NTNU

Norwegian University of
Science and Technology

Characterizing Twitter Data using Sentiment Analysis and Topic Modeling

Asbjørn Ottesen Steinskog

Jonas Foyn Therkelsen

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Björn Gambäck, IDI

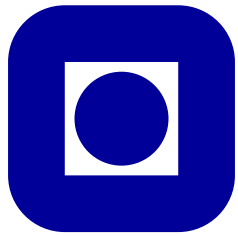
Norwegian University of Science and Technology
Department of Computer and Information Science

Asbjørn Ottesen Steinskog, Jonas Foyn Therkelsen

Characterizing Twitter Data using Sentiment Analysis and Topic Modeling

Master's Thesis, Spring 2016

Artificial Intelligence Group
Department of Computer and Information Science
Faculty of Information Technology, Mathematics and Electrical Engineering



Abstract

As the global community becomes increasingly connected, it gets more and more common to express thoughts and opinions through social networking websites. Twitter, currently the largest microblog website in the world, is heavily used for this purpose. Well known politicians, comedians and trending persons use this medium to express their minds through 140-character messages. This makes Twitter one of the platforms being most influential on the global web communities' way of thinking.

This thesis combines topic modeling and sentiment analysis in order to obtain information from tweets. While sentiment analysis seeks to find out what *opinions* people have, topic modeling tries to find out what they *talk about*.

Convential topic modeling schemes, such as Latent Dirichlet Allocation, are known to perform inadequately when applied to tweets, due to the sparsity of short documents. To alleviate these disadvantages, we apply several pooling techniques, aggregating similar tweets into individual documents. We specifically study the aggregation of tweets sharing authors or hashtags.

Our Twitter Sentiment Analysis system is comprised of seven different machine learning classifiers. These aim to predict whether a message's polarity is of neutral, negative or positive sentiment. Four machine learning algorithms, Maximum Entropy, Naïve Bayes, Support Vector Machines and Stochastic Gradient Descent, have been proposed for performing sentiment classification in this thesis. The classifiers were trained through experiments of extensive grid searches on a parameter space and preprocessing methods in order to achieve optimal classification scores.

To combine topic modeling with sentiment analysis, a visualization application called *TweetMoods* was built. TweetMoods simultaneously examines the topics contained in a Twitter corpus retrieved by a search query, and the sentiments expressed in these tweets.

Our topic modeling results show that aggregating similar tweets into individual documents increases the topic coherence significantly. On performing message polarity classification on tweets, the Maximum Entropy classifier yielded results outperforming most earlier submitted work to the International Workshop on Semantic Evaluation of 2015. This proves the importance of our extensive grid searches on optimizing the parameter space of the classifiers.

Sammendrag

Ettersom det globale samfunnet blir stadig mer sammenkoblet, blir det mer populært å uttrykke tanker og meninger gjennom sosiale nettverk. Twitter, verdens største nettside for mikroblogging, brukes mye til dette formålet. Kjente politikere, komikere og andre trendsettere bruker mediet til å uttrykke seg gjennom 140-tegns innlegg. Dette gjør Twitter til en av plattformene med størst innflytelse på de globale nettsamfunnenes tankegang.

Denne masteroppgaven kombinerer emnemodellering og sentimentanalyse for å hente ut informasjon fra tweets (Twitter-innlegg). Mens sentimentanalyse har som mål å finne ut hvilke meninger folk har, søker emnemodellering å finne ut *hva* folk snakker om.

Konvensjonelle emnemodelleringsmetoder slik som Latent Dirichlet Allocation, er kjent for å ha problemer med tweets på grunn av den korte lengden til innleggene. For å lette disse ulempene, anvender vi flere samlingsteknikker for å aggregere lignende innlegg til individuelle dokumenter. Her ser vi spesielt på aggregeringen av tweets fra samme forfatter eller som deler samme emneknagg.

Sentimentanalysesystemet består av syv forskjellige maskinlæringsklassifikatorer. Disse har som mål å klassifisere om et innlegg uttrykker et negativt, positivt eller nøytralt sentiment. Vi foreslår i denne masteroppgaven fire forskjellige maskinlæringsalgoritmer for utførelse av sentimentklassifisering: Maximum Entropy, Naïve Bayes, Support Vector Machines og Stochastic Gradient Descent. Hver klassifikator er trent opp gjennom omfattende rutenettssøk på dets parameterområde og preprosesseringsmetoder, for å oppnå optimale klassifiseringsmål.

En visualiseringsapplikasjon, kalt *TweetMoods*, ble utviklet for å kombinere emnemodellering med sentimentanalyse. Denne applikasjonen undersøker både emne og sentiment i et Twitter-datasett basert på en spørring.

Våre emnemodelleringsresultater viser at aggregering av lignende Twitterinnlegg til individuelle dokumenter øker sammenhengen mellom emner betydelig. Ved polaritetklassifisering av tweets viser vi at Maximum Entropy-klassifikatoren gir best resultater i våre eksperimenter. I sammenligning med innsendte resultater ved "International Workshop on Semantic Evaluation"i 2015, oppnådde våre systemer bedre resultater enn de fleste konkurrentene som deltok. Dette beviser viktigheten av vårt omfattende rutenettssøk ved optimering av klassifikatorenes parameterområde.

Preface

This Master's Thesis has been carried out at the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The report discusses a system using sentiment analysis and topic modeling for retrieving information from tweets, as well as relevant theory on the topics. The thesis concludes our Master of Computer Science degrees at the Department of Computer and Information Science (IDI) at NTNU and was supervised by professor Björn Gambäck and co-supervised by Lars Bungum.

Asbjørn Ottesen Steinskog, Jonas Foyn Therkelsen
Trondheim, 26.06.2016

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation

Due to the tremendous amount of data broadcasted on microblog sites like Twitter, extracting information from microblogs has turned out to be useful for establishing the public opinion on different issues. O'Connor et al. [2010] found a correlation between word frequencies in Twitter and public opinion surveys in politics. Analyzing tweets over a timespan can give great insights into what happened during that time, as people tend to tweet about what's concerning them and their surroundings. The combination of sentiment analysis and topic modeling of tweets can therefore help reveal what is going on in the world and what people feel about it. The annual SemEval workshop[1] and its tasks in the fields of sentiment analysis and topic modeling of Twitter inspired the main subject of the paper. Many influential people post messages on Twitter, and investigating the relation between the underlying topics of different authors' messages could yield interesting results about people's interests. One could for example compare the topics different politicians tend to talk about to obtain a greater understanding of their similarities and differences. Twitter has an abundance of messages, and the enormous amount of tweets posted every second makes Twitter suitable for such tasks.

1.2. Twitter

Twitter is the world's largest social network for micro-blogging, averaging 310 million monthly active users. Moreover, websites with embedded tweets get one billion unique visits every month.[2] Twitter was originally based on the idea that an individual should be able to communicate with a small group of people using an SMS service. The company was launched in 2006, but had its breakthrough during the South by Southwest Interactive conference in 2007, where Twitter usage increased from 20,000 tweets to 60,000 tweets per day. Quickly recognized globally due to its rapid growth, Twitter took its place among the largest social networks in the world. From 400,000

[1] https://en.wikipedia.org/wiki/SemEval
[2] https://about.twitter.com/company

tweets per quarter in 2007, to 100 million per quarter in 2008, and 50 million tweets per day in 2010. Currently, Twitter has an average network traffic of 500 million tweets per day, and is the 9th most popular website on the internet.

Among its users, Twitter is actively used for expressing opinions towards different topics. Due to the limitation of 140 characters per tweet, Twitter users are generally known for expressing their thoughts through informal language, using abbreviations and gramatically incorrect language. The community has also spawned user-generated metatags, like hashtags and mentions. These relatively new expressive terms have analytical value when it comes to opinion mining. The informal language also proposes challenges within natural language processing and information retrieval, as tweets are usually more incoherent than traditional documents. Despite this, the extensive amounts of tweets posted daily makes Twitter a great resource for opinion mining.

1.3. Project goals

We investigate what possibilities are available when analyzing topics and opinions of tweets over a given timespan. We hypothesize that one can achieve knowledge of current events by examining the content of microblog data, by finding what topics are trending and what sentiment is generally conveyed in a corpus of tweets. These are the subgoals carried out during the project:

G1: Establish the state-of-the-art
Present an extensive survey of the current state-of-the-art methods for topic modeling and sentiment analysis of microblogs. There has been a lot of related research during the latest years, much thanks to the API and extensive data on Twitter. The research is split in two parts, where one focus will be on topic modeling and how to effectively retrieve the topics from a corpus of microblogs. The second part is sentiment analysis and retrieving the sentiment from small texts known as microblogs. By investigating previous research done on sentiment analysis and topic modeling, we can establish the current state-of-the-art.

G2: Create a topic modeling system for microblogs
We aim to create a system for discovering trending topics and events in a corpus of tweets, as well as exploring the topics of different Twitter users and how they relate to each other. Utilizing Twitter metadata might mitigate the disadvantages tweets typically have

when using standard topic modeling methods; user information as well as hashtag co-occurrences can give a lot of insight into what topics are currently trending.

G3: Create a sentiment analysis system for microblogs
We aim to create a state-of-the-art sentiment analysis system optimized for informal microblog posts from Twitter. We wish to attain the highest accuracies and F1-scores compared to a baseline system by running a series of experiments on the SemEval data sets.

G4: Create a visualization system
We aim to create a state-of-the-art visualization application of sentiments and topics of tweets. The system will incorporate both the topic modeling and sentiment analysis systems developed during the project.

1.4. Contributions

C1 A literature review on the subject of sentiment analysis and topic modeling.

C2 The implementation of a Twitter topic modeling system.

C4 The implementation of a Twitter sentiment classification system.

C5 The implementation of a visualization application that incorporates the Twitter topic modeling and sentiment analysis systems.

C6 A study on the topic modeling and sentiment analysis system in comparison to similar proposed systems.

1.5. Overview of the thesis

Chapter 2 is an introduction to the some of the background theory relevant to the issue of sentiment analysis and topic modeling in Twitter.
Chapter 3 introduces the tools that helped making the project realizable.
Chapter 4 is comprised of a literature study on topic modeling and sentiment analysis in microblogs, and presents the current state-of-the-art uncovered by our survey.
Chapter 6 introduces topic modeling and the task of estimating the number of topics to use for a topic model. Further, it discusses the use of hashtags in Twitter, and how they can be utilized to find the currently popular topics in the social network using a co-occurrence network to connect the most

frequent and trending hashtags.
Chapter 7 introduces the setup and architecture of the Twitter Sentiment Analysis classifiers.
Chapter 5 discusses the two applications built during the project as well as their architectures. An application for creating a dataset of reliable tweets for topic modeling, and a combined topic modeling and sentiment analysis web application.
Chapter 8 and 9 presents the experiments conducted and results found with topic modeling and sentiment analysis, and the combination, on Twitter posts.
Finally, in Chapter 10 we discuss the accomplishments of the project goals put forward in the introduction, and what remains seen as future work.

2. Background Theory

The following chapter on background theory presents an introduction to the most relevant theory to sentiment analysis and topic modeling in Twitter. Moreover, we discuss some components of Natural Language Processing, Machine Learning, and Topic modeling. This theory lays ground for the architecture of the system developed.

2.1. Natural Language Processing

Natural language processing (NLP), extensively explained by Manning and Schütze [1999] is concerned with the interactions between computers and human (natural) languages. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation.

2.1.1. Basic text processing

In sentiment analysis and topic modeling systems, the corpus of tweets is most often represented as a collection of sets, called bag(s)-of-words. Each vector's indices represent words in the corpus. The corpora can grow immensely large in size, often comprised of several hundred thousand tweets. As such, the amount of words encountered grows with the size of the corpus, and therefore also the length of the vectors. Therefore, a set of basic text processing techniques are required to simplify the complex data.

Yang and Pedersen [1997] presented the importance of textual feature selection. Among which stop word removal and word stemming play essential parts.

Stop word removal

Stop words refer to the most common words in a language. It is common to remove these words as they are the least discriminative to a document. They are removed to relieve the system of excessive computation.

Stemming

Stemming reduces a word to its word stem. A quick example would be to reduce all words in the set *[run, running, runner, runners]* to the same stem *run* or similar. Affix stemmers refer to stripping similar words of the prefixes and suffixes to the same root.

Bag-of-Words

In their work, Zhang et al. [2010] give an extensive introduction to the Bag-of-words model. The Bag-of-words model is a way to represent a textual document by placing all its words into a bag (set of elements), ignoring the grammar and word order. A simple example is representing a list of the words ["hello", "good-bye", "greeting", "hello"] as the vector [2, 1, 1]. Each index represent new words encountered, and its value represents the number of encounters throughout the document.

Term Frequency-Inverse Document Frequency (TF-IDF)

While the term weighting scheme IDF was originally defined by Sparck Jones [1988], Ramos [2003] explains how TF-IDF can be used to determine word relevance in document queries. TF-IDF is a statistic comprised of a document's term frequency and inverse document frequency. It is a common weight in information retrieval, measuring how important a word is to a document in a corpus. The term frequency is the amount of times a term occurs in a document. The inverse document frequency gives a measure of how important the term is.

$$TFIDF = TF * IDF = tf_{t,d} * log(N/df_d)$$

N-gram

An n-gram is a sequence of n continuous items of either words or characters in a text. Sequences of one by one item are called unigrams, two by two items are called bigrams, sequences of three items are called trigrams, and larger sequences are called n-grams (where n is the length of the sequence).

2.1.2. Part-of-speech tagging (POS-tagging)

Schmid [1994] explains POS-tagging as the process of marking a word in a text as corresponding to a particular part-of-speech category. This includes identifying words as nouns, verbs, adjectives, etc. and how they relate to different scopes of the sentence. POS-tagging has many applications within natural language processing, like solving word ambiguity . Traditionally, off-the-shelf POS-tagging systems have targeted corpora of news articles. As with Named Entity Recognition (NER) systems, Twitter has proved a

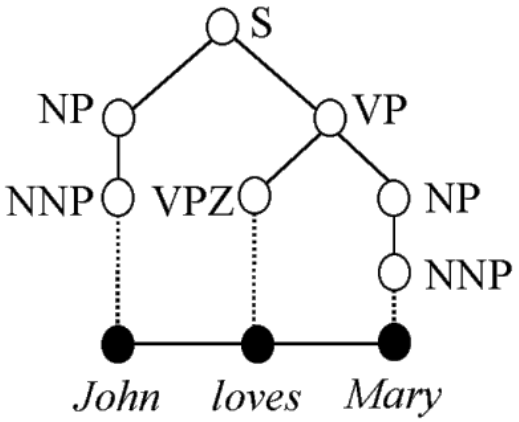


Figure 2.1.: An example of a part-of-speech treebank

difficult domain for POS-tagging. Gimpel et al. [2011] propose a tagger that effectively categorizes words in tweets.

2.1.3. Named Entity Recognition (NER)

A recent survey by Nadeau and Sekine [2007] explains how NER seeks to find and classify elements in text into categories such as the names of persons, organizations, locations, etc. Today, state-of-the-art systems that perform NER methods achieve near-human performance in identifying entities in texts in given domains. NER can be used in combination with topic modeling algorithms to give named entities a higher weight. This would make organizations, locations and persons more likely to be classified as a topic. This would also affect the entities and words that do not fall under these categories. Twitter has been proven to be somewhat difficult, due to the informal language used by its users. A proposed open-source system to solve the Twitter-specific problems can be seen in Ritter et al. [2011].

2.2. Machine Learning for Sentiment Analysis

This section includes a brief overview of classification scoring and evaluation methods for sentiment analysis. Based on the literature study from Chapter 4 we settled on Support Vector Machine (SVM), Naive Bayes, Maximum entropy (MaxEnt) and the Stochastic gradient descent (SGD) machine learning methods.

2.2.1. Classification Scoring

There are many ways to perform classification scoring and evaluation. The most common metrics of classification scoring includes precision, recall, accuracy and the F-score.

False / True positives and negatives

True positives (T_p) are the samples correctly identified as belonging to a specific category. True negatives (T_n) are the samples that are correctly identified as not belonging to a specific category. False positives (F_p) and false negatives (F_n) are the samples that are incorrectly identified as belonging to a category.

Precision

Precision is the measure of result relevancy. It is defined as the number of true positives over the sum of true positives and false positives.

$$precision = \frac{T_p}{T_p + F_p} \tag{2.1}$$

Recall

Recall is the measure of how many truly relevant results are returned. It is defined as the number of true positives, over the number of true positives and false negatives.

$$recall = \frac{T_p}{T_p + F_n} \tag{2.2}$$

Accuracy

Accuracy is the number of correctly predicted cases out of all existing cases.

$$accuracy = \frac{T_p + T_n}{T_p + T_n + F_p + F_n} \tag{2.3}$$

F1-score

The $F_1 score$ is the harmonic mean of precision and recall, and is one of the most common metrics for evaluating machine learning classifiers.

$$F_1 score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.4}$$

2.2.2. Naïve Bayes

Naïve Bayes is a particularly popular method for text categorization in supervised machine learning. Because of its simplicity, Naïve Bayes usually requires appropriate preprocessing to compete with more advanced methods, like Support Vector Machines.

Naïve Bayes builds on the concepts of conditional probability and Bayes' rule. Simply stated, conditional probability is finding the probability of something that will happen, given that something else already has happened. This is the base for Bayes' rule in statistics. Bayes Rule is in Naïve Bayes used for going from knowing the probability of A given B, P(A|B), to finding the probability of B given A, P(B|A). In reality, it's more complicated because we have to find out the probability of A given multiple evidence — instead of just one. To simplify the complication, we uncouple all the evidence and treat each piece of evidence individually, which is why we call the method naïve.

2.2.3. Support Vector Machines

Support Vector Machines (SVM) is a popular classification method, first introduced by Cortes and Vapnik [1995]. It is among the most common classification techniques for supervised learning of textual data in tweets. The algorithm is a binary classification technique, assigning input data to belong to one of two classes. Input data are vectors containing features. In conjunction, the features can be put into a high-dimensional feature space. The SVM looks at the data and splits the feature space into optimal class segments. This is done by constructing a hyperplane that maximizes the margin between the two classes. The margin is an essential part of Support Vector Machines, and is the largest distance to the nearest training-data point between the classes. The larger the margin, the lower the generalization error of the classifier.

In our case, input data consists of data labeled within one of three classes. Since SVM is a binary classification technique, we reduce the approach into several binary classification problems (either one-versus-one or one-versus-many).

Here are some advantages and disadvantages of SVM, based on information from the open-source library SciKit Learn[1].

Advantages:

[1] http://scikit-learn.org

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

Disadvantages:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

2.2.4. Maximum Entropy (MaxEnt)

Maximum Entropy is a multinomial logistic regression model suited for classifiying problems with more than two classes. It is a probabilistic classification model with no assumptions of conditional independence. It is therefore suited for feature selection methods where there is no conditional independence among the features, like textual domains. The model is called Maximum Entropy (or MaxEnt for short) because it seeks the model with the maximum information entropy and best represents the dataset.

