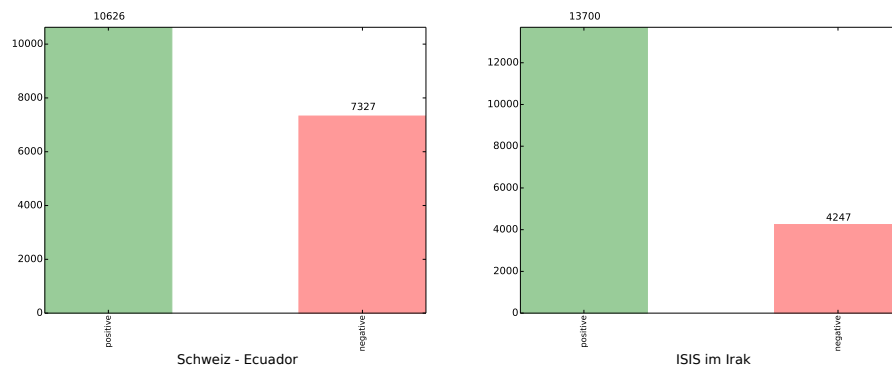


Abbildung 3: NLTK Naive Bayes: Schweiz - Ecuador vs. ISIS im Irak

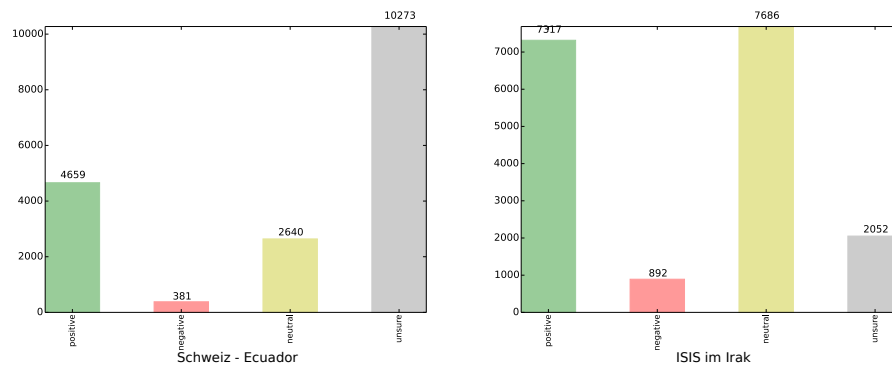


Abbildung 4: SASA: Schweiz - Ecuador vs. ISIS im Irak

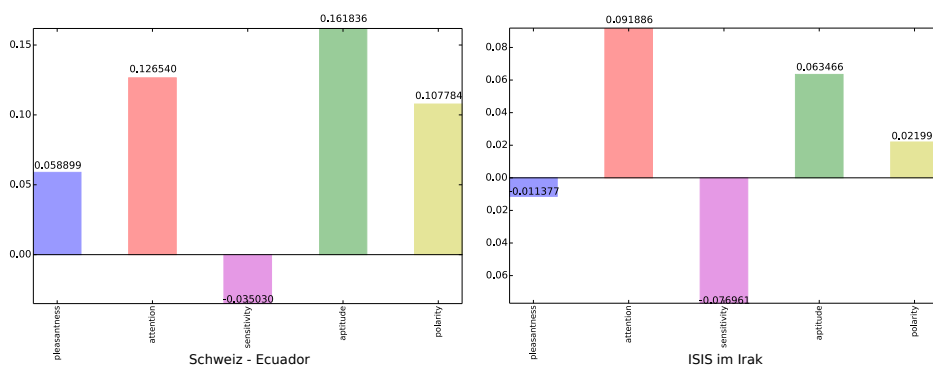


Abbildung 5: SASA: Schweiz - Ecuador vs. ISIS im Irak

Die SASA Library (vgl. Abbildung 4) hat viele der Irak-Tweets als neutral gelabelt. Dies würde sich mit meiner Beobachtung der vielen Zeitungsüberschriften decken. Weiter scheint der Algorithmus Probleme mit den sehr kurzen Tweets im Stil von «[#Switzerland](#)

beats #ecuador 2-1 at World #EnnerValencia #WorldCups»zu haben. Er Kategorisiert diese gar nicht erst.

Der SenticNet (vgl. Abbildung 5) Algorithmus zeigt für die Schweiz-Ecuador Tweets deutlich positivere Resultate als für die Tweets zum Treiben der ISIS Jihadisten im Irak. Diese Resultate scheinen plausibel zu sein.