2.2.5. Stochastic Gradient Descent (SGD)

SGD is an optimization method for unconstrained optimization problems. Both gradient descent and stochastic gradient descent updates a set of parameters in an iterative manner to minimize an error function. Stochastic gradient descent only uses one training sample from a training set to update a parameter in a particular iteration, rather than running through all samples before updating. This way the linear classifier converges quickly and is well suited for large datasets. Using convex loss functions like hinge or loss, the classifier can act much like an SVM or MaxEnt classifier, respectively.

2.3. Topic modeling

This section introduces some important topic modeling methods, as well as describing the task of evaluating the coherence of a topic model. A more thorough description of topic modeling and its applications is given in Chapter 6.

2.3.1. Latent Dirichlet Allocation

LDA (Latent Dirichlet Allocation) is an unsupervised generative probabilistic model of a set of documents, introduced by Blei et al. [2003]. It has become one of the most commonly used methods for document modeling, proving efficacy in its ability to accentuate the latent topics contained in a collection of documents. LDA is a generative topic model which generates mixtures of latent topics from a collection of documents, where each mixture of topics produces words from the collection's vocabulary with certain probabilities. These documents might consist of text corpora or other types of discrete data. The generative process of running LDA consists of several steps. A distribution over topics is first sampled from a Dirichlet distribution, and a topic is further chosen based on this distribution. Moreover, each document is modeled as a distribution over topics, and a topic is represented as a distribution over words. The following steps outlines the generative process of running LDA on a set of documents, as seen in Blei [2012]:

1. Randomly choose a topic distribution t.
2. For each word in document d:
    a) Randomly choose a topic from the topic distribution in t.
    b) Randomly choose a word from the corresponding distribution over the vocabulary.

LDA consequently represents the topics as a list of words and their probabilities of being generated by that particular topic.

Online Latent Dirichlet Allocation

Hoffman et al. [2010] implements Online Latent Dirichlet Allocation (OLDA), which is an extension to LDA that allows the model to be updated with new documents incrementally. This could be useful when analyzing documents arriving in a stream, allowing the topic model to adapt to changes in incoming documents. The visualization application presented in Section

5.1 incorporates online topic modeling as a way for users to update a topic model with tweets retrieved from a search.

2.3.2. Dynamic Topic Model

Dynamic Topic Model (DTM), proposed by Blei and Lafferty [2006], takes the temporal ordering of the documents into consideration. It is a sequential extension to LDA. Blei and Lafferty [2006] originally used dynamic topic modeling to show how trends in word usage in topics evolve over time, finding that a topic model trained with past documents better fit incoming documents for a new timeslot than standard LDA. Furthermore, Wang et al. [2012] developed a continuous dynamic topic model to predict the timestamps of documents.

2.3.3. Author-topic model

Several methods for retrieving information about documents and their respective authors have been proposed by researchers. The Author-topic model [Rosen-Zvi et al., 2004] is an extension to LDA which takes information about an author into account. The process of running the Author-topic model on a corpus resembles LDA: for each word in a document d, an author from the document's set of authors is chosen at random. A topic t is then chosen from a distribution over topics that are specific to that author, and the word is generated from that topic. Note that the Author-topic model manages to handle multiple authors per document, although tweets only have one author. The model gives information about the diversity of the topics covered by an author, and makes it possible to calculate the distance between the topics covered by different authors, to see how similar they are in their themes and topics. Hong and Davison [2010] proposes a simpler extension to LDA for modeling the authors of tweets. By aggregating tweets written by an author into one individual document, they mitigate the disadvantages caused by the sparse nature of tweets. They found that this method of aggregating messages from the same author can improve classification and topic model tasks in short text environments. We perform experiments employing both the Author-topic model proposed by Rosen-Zvi et al. [2004] and a variant of the Author-topic model proposed by Hong and Davison [2010]. See Section 8.1.4 for the experiments conducted using these methods.

2.3.4. Topic model scoring

There isn't a single method for evaluating the accuracy of a topic model, and several evaluation metrics have been proposed. The unsupervised nature of

topic discovery makes the assessment of topic models challenging. Quantitative metrics don't necessarily provide an accurate reflection of a human's perception of a topic model, as it's difficult to mathematically reproduce human judgement. Moreover, human judgement is not clearly defined, as different people might disagree on the efficiency of a topic model. The topic models produced in the experiments conducted in this thesis are therefore evaluated by a variety of metrics.

UMass coherence metric

The UMass coherence metric, introduced by Mimno et al. [2011], measures the *topic coherence* in a topic model. The topic coherence C for a topic is defined as:

$$C = \sum_{m=2}^{M} \sum_{l=1}^{m-1} log \frac{D(w_m, w_l) + 1}{D(w_l)} \tag{2.5}$$

$(w_1, ..., w_M)$ being the M most probable words in the topic, $D(w)$ being the number of documents that contain word w, and $D(w_m, w_l)$ being the number of documents that contain both words w_m and w_l. This metric utilizes word co-occurrence statistics gathered from the corpus, which ideally already should be accounted for in the topic model. Mimno et al. [2011] concludes that standard topic models do not fully utilize the co-occurrence information, and they achieved reasonably good results when comparing the scores obtained by this measure with human scoring on a corpus containing 300,000 journal paper abstracts from the NIH (National Institutes of Health). However, statistical methods cannot model a human's perception of the coherence in a topic model perfectly.

Human judgement

Due to the difficulties of calculating the efficacy of a topic model mathematically, researchers also employ human judgement to evaluate topic models. Mimno et al. [2011], in their attempt to produce a method that automatically evaluates topic models, simply let two *experts* evaluate the topics as either *good*, *intermediate* or *bad*. Chang et al. [2009] propose two tasks where humans can evaluate topic models: *word intrusion* and *topic intrusion*. We also propose a new method for evaluating an author-topic model using human judgement.

Word intrusion The *word intrusion* task lets humans measure the coherence of the topics in a topic model by evaluating the latent space in

the topics. The human subject is presented with six words, and the task is to find the *intruder*, which is the one word that does not belong with the others. The idea is that the subject should easily identify the *intruding* word when the set of words minus the intruder makes sense together. For a set of words lacking coherence, it is difficult to find the *intruder*, and it will typically be chosen at random. The following method is proposed for constructing a set of words to present to the subject:

1. Select a random topic from the model.
2. Select the five most probable words from the topic.
3. Select an intruder at random from a pool of words with low probability in the current topic.

One example mentioned in their paper is that it is easy to identify the out of place word *apple* in the set *{dog, cat, horse, apple, pig, cow}*, since the other words are all animals, but it is difficult to identify one out of place in a set of words lacking coherence, like the set *{car, teacher, platypus, agile, blue, Zaire}*.

Topic intrusion In the *topic intrusion* task, subjects are shown a document's title along with the first few words of the document. They are also presented four topics, three of those being the highest probability topics assigned to that document, and the remaining *intruder topic* being chosen randomly from the low-probability topics in the model. The task of the subject is to find the low-probability topic.

In our attempt to evaluate the different Twitter topic models generated in Chapter 8, we used methods based on human judgement as well as the UMass measure. In Section 6.3, we propose a method for evaluating an Author-topic model, specifically with the Twitter domain in mind.

3. Tools

This section includes a brief overview of the several tools and resources we used in this project. They include Twitter-specific tools, such as the Twitter API, as well as several tools and resources useful for sentiment analysis and topic modeling tasks.

3.1. Twitter API

The Twitter API was used for retrieving tweets for our datasets. There are two main components to the Twitter API, the *Twitter Streaming API*[1] and the *Twitter Search API*[2].

3.1.1. Twitter Streaming API

The Streaming API allows retrieving tweets in real-time, returning a small random sample of all public statuses. This was used for collecting large datasets of tweets to be used for both sentiment analysis and topic modeling. This API is favorable to the Search API if the goal is to retrieve a large amount of random tweets.

3.1.2. Twitter Search API

The Search API allows conducting searches, reading user data and posting tweets. The Search API should be used if you need *specific tweets*, such as tweets from certain users, or tweets containing specific hashtags or words. This API was mainly used for retrieving tweets for the Author-topic model and for the visualization application.

3.2. Scikit-learn (sklearn)

Scikit-learn by Pedregosa et al. [2011] is a free machine learning library for the Python programming language. The module contains state-of-the-art implementations of many machine learning algorithms, as well as tools for

[1] https://dev.twitter.com/streaming/overview
[2] https://dev.twitter.com/rest/public/search

customizing and optimizing these methods. It offers an extensive API and documentation of its library, and has been used in the sentiment classification system of the project.

3.2.1. Transformers

Scikit-learn provides a library of transformers that can be used to clean, reduce, expand or generate feature representations. In this project they are used to generate feature representations, e.g. transforming a tweet into a simple vector containing the amount of negations found in the tweet.

3.2.2. Feature Union

The Feature Union module takes a set of transformers, which it concatenates resulting in a combined list of transformer objects. It essentially serves the same purpose as the first step of an sklearn pipeline, therefore it is common to include a feature union in a pipeline.

3.2.3. Pipeline

The pipeline sequentially applies a list of transformers and a final estimator (e.g. a machine learning classifier). It is used as a convenient way of combining the several stages of feature extraction and setting a classifier's parameters. The pipeline allows for cross validation and grid searching on a set of parameters for each estimator, or just training the final estimator with a set of custom or default parameters.

3.2.4. Grid Search

A parameter space can be set for each estimator in the Pipeline framework, which allows us to perform an exhaustive grid search across the various combinations of parameters to find the setup with the best cross-validation score. The grid search is a very important step of feature engineering for optimizing a classifier, and was found to be one of the most effective ways to improve results of classification during this project.

3.3. Sentiment lexicons

Bing Liu sentiment lexicon presented in Liu [2010] is an english sentiment lexicon first started in 2004 on product reviews, but has been extended several times since. The version used for this project was published in 2012.

NRC Twitter sentiment lexicon presented in Kiritchenko et al. [2014] is an english sentiment lexicon that was integrated in some of the winning systems of the SemEval workshop of 2013 and 2014.

MPQA sentiment lexicon presented in Wilson et al. [2005] is an english sentiment lexicon from 16,000 subjective expressions from a system built in 2005.

3.4. Pandas

Pandas by McKinney [2014] is an open source high-performance data structure and data analysis tool for the Python programming language. It has been utilized in the project for creating appropriate data structures for sentiment classification on the different Twitter datasets imported.

3.5. Twitter NLP

Twitter NLP by Ritter et al. [2011] is a Python library providing useful resources for Twitter natural language processing. Many similar solutions are also available in the well-known Natural Language ToolKit (NLTK) library; however, Twitter NLP is made specifically for processing tweets, appropriate for our project. We used Twitter NLP's *Twokenize* by Owoputi et al. [2012] which is tailored for tokenizing tweets.

3.6. Gensim

Gensim by Řehůřek and Sojka [2010] is an open-source topic modeling toolkit for Python, providing implementations of LDA and other topic models. We used Gensim's LDA implementation for performing the topic modeling experiments, including generating topic models, calculating the UMass coherence scores and inferring topic distributions for new documents.

3.7. Tornado

Tornado[3] is a web application framework for Python. This was used for creating an API used for sharing data between the Python back-end and the visualization application.

[3] http://www.tornadoweb.org/en/stable/

3.8. MeteorJS

MeteorJS[4] is a web application framework built on top of Node.js[5]. We chose this framework for building two applications; the classification application and the visualization application. MeteorJS integrates with MongoDB by default, which makes it suitable for handling Twitter data; the Twitter API represents the tweets as JSON, and the database fields in MongoDB are JSON objects.

3.9. langid

Langid[6] is a language identification system for Python. We used it to identify the language of tweets that were classified as *undefined* by the automatic Twitter language identificator[7], helping discard non-english tweets.

[4] https://www.meteor.com/
[5] https://nodejs.org
[6] https://github.com/saffsd/langid.py
[7] https://blog.twitter.com/2015/evaluating-language-identification-performance

4. Related work

We will in this section establish the state-of-the-art regarding topic modeling and sentiment analysis, by performing a systematic literature review. The structure of this literature review is inspired by the work of Kofod-Petersen [2012] on the topic, as well as his course held at NTNU during the Fall semester of 2015.

4.1. Structured Literature Review Methodology

Step 1 & 2: Planning the review
For the purpose of this document we can assume that a need has already been identified (step 1) and that a review has been commissioned (step 2). This description will cover steps 3 and 4 in the planning phase.

Step 3: Specifying research questions
It is assumed that a specific problem (P) is tackled using some specific constraints, methods and/or approaches (C) to develop a system, application or algorithm (S).

1. *What are the existing solutions to the problem of finding sentiment of Twitter posts regarding specific topics (P)?*
2. *How does the different solutions found by addressing question 1 compare to each other with respect to the different methods for finding sentiment in tweets?*
3. *What is the strength of the evidence in support of the different solutions (C)?*
4. *What implications will these findings have when creating the system (S)?*

Step 4: Developing a review protocol

The search of related and relevant research papers to our literature study will be executed through the Google Scholar search engine, which searches

several scientific literature websites (such as citeseerx, ACM, major universities, etc.).

The grouping of keywords developed for the literature review can be seen in Table 4.1:

	Group 1	**Group 2**	**Group 3**
Term 1	Sentiment analysis	Topic (modeling)	Twitter
Term 2	Opinion mining	Named Entity Recognition	Microblog

Table 4.1.: Grouping of queries

The following queries were developed based on keyword grouping:

- Topic AND ("Sentiment Analysis" OR "Opinion Mining") AND (Twitter OR Microblog)
  Which produced 9,530 results.
- ("Named entity recognition" OR "Topic Modeling") AND ("Sentiment analysis" OR "Opinion mining")
  Which produced 3,830 results.

The following prioritized criteria are the basis for selection of papers:

1. *Title and Abstract*
2. *Conclusion and Future Work*
3. *Experiments and results*
   - Data and Algorithms
4. *Discussion*
   - Related work
5. *Background reading*
   - Introduction and Reference

As both social media platforms and sentiment analysis techniques are rapidly evolving, we reject papers submitted before 2011.

4.2. Review of literature

In this section we apply the protocol developed in Section 4.1, choose our literature and present the results of the review.

4.2.1. Identification of research

Google Scholar is effective at retrieving the most relevant articles to your queries, supplied with metadata such as citations, as well as grouping duplicate results. This was clearly evident, as the articles became decreasingly relevant to our literature review. Google Scholar makes for efficient searching and inspection of which articles to read and inspect furtherly, and is an appropriate tool for this task.

For each query we went thirty results into the search.

4.2.2. Selection of primary studies

The selection process of the articles is split into three different stages, the primary and secondary inclusion stage and the quality assessment.

The primary inclusion stage concerns inspecting the title, abstract and metadata of the article for sentiment analysis or topic modeling, and whether it has produced relevant results.

The secondary inclusion stage concerns whether the study focuses on microblogs, or short textual data — as well as whether it has proposed techniques for implementing the previous criteria.

This can be split into the following inclusion criteria:

Primary:

IC 1 The article concerns sentiment analysis, topic modeling or entity recognition.

IC 2 The article has produced relevant results.

Secondary:

IC 3 The article focuses on microblogs, or specifically Twitter.

IC 4 The article proposes techniques for implementing previous inclusion criteria.

4.2.3. Study quality assessment

The quality of the articles that pass the first two inclusion stages, are evaluated by a set of ten different quality assessment criteria. This allows us to score the article and rank all retrieved articles, and finally choose the set of articles with the highest scores. The articles score 1 point for including a criterion, $\frac{1}{2}$ point for partial inclusion, and 0 points for exclusion.

Quality assessment criteria:

QC 1 Is there a clear statement of the aim of the research?

QC 2 Is the study put into context of other studies and research?

QC 3 Are system or algorithmic design decisions justified?

QC 4 Is the test data set reproducible?

QC 5 Is the study algorithm reproducible?

QC 6 Is the experimental procedure thoroughly explained and reproducible?

QC 7 Is it clearly stated in the study which other algorithms the study's algorithm(s) have been compared with?

QC 8 Are the performance metrics used in the study explained and justified?

QC 9 Are the test results thoroughly analysed?

QC 10 Does the test evidence support the findings presented?

4.3. Results

The results are presented, in no particular order, in Figures 4.1, 4.2, and 4.3 found on the following pages. Figure 4.1 presents the scores of each paper reviewed. Figure 4.2 presents the results from the first query, while Figure 4.3 presents the results from the second query. The most relevant papers are discussed in Sections 4.4.2 and 4.4.3.

4.4. Other

This section describes the International Workshop on Semantic Evaluation (SemEval), as well as discussing the most relevant papers from the literature review.

4.4.1. International Workshop on Semantic Evaluation (SemEval)

The International Workshop on Semantic Evaluation[1] is an annual series of evaluations of computational semantic analysis systems. The workshop presents several tasks each year, which researchers around the world can take part in. The tasks are intended to explore the nature of meaning in

[1] https://en.wikipedia.org/wiki/SemEval

Quality assessment														
ID	**Query #**	**Author**	**Title**	**QC1**	**QC2**	**QC3**	**QC4**	**QC5**	**QC6**	**QC7**	**QC8**	**QC9**	**QC10**	**Score**
1	1&2	Wang, Wei, Liu, Zhou, Zhang (2011)	Topic Sentiment Analysis in Twitter: A Graph-based Hashtag Sentiment Classification Approach	1	1	1	1	1	1	0.5	1	1	0.5	9
2	1&2	Pak, Paroubek (2010)	Twitter as a Corpus for Sentiment Analysis and Opinion Mining	1	0.5	0.75	1	1	1	1	1	0.5	1	**8.75**
3	1&2	Kouloumpis, Wilson, Moore (2011)	Twitter Sentiment Analysis: The Good the Bad and the OMG!	0.5	1	0.5	1	0.5	0.5	0.5	0.5	0.5	0.5	**6**
4	1&2	Agarwal, Xie, Vovsha, Rambow, Passonneau (2011)	Sentiment Analysis of Twitter Data	1	1	0.5	0.5	0.5	1	1	1	1	1	**8.5**
5	1	Meng, Wei, Lio, Zhou, Li, Wang (2012)	Entity-Centric Topic-Oriented Opinion Summarization in Twitter	1	1	1	0.5	1	1	0.5	1	1	1	**9**
6	1	Jiang,Yu, Zhou, Liu, Zhao (2011)	Target-dependent Twitter Sentiment Classification	1	1	0.5	0.5	0.5	0.5	1	1	1	1	**8**
7	1	Si, Muhkerjee, Liu, Li, Li, Deng (2013)	Exploiting Topic based Twitter Sentiment for Stock Prediction	1	1	1	1	0.5	1	1	1	0.5	1	**9**
8	3	Olessia Koltsova and Sergei Koltcov (2013)	Mapping the Public Agenda with Topic Modeling: The Case of the Russian LiveJournal	1	1	1	0.5	1	1	1	0.5	1	1	**9**
9	3	Nguyen, Shirai (2015)	Topic Modeling based Sentiment Analysis on Social Media for Stock Market prediction	1	1	1	1	0.5	1	1	0.5	0.5	1	**8.5**
10	3	Wang, Wu, Li, Tang, Shao, Zhuang (2014)	Jointly Discovering Fine-grained and Coarse-grained Sentiments via Topic Modeling	1	0.5	1	1	0.5	1	1	0.5	0.5	1	**8**
11	3	Mitchell, Aguilar, WIlson, Durme (2013)	Open-Domain Targeted Sentiment	1	1	1	1	1	1	0.5	1	1	1	**9.5**
12	3	Wu, Chen, Ou, Wang, Yang, Lei (2014)	Topic-Based Sentiment Analysis Icorporating User Interactions	1	0.5	1	0	1	1	0.5	0.5	0,5	0.5	**6**
13	3	Waila, Singh, Singh (2013)	Blog Text Analysis using Topic Modeling, Named Entity Recognition and Sentiment Classifier Combine	1	0.5	0.5	0	0.5	0.5	0.5	0.5	1	1	**6**
14	3	Sotiropoulos, Kounavis, Kourouthanassis, Giaglis (2014)	What drives social sentiment? An entropic measure-based clustering approach towards identifying factors that influence social sentiment polarity	0.5	1	1	1	1	1	0	1	0.5	1	**8**

Figure 4.1.: Quality assessment scores.

Results						
ID	**Authors**	**Title**	**Publication**	**Method information**	**Dataset**	**Score (QA)**
1	Wang, Wei, Liu, Zhou, Zhang	Topic Sentiment Analysis in Twitter: A Graph-based Hashtag Sentiment Classification Approach	2011	Analyses sentiment classification of #Hashtags in twitter. Three tasks: (1) sentiment polarity of tweets containing the hashtag; (2) hashtags co-occurrence relationship and (3) the literal meaning of hashtags. Uses a two-stage SVM classifier to determine the sentiment polarity of a tweet.		9
2	Pak, Paroubek (2010)	Twitter as a Corpus for Sentiment Analysis and Opinion Mining	2010	Used TreeTagger for POS-tagging and observed the difference in distributions among positive, negative and neutral sets. Uses multinomial naive Bayes, tried SVM and CRF.	Dataset: http://www.stanford.edu/~alecmgo/cs224n/twitterdata.2009.05.25.c.zip	8.75
3	Kouloumpis, Wilson, Moore (2011)	Twitter Sentiment Analysis: The Good the Bad and the OMG!		Controversial conclusion that POS-tagging does not provide useful results in sentiment analysis of microblogs. Though, states that using hashtags to collect training data did prove useful, as did using data collected based on positive and negative emoticons.	(HASH) dataset, compiled from Edinburgh Twitter copus. Test set: (ISIEVE): http://twittersentiment.appspot.com	6
4	Agarwal, Xie, Vovsha, Rambow, Passonneau (2011)	Sentiment Analysis of Twitter Data	2011	General step-by-step implementation on sentiment analysis. (1) We introduce POS-specific prior polarity features. (2) We explore the use of a tree kernel to obviate the need for tedious feature engineering. Tree kernel and feature based models. Gain of 4% over the unigram baseline model	Uses 11.875 manually annotated tweets.	8.5
5	Meng, Wei, Lio, Zhou, Li, Wang (2012)	Entity-Centric Topic-Oriented Opinion Summarization in Twitter	2012	Opinion summarization for entities, such as celebrities and brands, in Twitter. Proposes an entity-centric topic-based opinion summarization framework. Uses hashtags as weak supervision in topic modeling.	Manually annotated tweets	9
6	Jiang,Yu, Zhou, Liu, Zhao (2011)	Target-dependent Twitter Sentiment Classification	2011	Incorporates target-dependent features; and takes related tweets into consideration. Incorporates syntactic features to distinguish texts used for expressing sentiments towards different targets in a tweet. Takes related tweets of the current tweet into consideration by utilizing graph-based optimization, which significantly improves performance.	2000 manually annotated tweets	8
7	Si, Muhkerjee, Liu, Li, Li, Deng (2013)	Exploiting Topic based Twitter Sentiment for Stock Prediction	2013	Proposes a technique to leverage topic based sentiments from Twitter to help predict the stock market. Utilizes a continuous Dirichlet Process Mixture model to learn the daily topic set.	624782 tweets from search results on Standard and Poor's 100 stocks, and their firms. From nov, 2012 to feb, 2013. No labeling.	9

Figure 4.2.: Part 1: Results of the quality assessment.

Results						
ID	**Authors**	**Title**	**Publication**	**Method information**	**Dataset**	**Score (QA)**
8	Olessia Koltsova and Sergei Koltcov (2013)	Mapping the Public Agenda with Topic Modeling: The Case of the Russian LiveJournal	2013	Modeling LiveJournal's (russian blogsite) topic structure and public opinion. Topic modeling using the LDA algorithm.	Hand labeled topics.	9
9	Nguyen, Shirai (2015)	Topic Modeling based Sentiment Analysis on Social Media for Stock Market prediction	2015	Builds a model that combines LDA and topic modeling in one technique. Method is called TSLDA, and is claimed better than both LDA and JST.	Data set was mined from LiveJournal with own constructed system	8.5
10	Wang, Wu, Li, Tang, Shao, Zhuang (2014)	Jointly Discovering Fine-grained and Coarse-grained Sentiments via Topic Modeling	2014	Proposes a system called LDA with multi-grained sentiments. The system will both find out what topic a text is talking about as well as find the author's sentiment to that topic. They use three datasets. Large Movie Review Dataset (aclImdb), Customer Review Datasets, Additional Review Datasets	http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html http://ai.stanford.edu/~amaas/data/sentiment	8
11	Mitchell, Aguilar, Wilson, Durme (2013)	Open-Domain Targeted Sentiment	2013	Predict sentiment in tweets for any named person or oganization, without giving a fixed set of topics. Use NER and Opinion Expression Extraction to jointly predict NE and sentiment towards them. Does not rely on Part-of-speech tagger which is often used. They use information from Sentiment Lexicons and hand-written features. AND Conditional Random Fields (CRF)	www.m-mitchell.com/code	9.5
12	Wu, Chen, Ou, Wang, Yang, Lei (2014)	Topic-Based Sentiment Analysis Icorporating User Interactions	2014	Proposes a novel probabilistic generative model (TSIUM) to extract topics and topic-specific sentiments from online comments Topic Sentiment Interaction Union Model (TSIUM)	Chinese datasets	6
13	Waila, Singh, Singh (2013)	Blog Text Analysis using Topic Modeling, Named Entity Recognition and Sentiment Classifier Combine	2013	Designed an integrated framework by utilizing Topic Modeling, Entity Extraction and Sentiment Analysis. Uses hierarchical Bayesian analysis of the text documents to find similar documents. Used LDA for topic modeling. Extracted entities and performed sentiment analysis on their blog posts. The entities were put up in a tag cloud, and each entitiy was colored red or green depending on their polarity.	Data is comprised of blog posts on social-political events related to the Arab spring.	6
14	Sotiropoulos, Kounavis, Kourouthanassis, Giaglis (2014)	What drives social sentiment? An entropic measure-based clustering approach towards identifying factors that influence social sentiment polarity	2014	Utilization of a semantically-aware clustering procedure that effectively combines topic modeling and sentiment analysis algorithms. 1. Data Collection & Data Preparation. 2. Corpus Separation. 3. VSM-based Corpus Vectorization. 4. Sentiment Classification. 5. Topic Modeling. 6. Sentiment per Topic. 7. Semantic Factors Identification.	Tested on a dataset of 135.000 tweets on two leading US mobile providers (ATT and Verizon). Time period from February 2nd to February 13th.	8

Figure 4.3.: Part 2: Results of the quality assessment.

	Team	Twitter 2015	Twitter sarcasm
1	TwitterHawk	50.51	31.30
2	KLUEless	45.48	39.26
3	Whu-Nlp	40.70	23.37
4	whu-iss	25.62	28.90
5	ECNU	25.38	16.20
6	WarwickDCS	22.79	13.57
7	UMDuluth-CS8761	18.99	29.91

Table 4.2.: Results of the SemEval 2015 workshop, task 10 subtask C

language, from a computational standpoint. The annual SemEval workshop of 2015 presented a task on Topic-Based Message Polarity Classification, a task which is similar to what is studied in this paper. The scores of each team can be found in Table 4.2, and were calculated by

$$(F1_{\text{pos}} + F1_{\text{neg}})/2 \tag{4.1}$$

The most successful teams of the SemEval workshop on the targeted sentiment task in 2015, used common methods in achieving their results. Boag et al. [2015] used a supervised learning approach using linear SVM, heavily focused on text pre-processing and feature engineering. Plotnikova et al. [2015] used a supervised approach on the SemEval datasets with the Maximum Entropy method, using text preprocessing, lexicon and emoticon scores and trigrams. They essentially ignored topics in their technique, which is interesting given the task — coming in second. Zhang et al. [2015] was different to the other techniques by focusing on word embedding features, as well as the traditional textual features and preprocessing. Though they argued that when the model was only extended with the word embedding features, it didn't necessarily significantly improve their results.

4.4.2. Topic Modeling

Based on the retrieved documents of the literature study, there was some variety to the proposed methods for modeling topics. The majority of the studied research used the Latent Dirichlet Allocation (LDA) algorithm and its variations.

LDA has proven efficacy in modeling the semantics in a document corpus. Koltsova and Koltcov [2013] present their findings on the use of LDA on mostly political topics regarding the presidential elections in Russia, but also on recreational and other topics. Applied to a dataset of all posts of 2,000 Russian bloggers on LiveJournal, they managed to correctly identify

between 30-40% of the topics in their corpus, despite the broad categories — giving evidence to the robustness to LDA. Similar research worth mentioning include the work by Sotiropoulos et al. [2014] on targeted sentiment towards topics related to the two major telecommunication firms, ATT and Verizon in the USA, as well as the work by Waila et al. [2013] on identifying socio-political events and entities during the Arab Spring and finding the global sentiment towards these events.

Topic modeling algorithms have gained increased attention in modeling tweets. Tweets do, however, pose some difficulties because of their sparseness, as the short documents might not contain sufficient data to establish satisfactory term co-occurrences. Several pooling techniques, which involves aggregating similar tweets into individual documents, have therefore been applied to mitigate the disadvantages of sparse data. Hong and Davison [2010] show that the effectiveness of topic models is highly influenced by the document's length, and that aggregating short messages produces better models. Moreover, Quan et al. [2015] present a solution for topic modeling for sparse documents, finding that automatic text aggregation during topic modeling is able to produce more interpretable topics from short texts than standard topic models.

4.4.3. Twitter Sentiment Analysis

The following section discusses research focused on the sentiment analysis part, rather than the topic modeling.

Part-of-speech tagging is an important feature of many sentiment analysis systems. Pak and Paroubek [2010] proposed that some POS-tags may be strong indicators of emotional text, contrary to Kouloumpis et al. [2011] who found that part-of-speech features may not be useful for sentiment analysis. Supporting the claim of POS-tags as important features, Agarwal et al. [2011] proposed a tree kernel and feature based method that beats their unigram base models and concludes in accordance with Pak and Paroubek [2010] that prior polarity of words and part-of-speech tags contribute greatly to identifying the sentiment of text.

Wang et al. [2011] propose a state-of-the-art Twitter Sentiment Analysis system, using a two-stage SVM classifier. The first of which determines whether a tweet is neutral or subjective, while the second determines the polarity of a subjective tweet. Their approach is incorporated with a hashtag graph model to boost results, which combines the information on hashtags co-occurrence and tweet sentiment. Meng et al. [2012] is another study with emphasis on hashtags, but only use them as weakly supervised information in their topic modeling algorithms. Jiang et al. [2011] presented a TSA model that incorporates syntactic features and related tweets.

Utilizing a graph-based optimization they take related tweets into consideration to finding the sentiment of the target, which significantly improves performance. Lastly, Si et al. [2013] proposed a technique to predict the stock market with sentiment analysis, using topic modeling to extract the related targets to the market. Their approach performs better than the existing state-of-the-art topic based methods.

5. Architecture

This chapter describes the main application for our system, a visualization application, called *TweetMoods*, that allows users to retrieve topic and sentiment information of tweets retrieved by a search. We also describe a polarity classification application that lets users label tweets manually.

5.1. TweetMoods

We created an application that combines the sentiment analysis components with the topic modeling components and gathers information about tweets searched for by the user. The idea is that the user should be provided with accurate information about the sentiments and the latent topics contained in a corpus of tweets. By searching for certain brands or products, the application should give an overview of the sentiment expressed in messages mentioning these particular brands or products, as well as providing information about which underlying topics are contained in these tweets. We first provide an overview of the API that serves as a layer between the front-end visualization application and the programs running the sentiment classification and topic modeling, before giving a general explanation of the architecture and the details of our application. Figure 5.1 shows the architecture of the visualization application.

5.1.1. API

An API was created for connecting the front-end of the visualization application with the Python code running the sentiment and topic modeling programs. The Python web framework Tornado (see 3.7) was used for this purpose, hence is it referred to as the *Tornado API* in Figure 5.1. The API displays several endpoints that trigger different parts of the underlying programs, which after completion returns the response as JavaScript Object Notation (JSON). Consider the following API request:

```
  sentiment/search?query=coca-cola&count=100
&result_type=recent&classifier=maxent
```

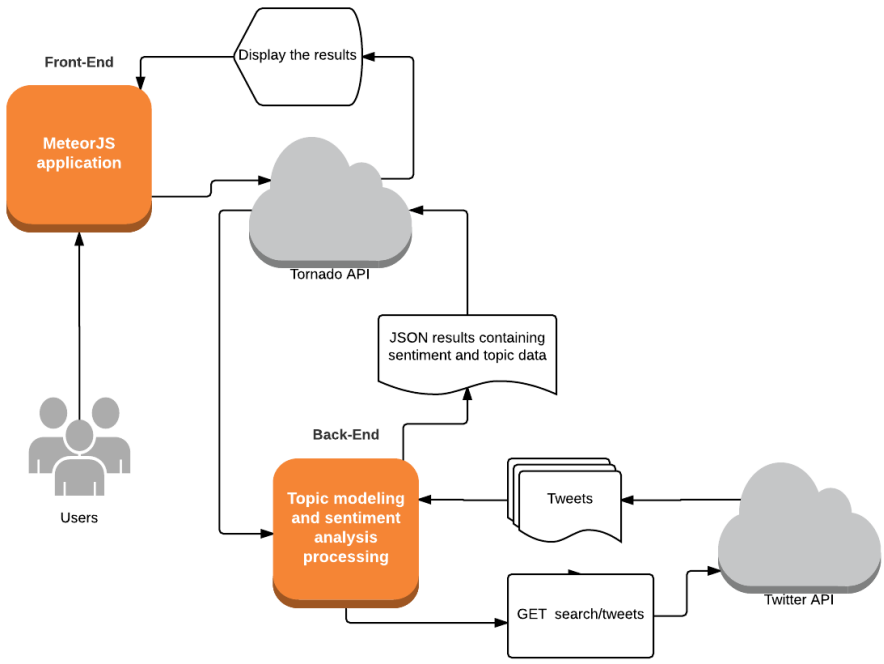


Figure 5.1.: The overall architecture of the visualization application. Users submits queries through the MeteorJS app, which sends requests to the Tornado API. The Python backend retrieves tweets through the Twitter API, runs the sentiment analysis and topic modeling algorithms on the tweets and returns the results.

This API requests accesses the endpoint `sentiment/search`, which gives a response with the sentiments of a group of searched tweets. Four parameters are given: *query* (the search query), *count* (the number of tweets to search), *result_type* (whether the tweets should be *popular*, *recent* or *mixed*) and *classifier* (which sentiment classifier to use for the classification).

The API request feeds instructions to the underlying Python backend, and the following tasks are performed:

1. Search for *100* tweets using the Twitter Search API, using the search query *coca-cola*, with the *result_type* parameter set to *recent*.

2. Classify each tweet as either *positive*, *neutral* or *negative*, using the specified *classifier*.

3. Display the response as a list of tweet JSON objects with their corresponding sentiment.

5.1.2. Application overview

TweetMoods was developed using the MeteorJS framework (3.8), and provides several features, the main feature being the possibility to predict the sentiment and infer topic mixture of a tweet corpus. The application contains three components, each serving different purposes related to analyzing the topics or sentiments of text.

Predict sentiment of text The *Predict sentiment of text* feature lets the user test the sentiment classification on an arbitrary string, and choose which sentiment classifier to use. Figure 5.2 shows how this feature is presented to the user. By clicking the *Predict sentiment* button, an API request is sent to the `sentiment` endpoint, providing the chosen *classifier* and *text* as queries. In this case, *i didn't like the new avengers movie* is chosen as the text, and *MaxEnt* is chosen as the classifier. As seen in Figure 5.2, the returned value for these parameters is *negative*.

Get topic distribution for text This feature provides a way of inferring the topic distribution of an arbitrary string over a pre-trained topic model. The user provides a string and chooses a topic model from a list of pre-trained topic models. An API request is then sent to the `topic/mixture` endpoint, which returns the topic distribution of the text. TweetMoods then presents the topic distribution in a column chart. See Figure 5.3 for a demonstration of this feature. In this case, the user chooses to infer the topic distribution over the text *galaxy and election* with an Author-topic

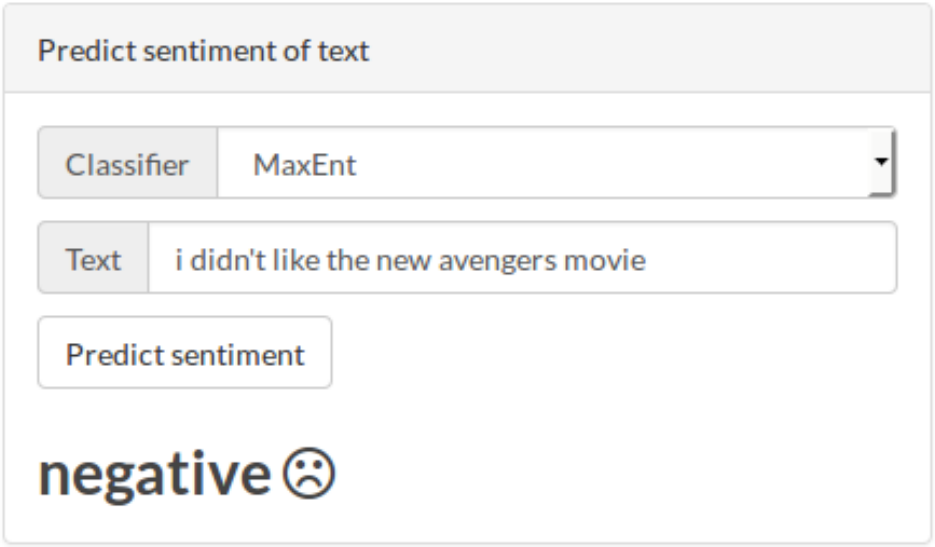


Figure 5.2.: Screenshot of the *sentiment prediction* feature of TweetMoods. The user chooses an arbitrary text to classify.

model created from the tweets of 16 Twitter users. Two topics are returned, each being represented by its 10 most probable words.