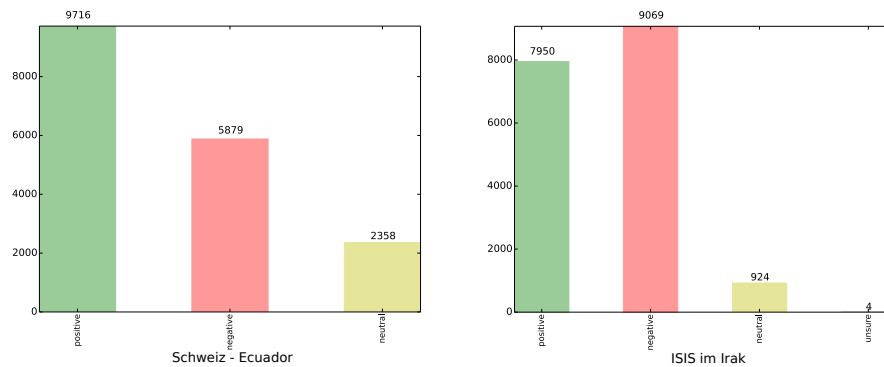


Abbildung 6: SentiWordNet: Schweiz - Ecuador vs. ISIS im Irak

Auch die Resultate des SentiWordNet (Abbildung 6) scheinen plausibel zu sein. Hier werden negative Emotionen aus den ISIS Tweets herausgelesen und das Schweizer Tor in der 93. Minuten wird zwar von der Mehrheit, jedoch nicht von allen Twitterern positiv beschrieben.

Diese Beobachtungen decken sich mit den Erkenntnissen von Pollyanna Gonçalves et al. [13]. Um wirklich etwas darüber auszusagen müsste gefilterte, bereinigte und manuell klassifizierte Tweets zu unterschiedlichen Themen untersuchen.

## 4.2 Parallelisierung

Die Vorteile der Parallelisierung auf meinem 4-Kerne System waren bereits ab wenigen tausend Tweets spürbar. Um zu sehen wie viel die Parallelisierung wirklich aus macht, wurden die Tweets über die ISIS Jihadisten mehrfach mit unterschiedlicher Anzahl Prozesse vom Programm analysiert. Dabei sind mir vor allem Unterschiede in den Zeiten für die Analysen an sich (der parallelisierte Teil) und der Initialisierung ins Auge gefallen.

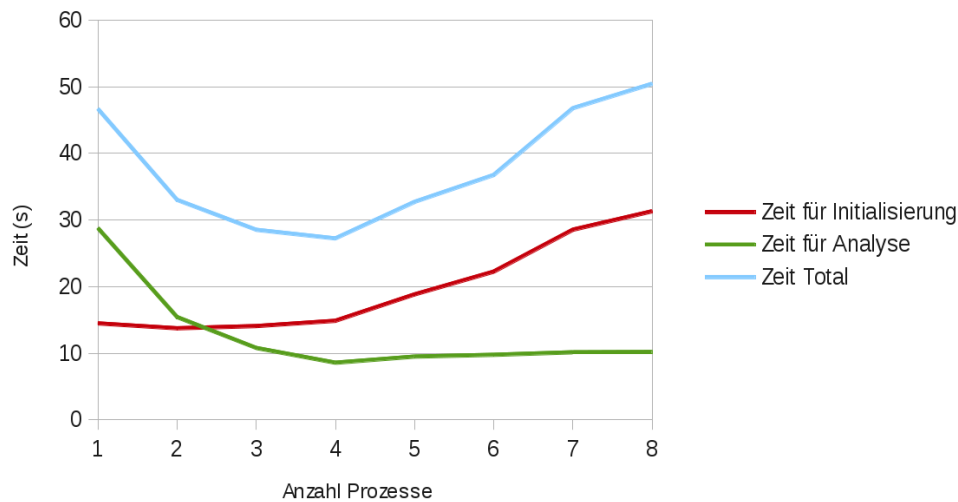


Abbildung 7: Laufzeitverhalten bei unterschiedlicher Anzahl Prozesse

In der Abbildung 7 ist ersichtlich, wie die Zeit für die Initialisierung bis zum Erreichen der physisch vorhandenen 4-Kerne praktisch konstant bleibt. Wenn man die Anzahl Prozesse weiter erhöht, nimmt die Dauer für die Initialisierung stark zu. Dies liegt daran, dass sowohl der SASA Algorithmus als auch der NLTK Classifier für jeden Prozess initialisiert werden müssen. Diese Initialisierungen sind sehr aufwändig und wenn die physisch vorhandenen 4-Kerne diese Aufgabe für mehr als 4 Prozesse erledigen sollen, dauert dies entsprechend länger.

Die Zeit für die Analyse ist bei einem Prozess am höchsten, und nimmt bis zum Erreichen der 4 vorhandenen Cores um fast  $\frac{2}{3}$  von 30s auf unter 10s ab. Wenn mehr Prozesse verwendet werden, wird diese Zeit nicht merklich erhöht, aber sie nimmt auch nicht mehr ab.

Daraus lässt sich schlussfolgern, dass in dem gegebenen Setup 4 Prozesse die beste Performance bieten. Weiter könnte man daraus ableiten, dass die optimale Anzahl Prozesse der Anzahl verfügbarer Cores entspricht. Diese These könnte unter gleichen Testbedingungen auf verschiedenen Setups weiter untersucht und gegebenenfalls verifiziert oder falsifiziert werden. Die Einstellung zur Anzahl Prozesse kann im `starter.py` angepasst werden.