Search for tweets The *Search for tweets* component is the main feature of TweetMoods, providing both sentiment analysis and topic modeling of tweets retrieved by a search query. Figure 5.4 shows the search interface being presented to the user. The users first choose whether they want to analyze only the sentiments, topics or both, before choosing which classifier and topic model to use for the sentiment classification and topic distribution inferral. By clicking the *search* button, a request is sent to the API. The Python program running the API then retrieves the wanted tweets using the Twitter Search API. The tweets are either sentiment classified or inferred a topic distribution for, or both (depending on the user's preferences). The topic distribution is calculated as the *average* topic distribution over the retrieved tweets, each topic being represented by its 10 most probable words. The tweets are also listed, with background colors indicating whether the tweet was classified as positive, neutral or negative. The user can also choose to update the topic model with the tweets retrieved from the search, providing incremental updates to the model (Section 2.3.1 gives an introduction to online topic models). See Figures 5.6, 5.5 and 5.7 for the results of clicking the *Search* button with the parameters shown in Fig-

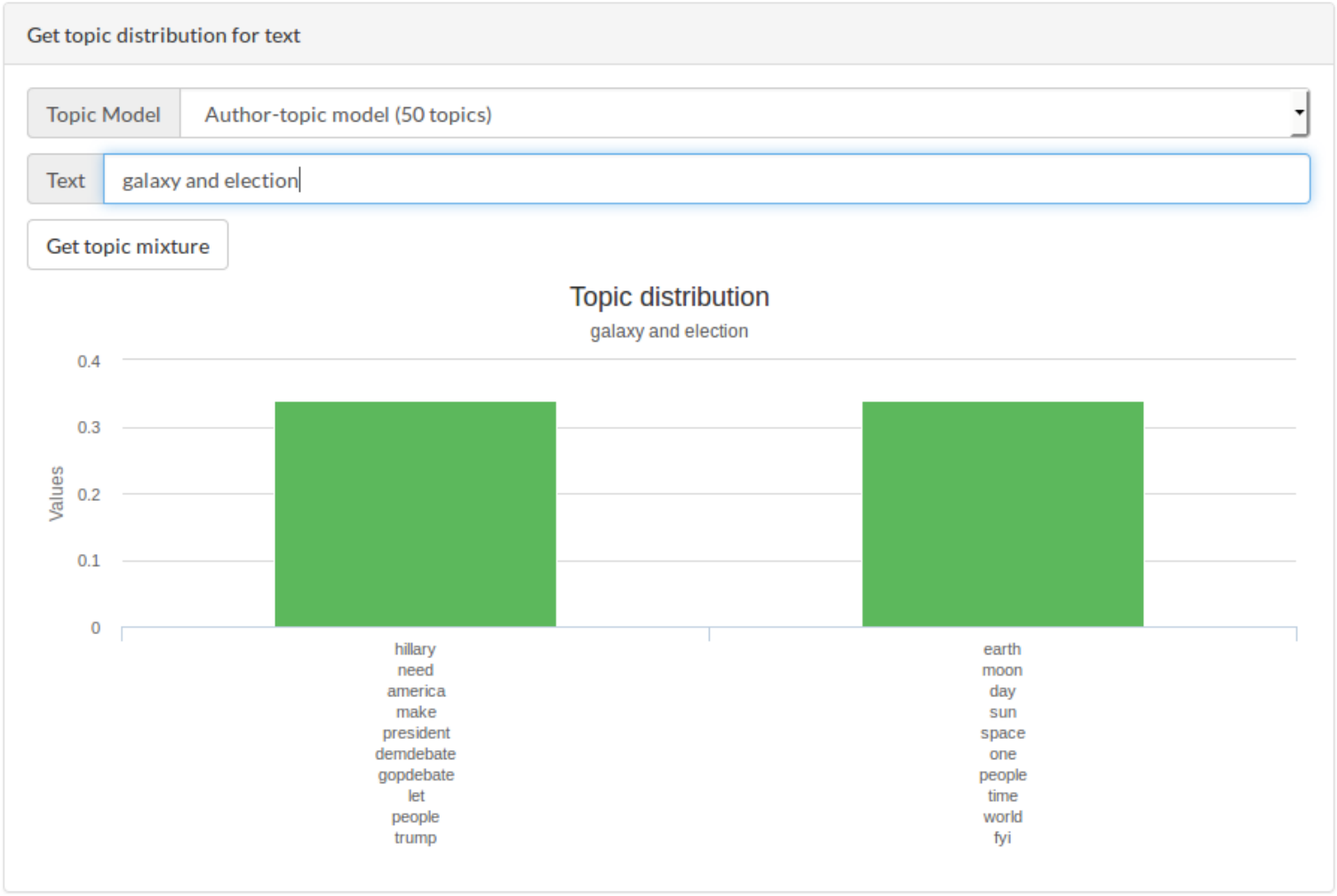


Figure 5.3.: Screenshot of the topic distribution feature of TweetMoods. The user provides a text to infer the topic distribution for.

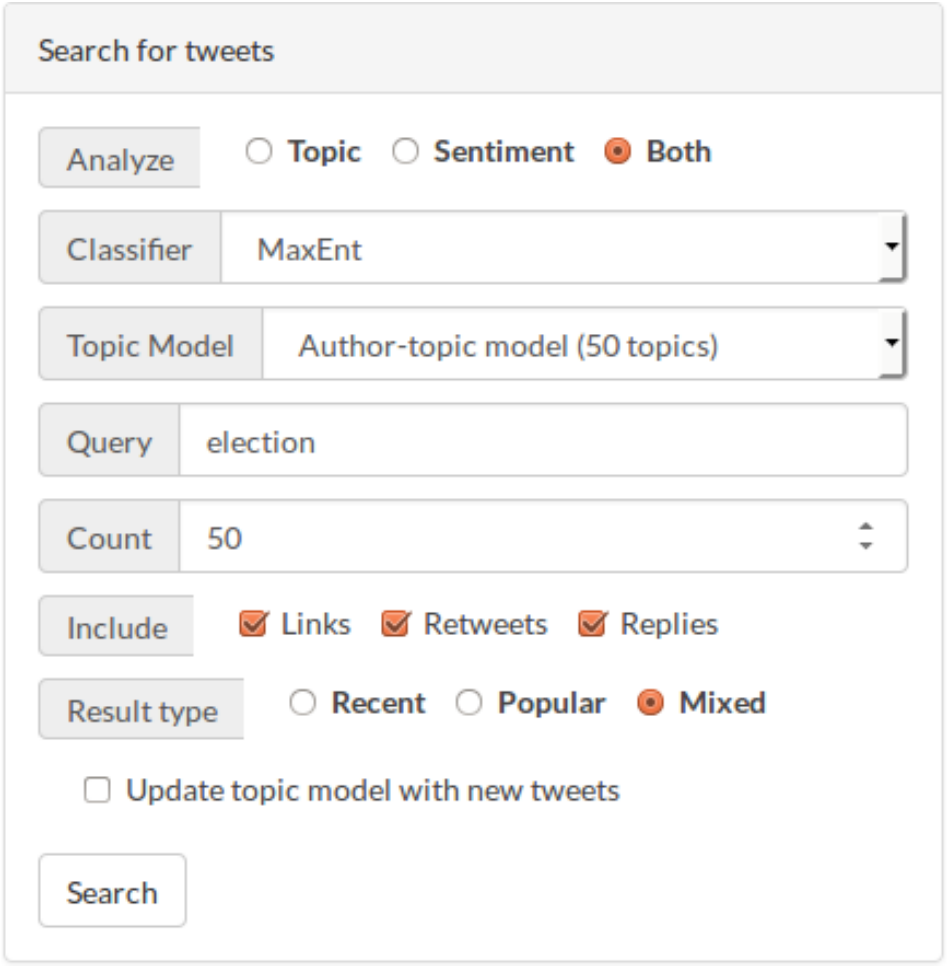


Figure 5.4.: Screenshot of the tweet search box. The user can analyze the sentiments, topics or both of queried tweets.

ure 5.4. In the scenario pictured in the figures, the user chooses to analyze both the topics and the sentiments of a tweet corpus retrieved by the query *election*. The chosen classifier is *MaxEnt*, and the topic model is chosen to be the pre-trained Author-topic model with 50 topics. Moreover, the user chooses to retrieve 50 tweets before applying the sentiment and topic modeling algorithms to them. The user also wants to include links, retweets and replies, with a mixed *result type*, which means that both recent and popular tweets will be returned. Finally, the user chooses not to update the pre-trained topic model with the retrieved tweets.

Figure 5.5 shows a sample of the 50 tweets retrieved from this search, with background colors indicating which sentiments the tweets were classified as. One can see that not all tweets were classified correctly. Tweet number 2 should probably be negative instead of neutral. Tweet number 3 and 5 is probably more neutral than positive overall, but words like *Good morning* and *I had a dream* could indicate positive sentiment. Tweet number 8 is clearly negative instead of positive, and after training the *MaxEnt* classifier on a more extensive set of training data and rerunning the sentiment classification on this tweet, it was classified as *negative*.

Figure 5.6 shows the sentiment ratio of all these 50 tweets, giving a result of 20% positive tweets, 24% negative tweets and 56% neutral tweets. Figure 5.7 shows the average topic distribution, using the pre-trained 50-topic Author-topic model, inferred from these 50 tweets. The three most probable topics are shown, each being represented by its 10 most probable words.

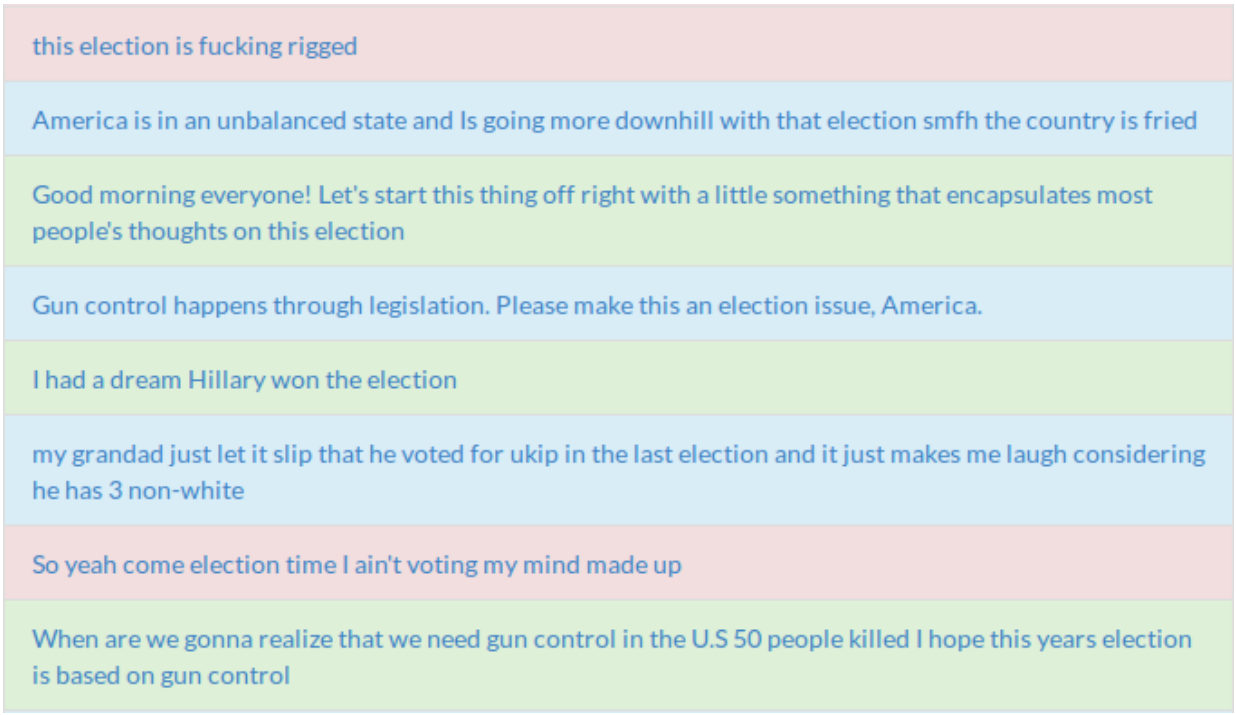


Figure 5.5.: A sample of tweets retrieved by TweetMoods when searching for *election*. Green indicates positive sentiment, blue indicates neutral, and a red background color indicates that the tweet was classified as negative.

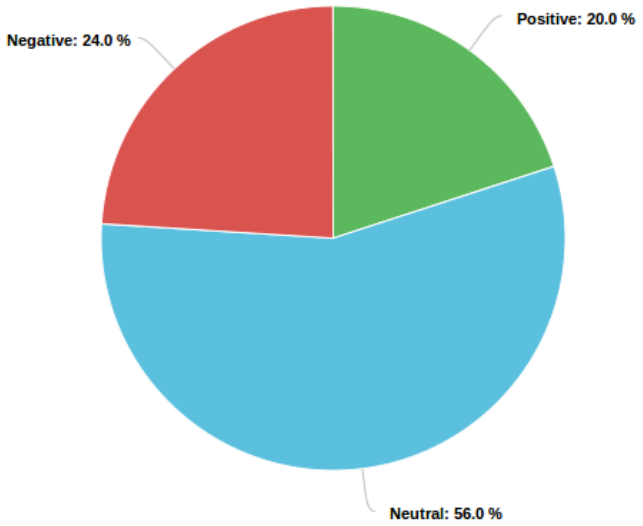


Figure 5.6.: Pie chart generated by TweetMoods, showing the sentiment ratio of 50 tweets retrieved from a search on *election*.

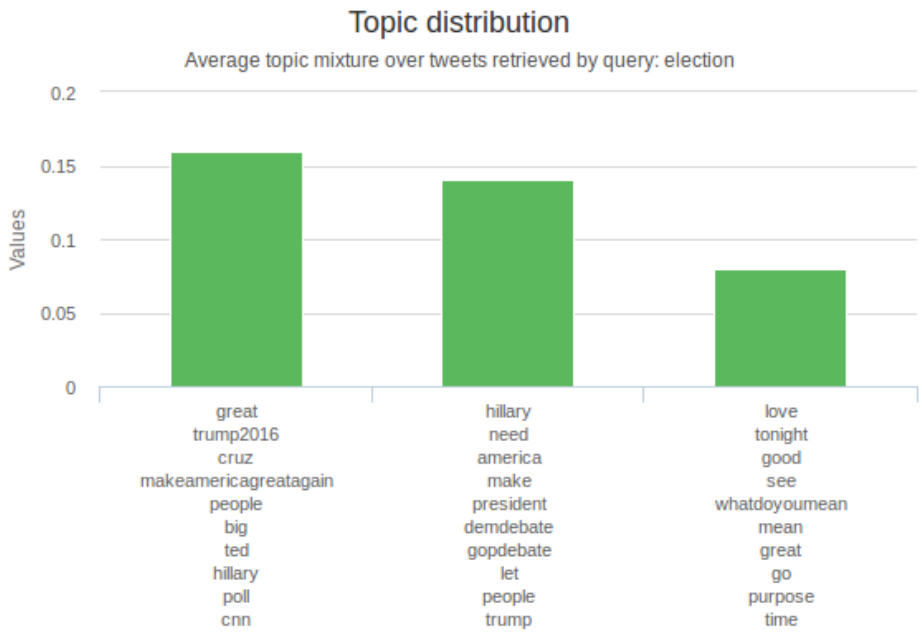


Figure 5.7.: The topic distribution inferred over 50 tweets retrieved by a search on *election*.

5.2. Polarity classification application

We built a web application for easing the manual classification of tweets. This application was developed using the Javascript web framework MeteorJS (see Section 3.8). The intention of this application is to provide a way to easily perform manual labeling of tweets. In addition to the fields provided by the Twitter API, we added fields for keeping track of the labels set by the user.

The web application presents one tweet at a time, with buttons corresponding to possible labels. When a user clicks one of the buttons, the tweet object is updated with the chosen label, and the application automatically skips to the next tweet. By allowing users to log in, we could keep track of which labels each user assigned each tweet, which could help resolve any ambiguity. Figure 5.8 shows the process of fetching a tweet through the Twitter API and displaying it to users who classify it.

Initially, we intended to use this application for creating a corpus with sentiment-classified tweets. However, since we already had access to labeled corpora, we considered this to be superfluous. We did, however, use this application for creating a corpus separating high-quality tweets from low-quality tweets, used in the Super Bowl experiment (Section 8.1.1) on Page 57. Figure 5.9 shows a screenshot of the working application.

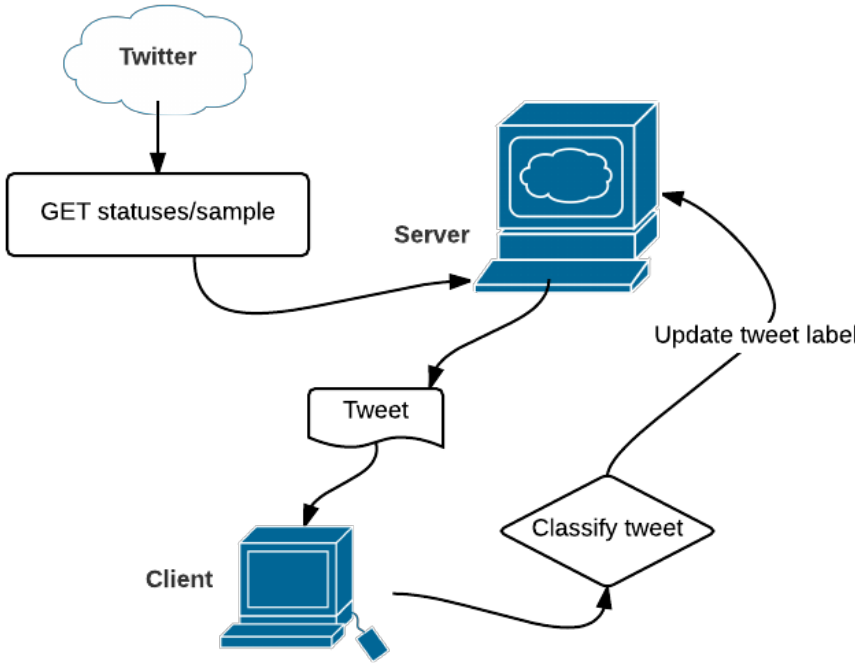


Figure 5.8.: The process of manually labelling tweets. Tweets are fetched from the Twitter API and displayed to the users who labels the tweets.

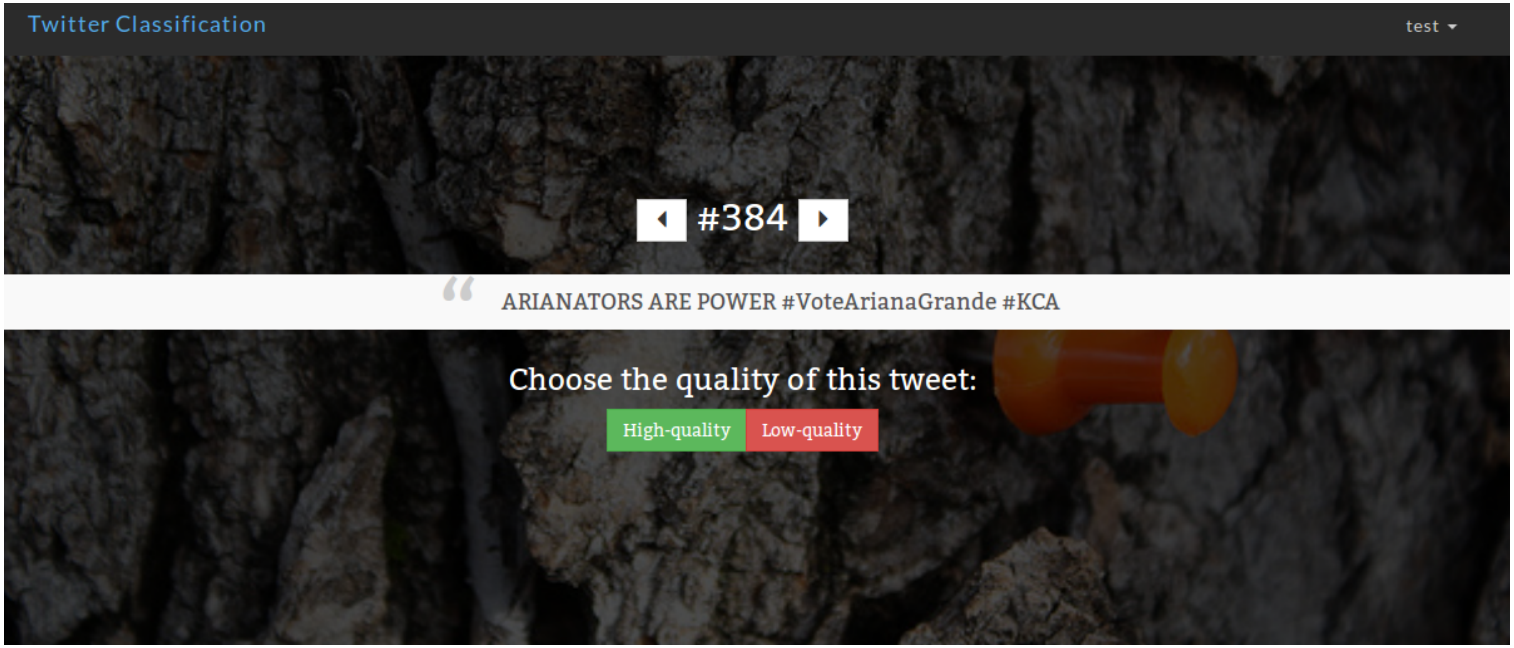


Figure 5.9.: Screenshot of the classification application.

6. Topic Modeling

Topic models are statistical methods used for representing the latent topics in a document collection. These probabilistic models usually present the topics as multinomial distributions over words, assuming that each document in the collection can be described as a mixture of these topics. A variety of topic models have been developed for the purpose of text analysis, and recently the analysis of tweets has become popular. The language used in tweets is known for being informal, often containing grammatically incorrect text, slang, emoticons and abbreviations, making it more difficult to extract topics from tweets than from more standard documents, like news articles or scientific journals.

Although the informal language and sparse text makes it difficult to retrieve the underlying topics in tweets, Weng et al. [2010] found that LDA (Latent Dirichlet Allocation) produced decent results on tweets. Quan et al. [2015] show a topic model tailored for noisy and sparse data, a phenomenon that is prevalent in tweets. Hong and Davison [2010] compare the LDA topic model with an Author-topic model for tweets, finding that the topics learned from these methods differ from each other. There are loads of approaches to topic modeling given different types of data, although we in this thesis specifically investigate techniques for modeling tweets. There are several modeling schemes that needs to be considered when modeling data: standard LDA models documents as a distribution over topics, and the Rosen-Zvi et al. [2004] Author-topic model models both authors and documents as a mixture of topics. Short documents don't provide satisfactory term co-occurrence. Therefore, pooling techniques (which involve aggregating related tweets into individual documents) might improve the results produced by standard topic model methods. Pooling techniques include, among others, aggregation of tweets that share hashtags and aggregation of tweets that share author.

6.1. Clustering tweets

Cluster analysis is an important part of topic modeling, and we will in this section demonstrate how different corpora of tweets generate particular clusters.

Number of clusters An important task related to topic modeling is determining the number of clusters to use for the model, which is denoted by k. There is usually not a single correct optimal value, and the answer is often ambiguous. One must find a balance between maximum compression, where the whole corpus is one cluster, and maximum accuracy, where the number of data points equals the number of clusters. Too few clusters will produce topics that are overly broad and too many clusters will result in overlapping or too similar topics. One common way to estimate the optimal number of clusters in a dataset, is by using the Elbow method. This is done by running k-means clustering on the dataset for different values of k, and then calculating the sum of squared error (SSE) for each value, as seen in Equation 6.1.

$$SSE = \sum_{i=1}^{n} (y_i - (f(x_i))^2 \tag{6.1}$$

The SSE decreases with higher values of k, until the SSE reaches 0 when the k-value is equal to the number of datapoints in the dataset. The idea behind the Elbow method is to increase the k until further increasing of k does not give us much better modeling of the data, which is presumed to be at the *elbow* of the graph. To achieve this, we use LSI (Latent semantic indexing) as a dimensionality reduction method, to model the data in two dimensions. Figure 6.2 shows the relationship between the terms and concepts contained in 91,404 tweets posted on January 27th, 2016. Tweets that are similar appear close to each other in the graph.

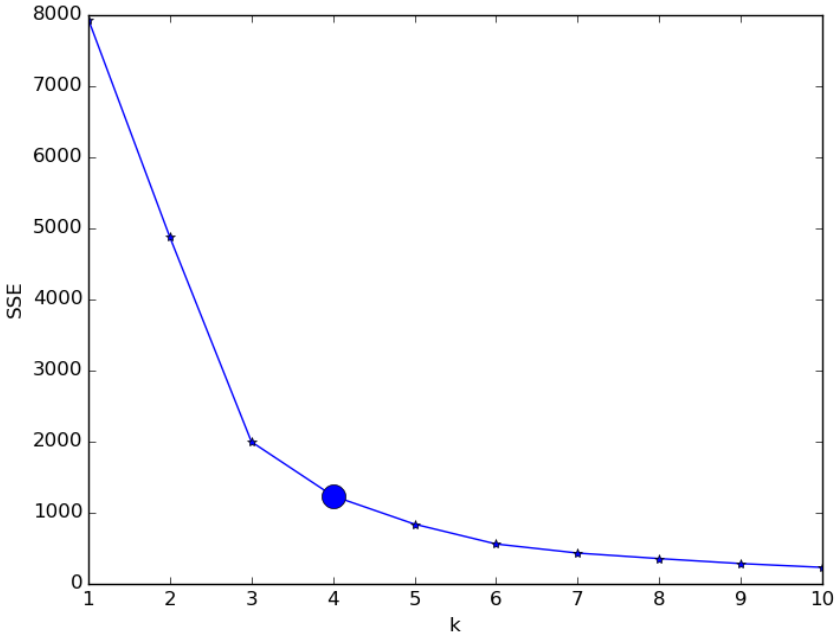


Figure 6.1.: The Elbow method. The elbow appears at k = 4.

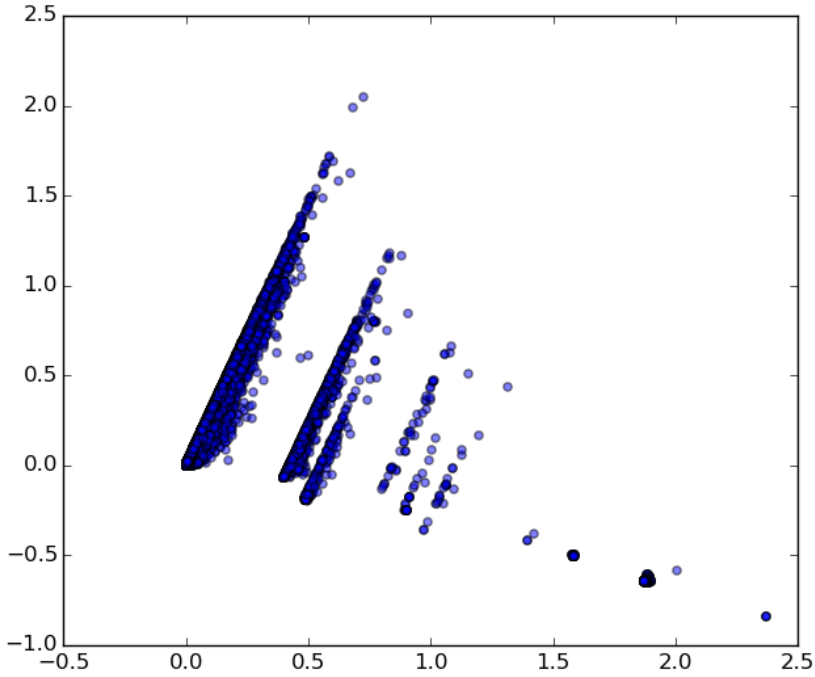


Figure 6.2.: Clusters of 91,404 tweets.

To find out where the elbow appears, we plot the SSE for different values of k. As seen in Figure 6.1, the elbow appears at $k = 4$, which indicates that 4 should be a reasonable number of topics for this particular dataset. It is apparent from Figure 6.1 that increasing the k further does not give a much better modeling of the data.

There are also other methods for estimating the optimal number of clusters in a dataset. Specifically for text datasets, Can and Ozkarahan [1990] proposes that the number of clusters can be expressed by the formula $\frac{mn}{t}$, where m is the number of documents, n is the number of terms and t is the number of non-zero entries in the document by term matrix.

Stability analysis Greene et al. [2014] address the issue of choosing the optimal number of topics by proposing a term-centric stability analysis strategy, their idea being that a topic model with an optimal number of clusters is more robust to deviations in the dataset. By repeatedly clustering different subsets of data originating from the same data set and

comparing the resulting topics, one can see whether or not the topic model is *stable* for a given value of k. They found that their method provides a useful guidance in choosing the number of topics for a topic model. It is, however, worth mentioning that they performed their research on news articles, which are much longer and usually more coherent than tweets. Greene et al. [2014] of the paper have also developed a Python implementation[1] of this stability analysis approach, which we used to predict the optimal number of clusters for one of our Twitter datasets. It generates topics for each k value in a range specified, before calculating the stability score for the topic model related to each k value. Greene et al. [2014] paper use NMF (Non-Negative Matrix Factorization) as their topic modeling method; however, their method holds true for any topic modeling method that represent topics as a ranked list of terms, including LDA. Here follows our experiment using stability analysis to estimate the number of topics in a corpus of tweets. We used a dataset containing 10.000 tweets posted the 27th of January 2016, using the range $\{k_{min} = 2, k_{max} = 10\}$ for the number of topics. An initial topic model is first generated from the whole dataset. Proceedingly, τ random subsets of the dataset is generated. For each of the subsets $S_1, ..., S_\tau$, one topic model per k value is generated. We chose $\tau = 50$, resulting in generating $\tau * k_{range} = 50 * 9 = 450$ topic models. The stability for a k value is then generated by computing the mean agreement score between the reference set and the sample sets for k, as shown in Figure 6.2, where $agree(S_0, S_i)$ denotes the agreement score between the reference ranking set S_0 and ranking set S_i. The number of terms to consider, denoted by t, also affect the agreement score. A t value of 20 indicates that the top 10 terms for each topic were used.

Figure 6.3 shows the stability score for the k values 2 through 10 for $t = 20$. The low scores are likely caused by the sparse and noisy data in tweets, as this method was originally used for longer, more coherent documents. The estimation of number of topics by this method does therefore not provide a good indication of the number of underlying topics for this kind of corpus.

Researchers have also achieved good results with using a higher number of topics for tweets than what is needed for larger, more coherent corpora, since short messages and a high number of tweets with diverse themes requires more topics. Hong and Davison [2010] uses number of topics for tweets in a range from 20 to 150, obtaining the best results for $t = 50$, although the chance of duplicate or overlapping topics increase with the amount of topics. We performed an experiment to investigate how different

[1] https://github.com/derekgreene/topic-stability

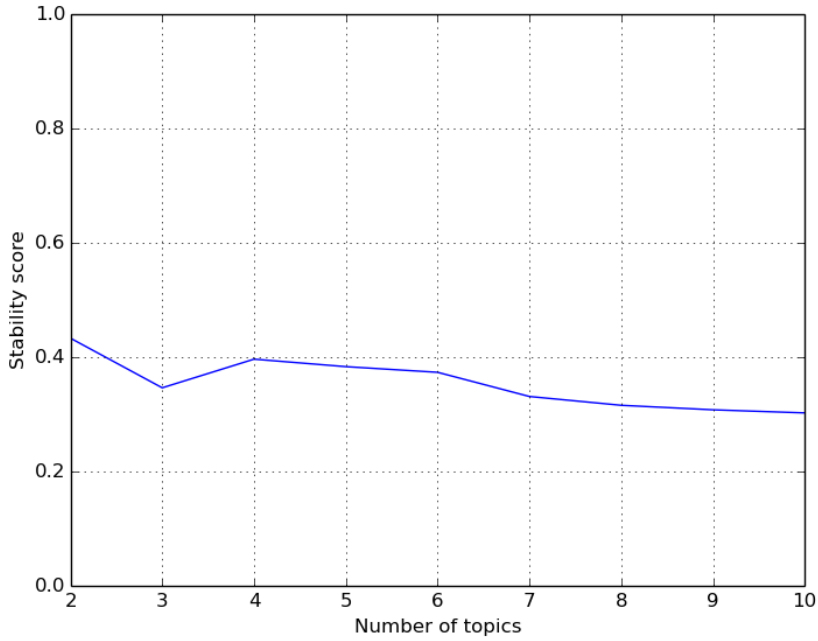


Figure 6.3.: Stability analysis plot for a corpus of 10.000 tweets.

$$stability(k) = \sum_{i=1}^{\tau} agree(S_0, S_i) \tag{6.2}$$

Figure 6.4.: Stability score, Greene et al. [2014]

tweet corpora produces different clusters. This experiment can be seen in 8.1.1.

6.2. Hashtags

Tweets have user-generated metatags which can aid the topic and sentiment analysis. Being labels that users tag their messages with, hashtags serve as an indicator of which underlying topics are contained in a tweet. A hashtag is usually a word or an abbreviation preceded by the hash character (#). During the Super Bowl, hashtags like *#SuperBowl* and *#SB50* were used extensively, *SB50* being an abbreviation for Super Bowl 50. If users attend an event, they might tweet about the event using a hashtag that indicates

that they are tweeting about that particular event. Searching for hashtags makes it easier to look up tweets about certain topics or themes. Moreover, hashtags can help discover emerging events and breaking news, by looking at new or uncommon hashtags that suddenly get a rise in attention. We will here present the usage of hashtag co-occurrences as a way to divulge the hidden thematic structure in a corpus of tweets, using a collection of 3 million tweets retrieved during the Super Bowl match, February 7th, 2016. Hereinafter, this tweet collection will be referred to as the *Super Bowl corpus*.

Hashtag co-occurrence Hashtag co-occurrences show how hashtags appear together in tweets. Since different hashtags appearing in the same tweet usually share the underlying topics, a hashtag co-occurence graph might give interesting information regarding the topics central to the tweet. Looking at which hashtags co-occur with the hashtag *#SuperBowl* gives us more information about other important themes and topics related to Super Bowl. Table 6.1 shows the 10 most popular hashtags from the Super Bowl corpora, about half of them being related to the Super Bowl. *SB50*, *SuperBowl50* and *SuperBowl* are all variations of the Super Bowl hashtag. Interestingly, *Broncos* (Denver Broncos), which was the winning team of this year's Super Bowl, was the 5th most mentioned hashtag, occurring 1290 times, while the losing team *Panthers* (Carolina Panthers) is mentioned only 476 times, being the 17th most popular hashtag that day. Some hashtags are related to a topic without it being apparent, since it requires further knowledge to understand how they are related, which is the case with the *KeepPounding* hashtag. A Google search reveals that "Keep Pounding" is a quote by the late Carolina Panthers player and coach Sam Mills. Hashtag co-occurrences help reveal such related hashtags. Figure 6.5 displays the co-occuring network graph for the Super Bowl corpus, revealing the hashtags *KeepPounding* and *SB50* to be the 8th most co-occurring hashtag pair, appearing together in 186 tweets. The hashtag co-occurrence network also tells that the hashtag *EsuranceSweepstakes* co-occurred with *SB50* 215 times, indicating that *EsuranceSweepstakes* is related to the Super Bowl. A Google search again reveals that this is the case; Esurance is a company that had a commercial during Super Bowl, and in the commercial they encouraged people to tweet using the hashtag *EsuranceSpeestakes*, which explains the high amount of that particular tweet.

Figure 6.5 displays the 20 most co-occurring hashtags in a co-occurrence network for the Super Bowl corpus, with the 20 most frequently co-occuring hashtag pairs. One can see three clusters of hashtags, with Super Bowl related hashtag being the largest one. Related topics and terms become more

Hashtag	Frequency
SB50	10234
KCA	3624
SuperBowl	2985
FollowMeCameronDallas	1899
Broncos	1290
EsuranceSweepstakes	1079
followmecaniff	995
FollowMeCarterReynolds	938
KeepPounding	794
SuperBowl50	783

Table 6.1.: Most occurring hashtags during Super Bowl 2016.

apparent when displayed in a co-occurrence graph like this. For example do the artists *Beyonce* and *Coldplay* appear in the Super Bowl cluster, which is because they both performed during Super Bowl's halftime show. Another cluster that appeared consists of the three hashtags *votelauramarano*, *KCA* and *VoteArianaGrande*. A Google search again reveals that KCA is an abbreviation of *Kid's Choice Awards*, an annual award show where people can vote for their favorite television, movie and music acts. Viewers can vote by tweeting the hashtag *KCA* with a specific nominee-specific voting hashtag (e.g., *VoteArianaGrande*).[2] Researchers have found that a hashtag-graph based topic model enhances the semantic relations displayed by standard topic model techniques [Wang et al., 2014].

6.3. Author evaluation

In Section 2.3.4, we introduced some previously proposed methods for evaluating the accuracy of a topic model. Here follows a method that we propose for evaluating an Author-topic model, specifically with the Twitter domain in mind. A topic mixture for each author is obtained from the model. The subjects should know the authors in advance, and have a fair understanding of which themes and topics the authors are generally interested in. The subjects are then presented a list of the authors, along with the topic distribution for each author (represented by the 10 most probable topics, with each topic being represented by the 10 most probable words). The task of the subject is to deduce which topic distribution belongs to which user. The idea is that coherent topics would make it easy to recognize the

[2] http://kca.nickelodeon.fr/info/info-promo-voting-rules/s1ws9a

authors from a topic mixture, as the authors' interests would be reflected by the topic probabilities. An experiment using this evaluation method is conducted in Section 8.1.4.