## 5 Fazit

Trotz anfänglichen Schwierigkeiten (Twitter API bietet keine Zeitraumsuche, keine Antwort vom SentiStrength Team) ist es gelungen einige Algorithmen zur Erkennung von Emotionen in Texten einzubinden. Es konnten sowohl Unterschiede zwischen den Resultaten der verschiedenen Algorithmen als auch die Performance Vorteile der Parallelisierung mit MPI aufgezeigt werden. Die Bewertung der Unterschiede der verschiedenen Algorithmen oder die Bewertung der Eignung der Algorithmen zum Erkennen von Emotionen in Tweets, würden einen umfangreichen Datenkorpus mit gelabelten Tweets zu verschiedensten Themengebieten erfordern, war nicht Teil dieser Arbeit und könnte in weiteren Studien untersucht werden.

## Abbildungsverzeichnis

1	Benutzerführung . . . . .	10
2	Emoticon Analyse: Schweiz - Ecuador vs. ISIS im Irak . . . . .	18
3	NLTK Naive Bayes: Schweiz - Ecuador vs. ISIS im Irak . . . . .	19
4	SASA: Schweiz - Ecuador vs. ISIS im Irak . . . . .	19
5	SenticNet: Schweiz - Ecuador vs. ISIS im Irak . . . . .	19
6	SentiWordNet: Schweiz - Ecuador vs. ISIS im Irak . . . . .	20
7	Laufzeitverhalten bei unterschiedlicher Anzahl Prozesse . . . . .	21

## Tabellenverzeichnis

1	Projektplanung . . . . .	3
2	Verwendete Ressourcen: Benutzerführung . . . . .	11
3	Verwendete Ressourcen: Twitterdaten Sammeln . . . . .	12
4	Verwendete Ressourcen: Parallelisierung . . . . .	17

## Listings

1	Python Anbindung an SentiStrength (JAVA) . . . . .	5
2	SentiWordNet Zeile) . . . . .	6
3	SenticNet Wort . . . . .	7
4	TwitterSearch Python-Twitter Anbindung . . . . .	8
5	Twitter Logfile Format) . . . . .	11
6	Emoticon Arrays . . . . .	12
7	SASA Classifier . . . . .	13
8	Text Preprocessing . . . . .	13
9	Naive Bayes Training & Klassifizieren . . . . .	15

## Literatur

- [1] Emoticons - show your friends how you really feel. <https://messenger.yahoo.com/features/emoticons>.



- [2] Get search/tweets. <https://dev.twitter.com/docs/api/1.1/get/search/tweets>.
- [3] Islamic state in iraq and the levant. [http://en.wikipedia.org/wiki/Islamic\\_State\\_in\\_Iraq\\_and\\_the\\_Levant](http://en.wikipedia.org/wiki/Islamic_State_in_Iraq_and_the_Levant).
- [4] List of text emoticons – the ultimate resource. <http://cool-smileys.com/text-emoticons>.
- [5] The liwc2007 application. <http://www.liwc.net/howliwcworks.php>.
- [6] Miscellaneous symbols. [http://en.wikipedia.org/wiki/Miscellaneous\\_Symbols](http://en.wikipedia.org/wiki/Miscellaneous_Symbols).
- [7] Sentistrength - java version. <http://sentistrength.wlv.ac.uk/#Java>.
- [8] The streaming apis. <https://dev.twitter.com/docs/api/streaming>.
- [9] What is the full list of emoticons? <https://support.skype.com/en/faq/FA12330/what-is-the-full-list-of-emoticons>.
- [10] Wordnet - a lexical database for english. <http://wordnet.princeton.edu/>.
- [11] Erik Cambria, Catherine Havasi, and Amir Hussain. Senticnet 2: A semantic and affective resource for opinion mining and sentiment analysis. 2012.
- [12] Dan Garrette, Peter Ljunglöf, Joel Nothman, Mikhail Korobov, Morten Minde Neergaard, and Steven Bird. Natural language toolkit. <http://www.nltk.org/>.
- [13] Pollyanna Goncalves, Fabricio Benevenuto, Matheus Araujo, and Meeyoung Cha. Comparing and combining sentiment analysis methods. *Proceedings of the first ACM conference on Online social networks*.
- [14] Christian Koepp. Twittersearch. <https://github.com/ckoepp/TwitterSearch>.
- [15] B Pang, L Lee, and S Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *ACL Conference on Empirical Methods in Natural Language Processing*.
- [16] Jacob Perkins. Text classification for sentiment analysis - naive bayes classifier. <http://streamhacker.com/2010/05/10/text-classification-sentiment-analysis-naive-bayes-classifier/>.
- [17] Petter Törnberg. Sentiwordnetdemocode. <http://sentiwordnet.isti.cnr.it/code/SentiWordNetDemoCode.java>.