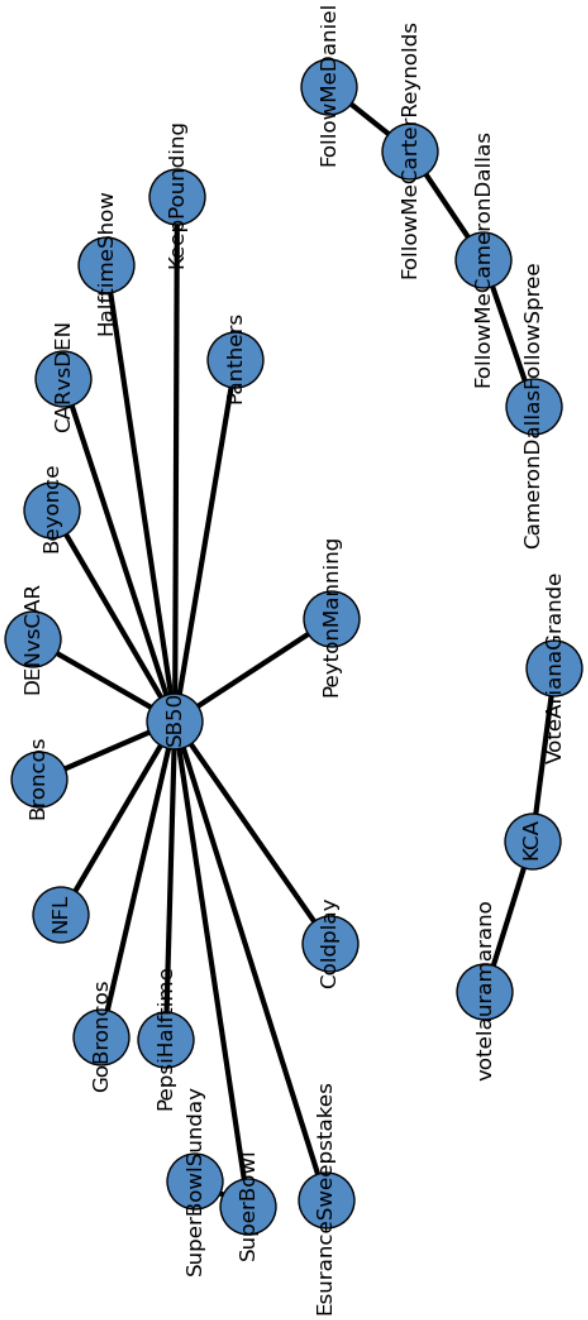


Figure 6.5.: Hashtag co-occurrence network for tweets posted during Super Bowl 2016

7. Sentiment Analysis

7.1. Twitter Sentiment Classifier

For our Twitter sentiment classifier, we've chosen to work with the world-wide popular Python programming language because of the immense library of modules and packages available for different applications. One of these is the Scikit-learn module which is a machine learning library earlier described in Section 3.2. The basic steps of the Twitter sentiment classifier can be summarized in three steps. First is the textual preprocessing of the data retrieved from Twitter, labeled data for training and raw data for predicting. Second is the feature extraction of the Twitter data, which the classifiers train on. And lastly, the training and prediction of the classifiers on the input data into three different classes; positive, neutral or negative sentiment regarding the text.

7.1.1. Textual preprocessing of Twitter data

For the textual preprocessing, some of the tweet contents are considered unnecessary and removed completely for all tasks. All usernames, in the tweet recognized as *@"username"*, are removed as they would most likely become considered as more important tokens than they are to the classifiers. URLs are removed, as their contents are not taken into consideration in this system. Elongated words with letters repeated more than three times are reduced to only occur up to two times. Negation contractions (such as *wasn't, isn't, etc.*) are expanded into their representations (e.g. *was not, is not, etc.*, respectively). Because of the informal language used on Twitter with a lot of strange capitalization, all text is made lower cased for simplification. Hashtagged words are added to its tweet in addition to the original hashtag (e.g. #obama and obama). Further, redundant whitespace and english stop words are removed. Symbols that are not possible to create on a standard English keyboard are also removed.

7.1.2. Feature Extraction and building a feature set

The feature set of the sentiment classifier was built with Scikit-learn's FeatureUnion (see Section 3.2.2. FeatureUnion simplifies the procedure of

building a matrix containing the features of all documents, by splitting the work to be done down to several transformers that take textual data as input and outputs a matrix (part of the final feature matrix).

Term frequency-inverse document frequency (TF-IDF) Transformers: word-level & character-level

The TF-IDF transformer transforms each document into a vector with TF-IDF scores for each token. The transformer assumes the documents processed to be English and does not include English stop words in its calculation. All tokens are converted into lower case, and includes a specified number of the most informative features. The higher the number of features, the more computation is needed to create the feature vector representing each document. An unspecified number will result in all tokens being features. It is common to set a high number of features for the TF-IDF transformer based on the Twitter corpora containing a lot of slang words in addition to standard words. How TF-IDF is calculated can be found in Section 2.1.1.

The system uses two separate versions of TF-IDF transformers. One transformer on the word-level, and one on the character level. The word-level transformer reduces each document representing a tweet to a vector with TF-IDF scores for unigrams, bigrams, up to a specified n-grams. The character-level transformer does the same but for characters on a specified combination of n-grams.

Lexicon Transformer

The Lexicon Transformer traverses through each word of each input document for whether or not contain tokens that are included in the sentiment lexicons. The lexicons used for this quantification are Bing Liu's Opinion Lexicon, NRC Emotion Lexicon and the MPQA Subjectivity Lexicons (see Section 3.3). The lexicon transformer gives the input document a sentiment score based on the tokens' values in the sentiment lexicons. Negated tokens are handled by checking whether the negation tag *_NEG* is appended the given token. The negated tokens (e.g. *not good – good_NEG*) will get the negative sentiment score, and are placed in on of the two latter indices of the lexicon vector. Normal affirmative tokens are placed in the two first indices of the lexicon vector, depending on it's sentiment score.

For a document collection with documents d, from i to m, with tokens of each document j to n, the lexicon score is a list of lexicon scores for for each document containing the scores:

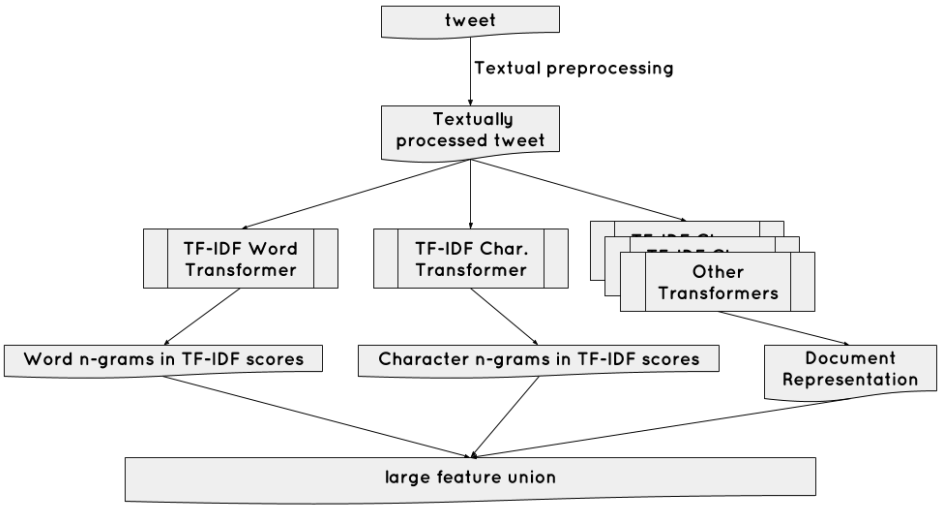


Figure 7.1.: Simple visualization of creating a feature vector representing each tweet. These feature vectors tend to grow immensely in size proportionally with the size of the dataset.

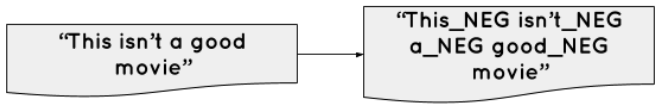


Figure 7.2.: Simple visualization of attaching negation tags to a tweet.

$$
\begin{aligned}
LexiconScore(d_i) = [& \\
& \textstyle\sum_j^n pos\_score(d_{i,j}), \\
& \textstyle\sum_j^n neg\_score(d_{i,j}), \\
& \textstyle\sum_j^n pos\_negated\_score(d_{i,j}), \\
& \textstyle\sum_j^n neg\_negated\_score(d_{i,j}), \\
&]
\end{aligned}
$$

For each document, the lexicon score is calculated for each of the three lexicons. Each of these three scores are added together using matrix addition, which is the final transformed vector representing the document.

Emoticon Transformer

Emoticons are representations of facial expressions using textual symbols, which is most often used for expressing mood in a text. Emoticons can therefore be valuable in the process of calculating sentiment in a document. The emoticon transformer is simple. It takes a document collection and preprocesses the documents with a script that converts all unicode emoticons to ascii emoticons. Next, it finds all emoticons using a regular expression search, and outputs the number of happy and sad emoticons found in the document as a vector.

Negation Count Transformer

Negation is a common feature of expression, and must be accounted for in a program that estimates the sentiment of a document. The TF-IDF transformer supports adding negation tags to terms by naively adding the tag *_NEG* to the preceding and three succeeding words.

7.1.3. Training classifiers and predicting textual input

For classification, we have experimented with several different classifiers. The classifiers are imported from the Scikit-learn module, which are state-of-the-art machine learning algorithms that are regularly updated. From

this library, we have chosen to work with SVC which is a Support Vector Machines (SVM) machine learning method. Logistic Regression, also known as a Maximum Entropy machine learning method. We've also worked with a couple of Naive Bayes classifiers, the Multinomial Naive Bayes (MNB) and the Bernoulli Naive Bayes - because of their simplicity. All of these machine learning algorithms have been proven by numerous papers (some presented in Section 4 to be effective at predicting categories for big amounts of textual data. These machine learning methods are passed feature sets in accordance with the previous Section 7.1.2, which is used for training on labeled data or prediction of unlabeled data.

8. Topic Modeling Experiments

This chapter describes the experiments conducted during this thesis, as well as displaying the results. Section 8.1 contains the topic modeling experiments, and Section 9 contains the experiments related to sentiment analysis.

8.1. Topic Modeling

We will in this chapter conduct several experiments related to topic modeling. First, we perform a cluster analysis experiment, comparing the clusters generated by different types of documents. Further, we perform several experiments for different types of topic models; one standard LDA topic model, one hashtag-aggregated topic model and two author-topic models.

8.1.1. Clustering experiment

To compare the clusters generated by different tweet corpora, we wanted a corpus of tweets that included many similar tweets. Moreover, to make sure that there would be a lot of tweets about similar topics, we decided on retrieving tweets posted during the Super Bowl 2016 match. Since this is a very popular event, we assumed that there would be an extensive amount of posts regarding this match on Twitter, which could be interesting to analyze. In the timespan from February 7, 16:32 to February 8, 13:05 (times are in GMT), we obtained 357,120 tweets (after removing non-english tweets), containing 3,634,977 disctinct words.

After retrieving the tweets, we wanted to filter out spam as well as tweets that didn't contain any clearly defined topics. This was done by manually classifying 1000 of the tweets, picked by random, as either *high-quality* or *low-quality*. This was an interesting task, as the definition of what makes a tweet high-quality or low-quality is often ambiguous, but we generally tried to distinguish tweets about Super Bowl or any other definite topic from vague tweets where it's hard to grasp the context. Here are examples of which tweets were considered *high-quality* and *low-quality*:

high-quality "Bob Sanders deserved Manning's first Super Bowl MVP. If Manning steals one from Von Miller I'll be pissed. A total NFL thing to do."

low-quality "i dont have the time lol"

Preferably, these 1000 tweets should be classified by more than two users. We considered crowd-sourcing the manual classification by using the *polarity classification application* introduced in Section 5.2. However, due to time and manpower constraints, we ended up manually classifying them ourselves, both being second-language English speakers. In total, 792 of the 1000 tweets were manually classified as *low-quality*, and the remaining 208 tweets were considered *high-quality*. For tweets where the sentiment was unclear, we discussed it among ourselves. 800 of these tweets were included in the training set, and 200 were used for the test set. A Naïve Bayes classifier was used for training the tweets, using word occurrence as the only feature. This was because we only needed to roughly filter the tweets to distinguish tweets that are somewhat similar to each other from the remaining tweets.

Out of the 200 tweets in the test set, 42 of them were *high-quality* and 158 of them were *low-quality*. All of the low-quality tweets were classified correctly, but only 6 of the high-quality tweets were, which shows that there were many false positive *low-quality* tweets. However, since we manage to filter out all low-quality tweets by using this classifier, we could use this on the whole set of 357,120 tweets to remove all tweets classified as low-quality, and be fairly sure that the remaining tweets are high-quality, even though we are likely to remove many high-quality tweets as well.

We were left with 12,306 tweets after running the classifier on the whole set and only returning the high-quality ones. Many of them seems to be about the Super Bowl, which was expected, seeing as a big share of the original set of high-quality tweets were about this topic. Some of them are indirectly related to Super Bowl, mentioning Lady Gaga and other artists that performed during Super Bowls halftime show. We wanted to compare the clustering of this corpus of tweets that are similar to each other to the clustering of the corpus shown in Figure 6.2. Figure 8.1 shows the cluster of the *good* tweets. As opposed to the random tweets shown in Figure 6.2, there are no clear gaps here, which was expected. To further show the difference in cluster structure generated by different corpora, we also include the cluster of the *low-quality* tweets. To avoid the result being biased by the *low-quality* corpus containing much more tweets than the *high-quality* corpus, we only included 12,306 tweets from the *low-quality*

corpus. Since the classifier yielded a lot of false positive *low-quality* tweets, it is likely that many of the tweets in the *low-quality* corpus should belong to the *high-quality* corpus instead. We do, however, see a clear distinction between the clusters, with the low-quality tweets indicating a more disperse corpus.

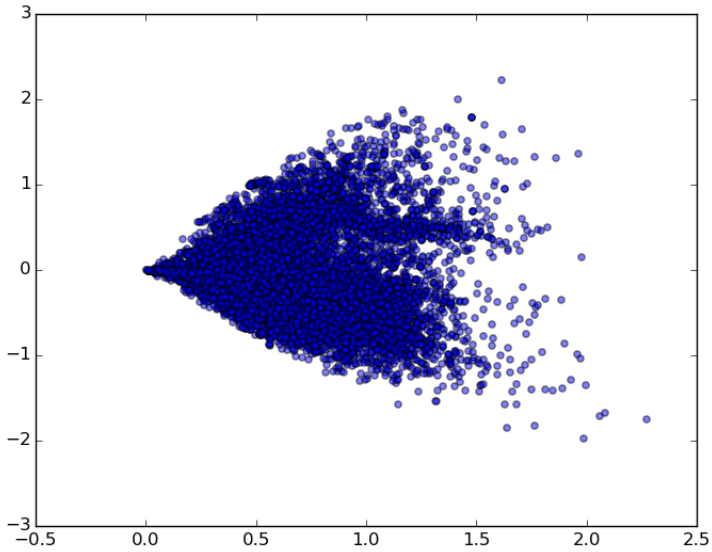


Figure 8.1.: Cluster of 12,306 *high-quality* tweets.

8.1.2. Topic Modeling based on raw tweets

The following topic model experiment was carried out on a corpus containing 500,000 English tweets posted between the 10th and the 12th of May, 2016. Non-english tweets as well as all words with fewer than 20 characters were removed. The experiments using standard LDA were performed with k values of 10 and 50.

The field of topic modeling is relatively new, and there isn't a general agreement on how to evaluate topic models. Human evaluation is generally preferred to mathematical metrics when it comes to assessing the coherence or the interpretability of a topic model, although some quantitative approaches could be used as an estimation of a topic model's coherence (see Section 2.3.4 for an overview of some evaluation methods for topic models). Table 8.1 shows a sample of the topics generated for $k = 50$. Many of the topics are generally incoherent, which is consistent with recent research on

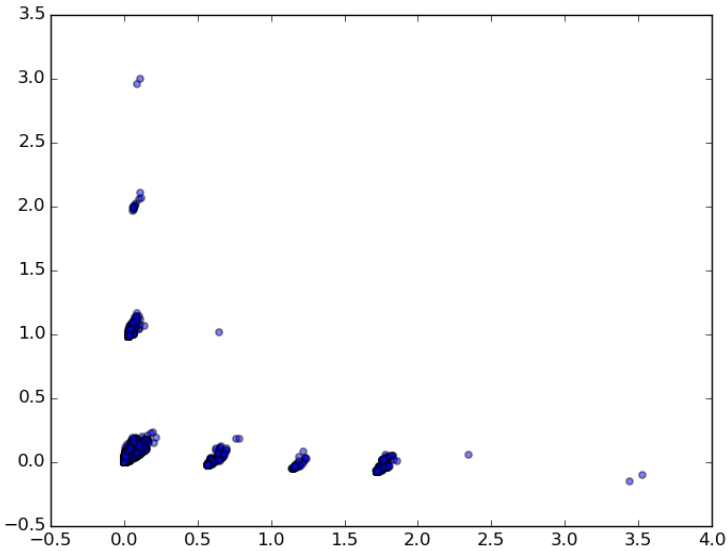


Figure 8.2.: Cluster of 12,306 *low-quality* tweets.

standard LDA applied to microblog data. It is, however, possible to gain some information from the topics. Topic #11 contains words related to the *Teen Choice Award*; however, the topics seem generally fairly incoherent to a human eye.

Table 8.2 shows the topics generated from the same corpus, but with $k = 10$ instead of 50. These topics are also incoherent and it isn't easy to infer a lot from the data. Moreover, a higher number of topics should be considered to depict the diversity of large corpora.

8.1.3. Hashtag-aggregated topic model

We apply a pooling technique that involves aggregating tweets sharing hashtags, the assumption being that tweets that share hashtags also share underlying topics. The main goal of this method is the same as for the Author-topic model and other pooling techniques; alleviating the disadvantages of short documents by aggregating documents that are likely to share latent topics. Some restrictions were made: We only used *single-hashtag* tweets, and we also only used hashtags that appeared in at least 20 of the documents in the corpus.

A sample of the resulting topics can be seen in Table 8.3. The topics appear more coherent than the topics generated on tweets as individual

Topic #7	Topic #11	Topic #31	Topic #44
go	teenchoice	may	watch
sex	nominee	tonight	hey
season	look	set	hillary
body	things	whole	following
proud	hate	date	saying
girlfriend	fans	nobody	dear
meeting	car	games	dad
parents	choicefandom	truth	room
human	united	eurovision	sick
happens	everybody	congrats	project

Table 8.1.: Four of the topics generated from the 500,000 tweets corpus with k=50.

Topic #1	Topic #6	Topic #8	Topic #10
free	video	get	thank
trump	new	long	please
new	youtube	cute	back
win	twitter	hair	amazing
now	photo	high	baby
last	lt	ya	everyone
girls	posted	nialofficial	found
instagram	thanks	hot	take
night	facebook	read	play
will	food	louis_tomlinson	also

Table 8.2.: Four of the topics generated from the 500,000 tweets corpus with k=10.

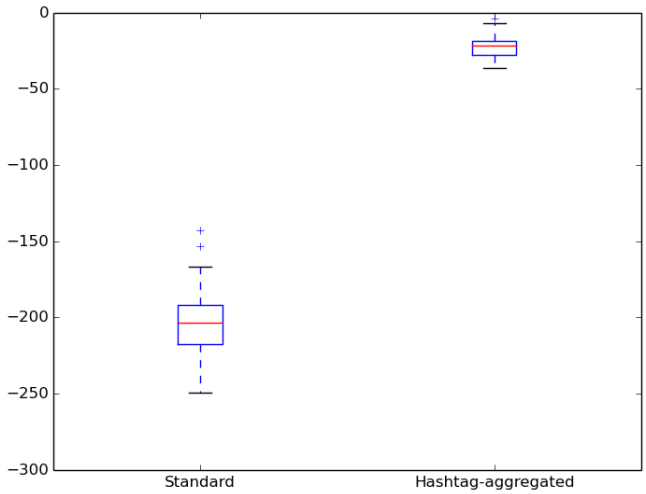


Figure 8.3.: Box plots comparing the coherence scores for the standard LDA topic model with the hashtag-aggregated topic model. The closer to zero the numbers are, the higher is the coherence.

documents; however, many of the less probable words in each topic might seem somewhat random. It is, however, easier to get an understanding of the underlying topics conveyed in the tweets, and aggregating tweets sharing hashtags can produce more coherence than a topic model generated by single tweets as documents. There might be *event-specific* hashtags that causes different events to show up in the topics. Two tours by respectively Selena Gomez and Justin Bieber occurred in two different topics, but some of the lower-probability words seems out of place. The UMass coherence scores for the topics in this topic model are also much higher than for standard LDA, and can be seen in Figure 8.3. The standard document scores are consistent with Mimno et al. [2011], while the hashtag-aggregated topic model show a much higher coherence.

8.1.4. Author-topic model experiments

Using the Twitter API, we obtained tweets from six popular Twitter users. We chose users that are known for tweeting about different topics, so that the results would be distinguishable. We would expect that Barack Obama tweets about different themes and topics than Neil deGrasse Tyson. While Barack Obama mostly tweets about topics related to politics, Neil deGrasse Tyson, being an astrophysicist and cosmologist, would assumingly tweet

Topic #7	Topic #21	Topic #24	Topic #34
revivaltour	new	purposetourboston	trump
selenagomez	soundcloud	justinbieber	hillary
whumun	news	boston	bernie
wtf	video	one	realdonaldtrump
getting	favorite	best	will
boyfriend	sounds	tonight	clinton
bitch	health	yet	greysanatomy
mad	blessed	shows	vote
resulted	efc	bitcoin	president
blend	mealamovie	redsox	people

Table 8.3.: Four of the 50 topics generated from the hashtag aggregated corpus.

about science-related topics. Moreover, we could expect the themes communicated by Barack Obama and Donald Trump to be quite similar, both being politicians. We also included tweets from two pop artists, Justin Bieber and Taylor Swift, expecting them to not share many topics with the previously mentioned users. Elon Musk, being an engineer and inventor, would expectedly share more similarities with Neil deGrasse Tyson than Justin Bieber or Taylor Swift. Since we wanted tweets that reflect the author's interests and views, we discarded all retweets and *quote tweets*,[1] since they aren't written by that particular user. Furthermore, we discarded all tweets containing media or URLs, as getting the context of these tweets require that you have access to the contents of the url or the media. We compare two approaches to author topic-modeling, one based on Rosen-Zvi et al. [2004] (hereinafter AT1) and the other based on Hong and Davison [2010] (hereinafter AT2). An introduction to these approaches is presented in Section 2.3.3.

AT1 experiment

The quality of an author-topic model can be measured in its ability to accurately portray the user's interests. A person that has knowledge of which themes and topics a user usually talks about, should be able to recognize the user by their topic distribution. We generated 10 topics from this author-

[1] Quote tweets are retweets that don't use the built-in retweet feature, usually by being copied and *quoted* instead.

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
will	just	earth	night	tonight	just	people	great	president	don
new	like	moon	today	love	one	much	thank	obama	fyi
now	will	day	get	thank	know	time	trump2016	america	time
get	now	sun	new	thanks	orbit	won	will	sotu	tesla
america	good	world	happy	ts1989	two	tonight	cruz	actonclimate	first
make	think	ask	back	taylurking	might	way	hillary	economy	rocket
poll	also	universe	time	show	long	bad	makeamericagreatagain	work	science
many	around	full	see	got	planet	show	big	change	space
trump	live	space	one	crowd	star	morning	cnn	americans	launch
don	landing	year	good	tomorrow	instead	really	said	jobs	model

Table 8.4.: The ten topics generated for AT1.

topic model, each topic being represented by the 10 most probable words. The resulting topics are reasonably coherent, and can be seen in Table 8.4. The topic distribution for each user can be seen in Figures 8.4 through 8.9. To evaluate how accurately the topic distributions represent the users, we asked 8 people to participate in the experiment (a thorough explanation of the evaluation method for this topic model is given in Section 2.3.4). Each participant examined each of the topic distributions without knowing which user it belonged to (the author's name was replaced by an ID).

All participants managed to figure out which topic distribution belonged to which author for all author's, except the distributions of Taylor Swift and Justin Bieber, which were very similar, both having Topic 4 and 5 as their most probable topics. These topics show a prevalence of words used in every-day language, conveying happiness and using words you would expect from pop artists like these users, such as *crowd*, *thanks*, *tonight*, *happy* and *good*. The remaining author's had easily recognizable topic distributions. Beside the obvious author-related words *president* and *obama*, Topic 9 include other tokens related to Barack Obama, like *american*, *actonclimate*, *economy*, *change* and *work*, indicating that the author having this topic as the most probable topic is Barack Obama. Among the most probable words occurring in Donald Trump's most probable topic (Topic 8), is *cruz* and *hillary*, referring to Ted Cruz and Hillary Clinton, two of his opponent's in the presidential election. Among the top words for this topic is also the token *makeamericagreatagain*, a hashtag version of his slogan *Make America Great Again*. By extending the words to including his two most probable topics, also including Topic 1, the words *make*, *america* and *great* also occur individually. The remaining two authors *Elon Musk* and *Neil deGrasse Tyson* are also easily recognizable from their topic distributions if you are familiar with their doings. Neil deGrasse Tyson, being an astrophysicist and cosmologist, had topics 3, 6 and 10 as his most probable topics, including words like *earth*, *moon*, *universe*, *planet*, *star* and *star*. Topic 10 is also Elon Musk's most probable topics, having words like *tesla*, *rocket* and *space* as its most probable words. People being familiar with Elon Musk being the CEO of Tesla and SpaceX, should be able to recognize his topic distribution, which was confirmed by the experimental results.

AT2 experiment

The Author-topic model proposed by Hong and Davison [2010] performs standard LDA on aggregated user profiles, a method they denote the USER scheme. The process is described as follows in their paper:

1. Train LDA on aggregated user profiles, each of which combines all

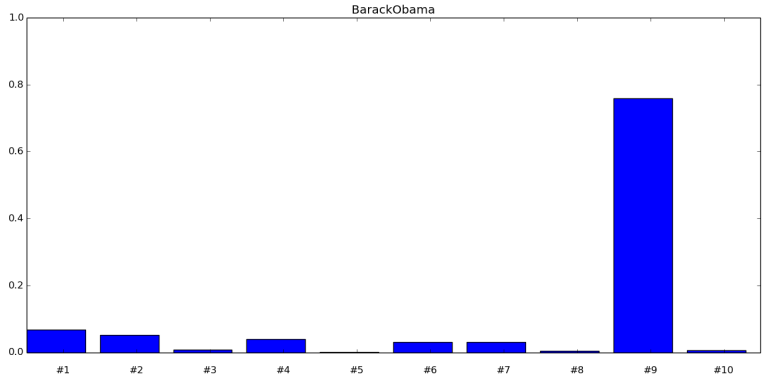


Figure 8.4.: AT1 topic distribution for Barack Obama.

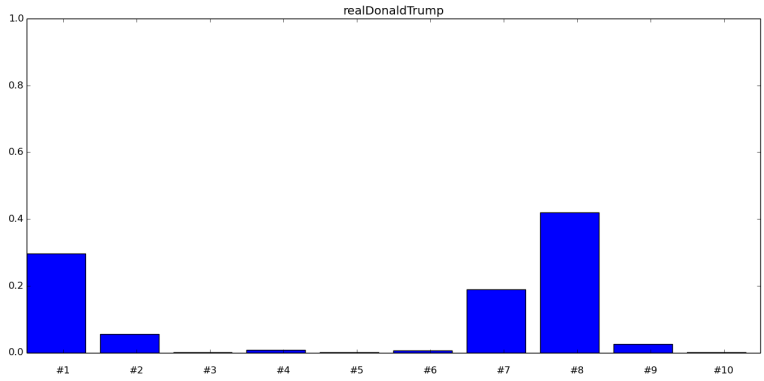


Figure 8.5.: AT1 topic distribution for Donald Trump.

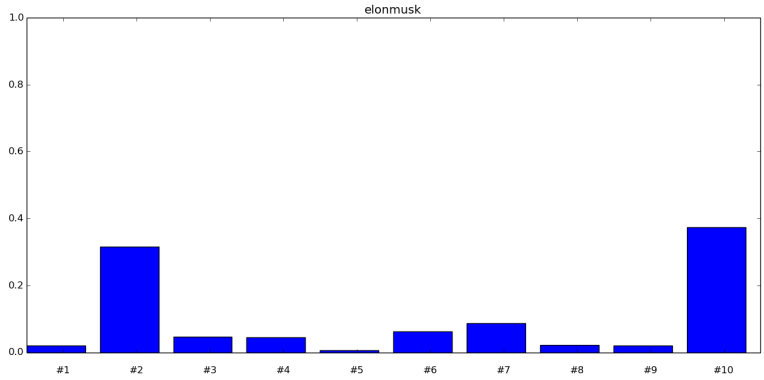


Figure 8.6.: AT1 topic distribution for Elon Musk.

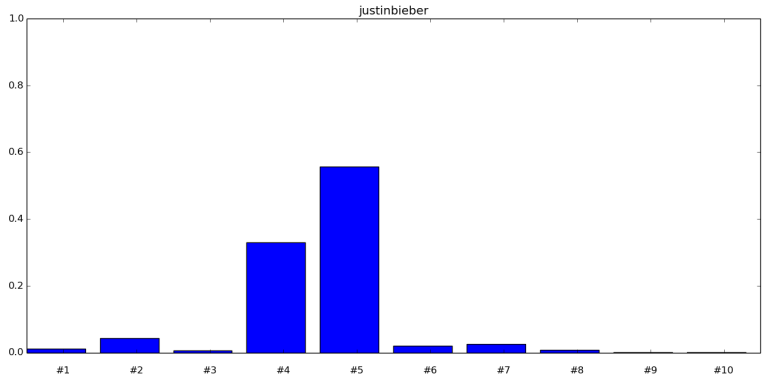


Figure 8.7.: AT1 topic distribution for Justin Bieber.

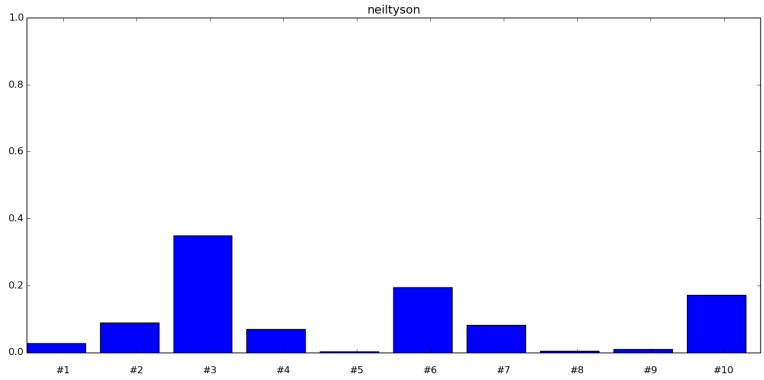


Figure 8.8.: AT1 topic distribution for Neil deGrasse Tyson.

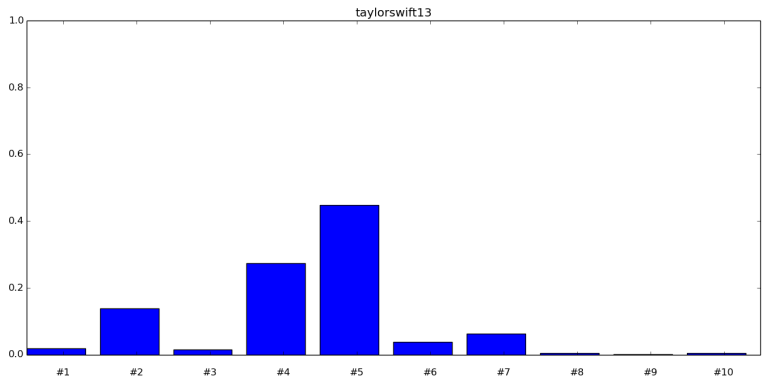


Figure 8.9.: AT1 topic distribution for Taylor Swift.

training messages generated by the same user.

2. Aggregate all testing messages generated by the same user into testing user profiles.

3. Take training messages, testing user profiles and testing messages as "new documents", use the trained model to infer a topic mixture for each of them.

A similar approach was chosen for conducting this experiment. We first train the model on a corpus where each document contains aggregated tweets for each user. Furthermore, new tweets for each author (which are not part of the training data) are downloaded, and the topic distribution is inferred for each of the new tweets. Finally, we calculate the topic distribution for each user as the *average* topic distribution over all new tweets written by that user. A higher number of original topics were used in this case, since a low number of topics produces too many topics containing the same popular words. One example is the entity *President Obama*: Many of the tweets posted by Barack Obama's Twitter account are posted by his Organizing for Action[2] staff. This leads to many tweets containing quotes by Barack Obama, being signed "— President Obama", which makes *President Obama* appear in an anomalous amount of topics. We therefore use 50 topics instead of the 10 topics used in the Rosen-Zvi et al. [2004] experiments, so that less occurring themes and topics don't get drowned in these popular terms. We want the original topics to cover a broad range of themes and topics, the idea being that we will subsequently filter out topics not relevant for that particular author when we infer the topic mixture based on *new* tweets by that author. This also means that the resulting amount of topics is not necessarily the same for all authors. A maximum of 10 topics were used for the resulting topic mixture inferred from the new tweets.

The resulting topic mixtures for the authors can be seen in Figures 8.10 through 8.15, and the most probable topics for each of the authors are tabulated in Table 8.5. As opposed to the topic mixtures in AT1, the topic mixtures for AT2 generally had one topic that was much more probable than the remaining topics. Therefore, the diversity in the language by each author might not be captured as well by AT2 as AT1. On the other hand, the most probable topic for each author generally describes the author with a higher precision than AT1. Using 50 topics instead of 10 allows the topics to be more specific. It is therefore easier to distinguish Justin Bieber from Taylor Swift than it was in the previous experiment; Justin

[2] https://www.barackobama.com/about-ofa/

Bieber's top topic includes words that are specific to him: *whatdoyoumean* is a hashtag version of a recent song he made, *"What do you mean"*, and the word *purpose* is one of his albums. Barack Obama, Elon Musk, Neil deGrasse Tyson and Donald Trump should also be easily recognizable from the topics, and their most probable words for AT2 resembles their most probable words for AT1. This Author-topic model accurately depicts the authors' interests.

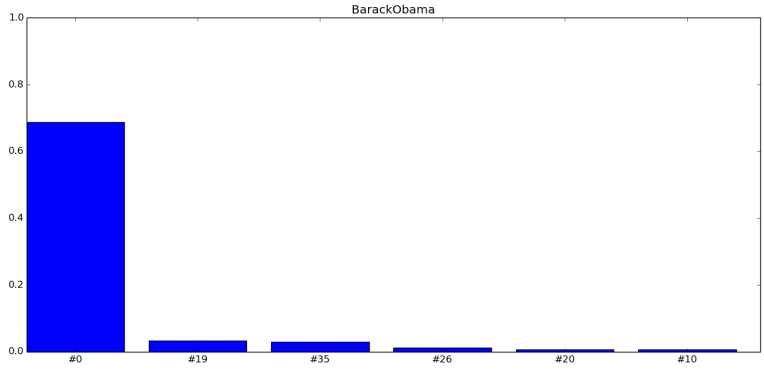


Figure 8.10.: AT2 topic distribution for Barack Obama.

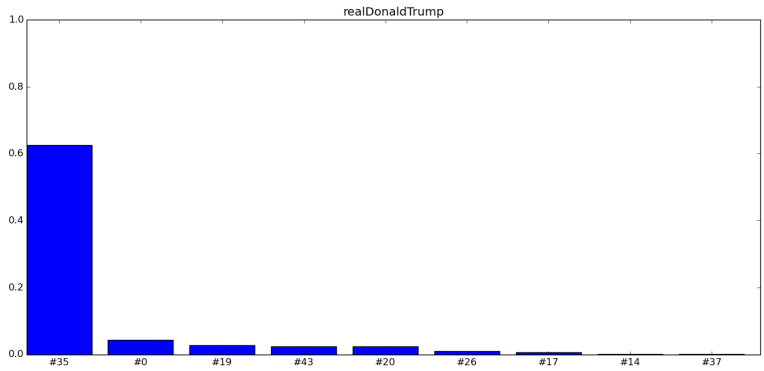


Figure 8.11.: AT2 topic distribution for Donald Trump.

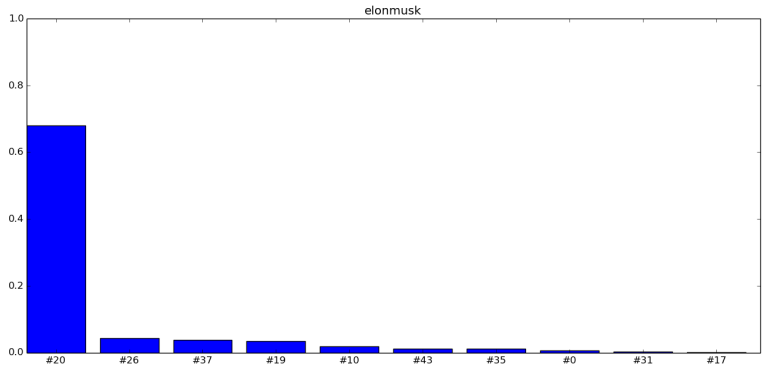


Figure 8.12.: AT2 topic distribution for Elon Musk.

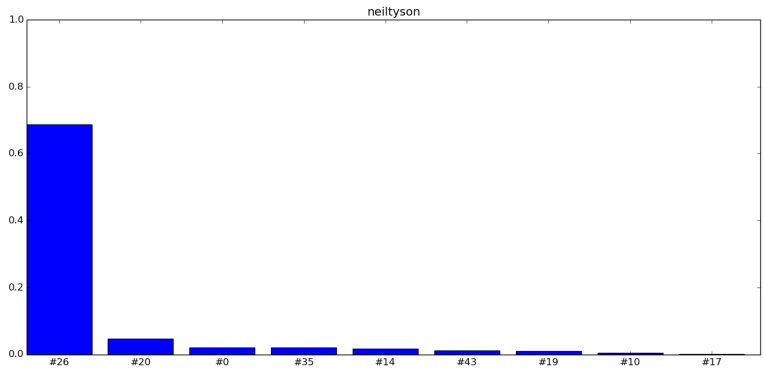


Figure 8.13.: AT2 topic distribution for Neil deGrasse Tyson.

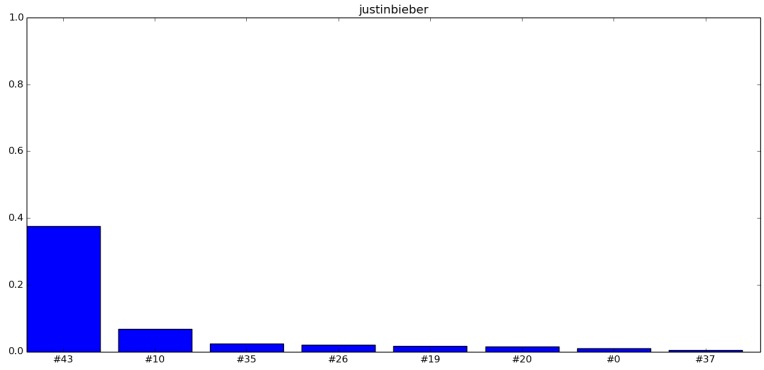


Figure 8.14.: AT2 topic distribution for Justin Bieber.

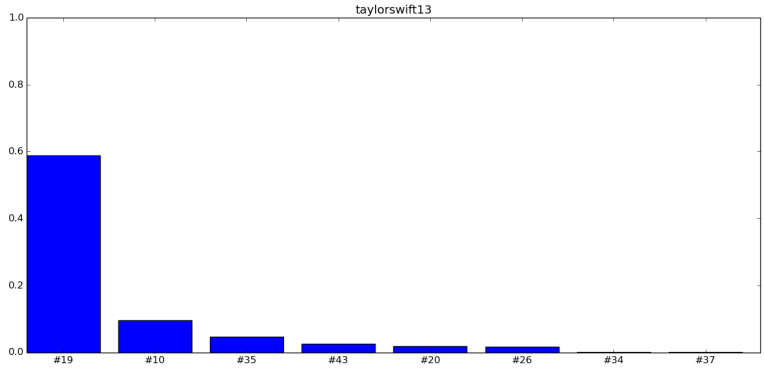


Figure 8.15.: AT2 topic distribution for Taylor Swift.

#0(*BarackObama*)	#20(*ElonMusk*)	#26(*NeildeGrasseTyson*)	#35(*DonaldTrump*)	#43(*JustinBieber*)	#19(*TaylorSwift*)
president	tesla	earth	will	thanks	tonight
obama	will	moon	great	love	ts1989
america	rocket	just	thank	whatdoyoumean	taylurking
sotu	just	day	trump2016	mean	just
actonclimate	model	one	just	purpose	love
time	launch	time	cruz	thank	thank
work	good	sun	hillary	lol	crowd
economy	dragon	people	new	good	night
americans	falcon	space	people	great	now
change	now	will	makeamericagreatagain	see	show

Table 8.5.: The most probable topic for each of the 6 authors from the topic distribution inferred from AT2.

9. Sentiment Analysis Experiments

In this chapter, we discuss the experiments conducted on the sentiment classifiers, as well as review the results achieved from these tests. It is worth noting that in the discussion on finding optimal values we are generally talking about finding sub-optimal values. It would be impractical to find the optimal solution through a single exhaustive grid search.

9.1. Training the classifiers and grid search

Seven different machine learning classifiers were created during our experiments. Two Näive Bayes (NB) based classifiers, the Bernoulli NB and the Multinomial NB classifier. Several Support Vector Machines (SVM) based classifiers, one with a sigmoid kernel, a linear kernel, a radial basis function (RBF) kernel and a Stochastic Gradient Descent (SGD) classifier on SVM. Lastly, a Maximum Entropy classifier (also known as logistic regression) was also built.

The outline of the experiments for each classifier was similar. First, the SemEval 2015 data set was loaded and each document preprocessed. This was the chosen data set to work with as this thesis was inspired by the SemEval 2015 TSA task (Twitter Sentiment Analysis, task 10, subtask B). The training data and test data was extracted from the SemEval dataset using stratification and a deterministic pseudo-random shuffling of the labels, to ensure the sets are equally balanced on classes in the partitioning. The FeatureUnion was then defined, Pipelined with the classifier and passed on for a grid search.

The baseline classifiers are defined as the version of each classifier only trained on the word n-gram feature, and the score for each baseline classifier can be found in Appendix A.

9.1.1. Grid Search

A coarse grid search was performed in an effort of speeding up the process of locating sub-optimal parameters for a classifier. A custom parameter

space to be searched was defined for each classifier. All classifiers include the features defined in Section 7.1.2, except the Multinomial NB which is unable to handle the lexicon feature because of negative values. The typical parameter space with their respective value ranges can be found in Figure 9.1. The coarse grid search is necessary because the amount of unique combinations of parameter candidates is typically around 10 million, that are to be tested on a k-fold value of 10 times each. This would take more than 30,000 years with the average time of testing each candidate on the NTNU Computer Science Department's server "translate", even though it is more powerful than your average desktop computer.[1] The k-fold value specifies the number of tests that are to be done on the same candidate set, and is commonly chosen around 10 to ensure good average cross validation scores.

To avoid using this impractical amount of time, the grid search is split into several steps. First, we find temporary values for the classifier specific parameters that are used for searching the other features' parameters. For the MaxEnt classifier, this would be finding the C value and penalty while using the other features' default values. Second, we deactivate all features except the TFIDF word n-gram feature and optimize its attributes one, two, or three at a time. Next, we optimize all the attributes of the TFIDF character n-gram feature up to three at a time. For the last step of the coarse grid search, we search the parameter space of the final features which is the Negation Count, Lexicon Score and the Emoticon feature. An example of the results from the iterative stages of the grid search can be seen in Figure 9.2. The rest of the classifiers' full grid search results can be found in Appendix A.

After the coarse grid search, we estimated our initial parameter values to be close to their local optima. Therefore a finer grid search was conducted, where some parameters were reconsidered. The n-gram ranges of the word and character feature were tested for their adjacent candidates, while some of the classifier specific parameters such as the C value were tested for values close to the current estimation. More often than not, the coarse grid search estimations held for the fine grid search as well.

9.2. Results

The most successful classifier by far was the Maximum Entropy classifier with an F1-score of 0.722, improved from its baseline score of 0.694. Also among the top classifiers, we have the SGD, Bernoulli NB and Sigmoid

[1] 'translate' is a server equipped with 125 GB internal memory, running four AMD Opteron 6128 processors.

Parameter space			
	Words Feature	N-grams range	(1,1), (1,2), (1,3), (1,4), (1,5)
		Sublinear TF	True, False
		Smooth IDF	True, False
		Max DF	0.1 → 1, increments +0.1
		Negation	True, False
	Character Feature	N-grams range	(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)
		Sublinear TF	True, False
		Smooth IDF	True, False
		Min DF, Max DF	{0.1 → 1} , increments +0.1
		Negation (T,F)	True, False
	Negation Count Feature	Normalization	True, False
	Emoticon Feature	Normalization (T,F)	True, False
	Lexicon Feature	Normalization (T,F)	True, False
	Classifier specific parameters	Penalty / Kernel	E.g. 'linear' or 'elasticnet'
		C value	(1 → 100)
		γ value	(10^{-6} → 1)
		α value	(10^{-6} → 100)

Figure 9.1.: The typical parameter space of our machine learning classifiers.

Top classifiers

	Bernoulli NB	**MaxEnt**	**SVM:Sigmoid**	**SGD**
Precision	0.662	0.723	0.649	**0.725**
Recall	0.659	**0.729**	0.644	0.722
F1-score	0.659	**0.722**	0.645	0.703

Table 9.1.: Top (four out of seven) scoring classifiers' average scores

Max Entropy			
Word feature		**Searched space**	**Result**
	ngram range	[(1,1),(1,2),(1,3),(1,4),(1,5)]	**(1,4)**
	sublinear tf	True, False	**FALSE**
	use idf	True, False	**FALSE**
	smooth idf	True, False	**TRUE**
	min df	0	**0**
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]	**0.8**
	negate	True, False	**TRUE**
CV Score	**0.645**	**Avg F1_score**	**0.694**

Character feature		**Searched space**	**Result**
	ngram range	[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]	**(2,4)**
	sublinear tf	True, False	**FALSE**
	use idf	True, False	**FALSE**
	smooth idf	True, False	**TRUE**
	min df	0	**0**
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]	**0.8**
CV Score	**0.692**	**Avg F1_score**	**0.694**

All / emoticon feature			
CV Score	**0.645**	**Avg F1_score**	**0.718**

All / neg. count feature			
CV Score	**0.659**	**Avg F1_score**	**0.722**

All / lexicon feature			
CV Score	**0.651**	**Avg F1_score**	**0.706**

All features		**Searched space**	**Result**
	C	[0.01. 0.05, 0.1, 0.5, 1, 10, 100	**0.05**
	penalty	['L1', 'L2']	**L1**
	Negcount (norm)	True, False	**TRUE**
	Emoticon (norm)	True, False	**FALSE**
	Lexicon (norm)	True, False	**FALSE**
CV Score	**0.642**	**Avg F1_score**	**0.714**

Combined (word + character)			
CV Score	**0.642**	**Avg F1_score**	**0.714**

Figure 9.2.: The iterative results from the grid search on the MaxEnt classifier

SVM classifiers as seen in Table 9.1. The SemEval Workshop of 2015's Task 10, subtask B was finding positive, neutral or negative polarity in a Twitter message, where scoring was based on the average of the F1-scores of the positive and negative tweets, excluding the neutral. The same scoring of our system would be 61.9 which would place us at an estimated eleventh place of 40 competing teams, had we taken part in the workshop. This estimation is based on the scores when run on a local computer several times, and not by the SemEval workshop — which means that the outcome might be different.

$$SemEvalScore = (F1_{\text{pos}} + F1_{\text{neg}})/2 \tag{9.1}$$

MaxEnt classifier

The grid search on the MaxEnt classifier established the best parameters for sentiment classification. MaxEnt performed best using all word n-grams up to 4-grams, while character n-grams included n-grams from 2- to 4-grams. Compared to other classifiers, MaxEnt is affected more than the others on the inclusion of the emoticon, negation count and lexicon features. As seen in Figure 9.2 the best results were from the tests without the negation count feature, which was probably interpreted as noise by the classifier. We see that the classifier performs a lot worse without the lexicon feature, which suggests the sentiment score of words have a noteworthy impact on this model. Finally, we see that the best classifier specific parameters for MaxEnt was a C-value of 0.05 and the penalty function "L1".

The confusion matrices for MaxEnt are visualized in Figure 9.3 and 9.4, where an optimal result would be heavy blue coloring only on the diagonal from top left to bottom right. This would imply that all the documents have the correct prediction. Coloring in any other grid indicates wrongly predicted documents. We see a clear tendency in both figures that neutral and positive documents are generally correctly predicted, while the negative documents tend to be predicted as either neutral or negative. These observations correlate well with the scores in Table 9.2. The extreme cases of negative documents predicted as positive, or opposite case, fortunately seem nonexistent.

MaxEntropy			
Label	**Precision**	**Recall**	**F1-Score**
positive	0.764	0.749	0.756
neutral	0.728	0.811	0.767
negative	0.599	0.403	0.482
Average	0.723	0.729	0.722

Table 9.2.: MaxEntropy scores on the test set. The MaxEnt classifier had the most highest scores in any category, including average F1-score, highest positive recall and highest positive F1-score.

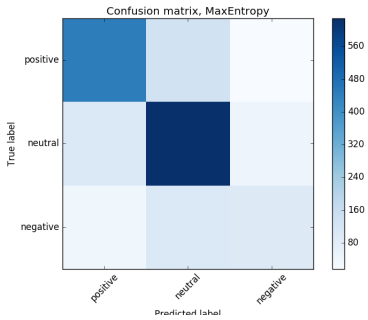


Figure 9.3.: Confusion Matrix for MaxEntropy

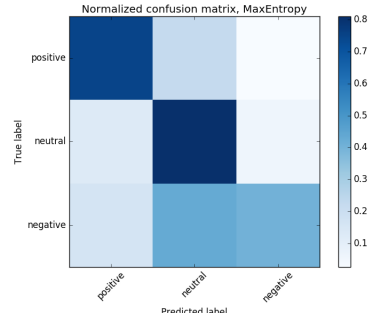


Figure 9.4.: Normalized Confusion Matrix for MaxEntropy

SGD classifier

Also quite successful was the Stochastic Gradient Descent classifier. It achieved optimal results with word n-grams up to trigrams, and character n-grams up to 6-grams. For the classifier specific parameters, the *elasticnet* penalty function received the highest scores combined with a *alpha* value of 0.1 and a *hinge* loss. The hinge loss function basically makes it a linear SVM classifier. The SGD classifier was one of the two classifiers (out of seven total) that undoubtedly benefitted most from including all the features. Looking at the SGD confusion matrix in Figure 9.5 and 9.6 we see good performance among the positive and neutral documents. However, the confusion matrix seems generally biased toward the neutral label. The middle column representing neutral predictions have strong blue coloring for all of the true labels. Table 9.3 shows a very impressive recall (0.890) on neutral labels, but very low recall on negative labels. On the other hand, it is worth noting that the Sigmoid SVM scored the highest precision on positive labels among the classifiers (0.800).

SGD based on SVM			
Label	**Precision**	**Recall**	**F1-Score**
positive	0.800	0.683	0.737
neutral	0.688	0.890	0.776
negative	0.659	0.258	0.370
Average	0.725	0.722	0.703

Table 9.3.: SGD: SVM scores on the test set. The SGD classifier had the highest positive precision, negative precision, neutral recall, and neutral F1-score. However, also the lowest negative recall and F1-score.

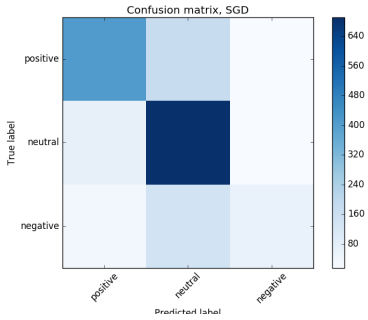


Figure 9.5.: Confusion Matrix for SGD SVM

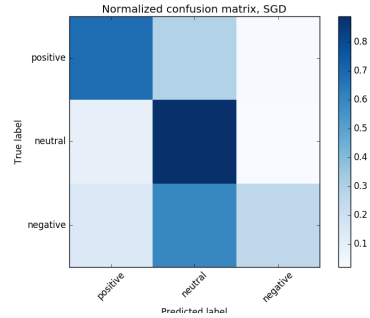


Figure 9.6.: Normalized Confusion Matrix for SGD SVM

Bernoulli classifier

The Bernoulli classifier is a simple Naïve Bayes classifier that achieved its best results on word unigrams and up to 4-gram character n-grams. Bernoulli was the other of the two classifiers that benefitted from the inclusion of all features, with an F1-score of 0.659. The best alpha value found through an extensive search was at 0.18, and normalization of all features. Looking at the Bernoulli confusion matrix in Figure 9.7 and 9.8 we see a strong diagonal line for the correctly predicted cases. The Bernoulli classifier generally resulted in minor supports in many categories, even the extreme case of negative labels being predicted as positive. However, Table 9.4 shows that the Bernoulli classifier, together with the Sigmoid SVM, had the best recall for negative documents (0.464).

Bernoulli Naïve Bayes			
Label	**Precision**	**Recall**	**F1-Score**
positive	0.638	0.719	0.676
neutral	0.732	0.673	0.701
negative	0.489	0.464	0.476
Average	0.662	0.659	0.659

Table 9.4.: Bernoulli Naïve Bayes scores on the test set. The Bernoulli had the highest scores in neutral precision, negative recall, and negative F1-score.

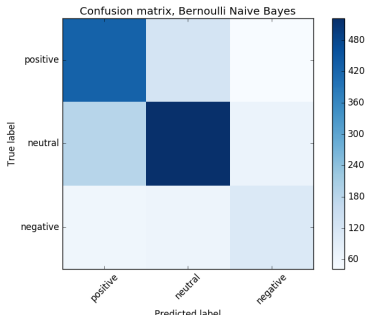


Figure 9.7.: Confusion Matrix for Bernoulli

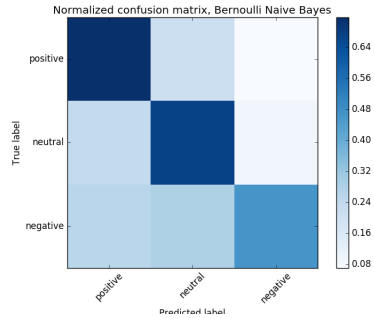


Figure 9.8.: Normalized Confusion Matrix for Bernoulli

Sigmoid SVM classifier

The Sigmoid SVM classifier scored the best when including all features, except the character n-grams, with an F1-score of 0.645. A correlation among the SVM classifiers (except linear) is that they are the only classifiers where the character n-grams feature seems to have a negative effect on performance. Most likely it is introduced as noise to the SVM classifier. Optimal scores were found with a C value of 1, and gamma of 0.001. The Sigmoid SVM scores seem quite similar to the Bernoulli classifier when comparing their confusion matrices. The normalized confusion matrix in Figure 9.10 shows a strong diagonal line of correctly predicted cases, but also considerable support in all other cases except positive cases predicted to be negative. Both the Bernoulli and Sigmoid SVM classifiers seem to act too generic, with significant support even in the extreme cases.

SVM: Sigmoid kernel			
Label	**Precision**	**Recall**	**F1-Score**
positive	0.660	0.739	0.697
neutral	0.706	0.626	0.664
negative	0.432	0.464	0.447
Average	0.649	0.644	0.645

Table 9.5.: SVM: Sigmoid scores on the test set. Shared the highest negative recall score with the Bernoulli classifier.

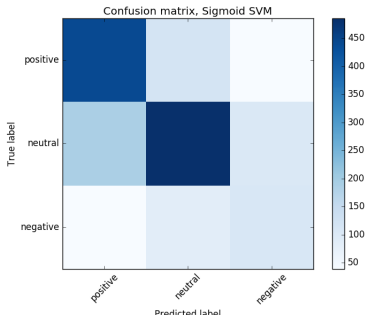


Figure 9.9.: Confusion Matrix for Sigmoid SVM

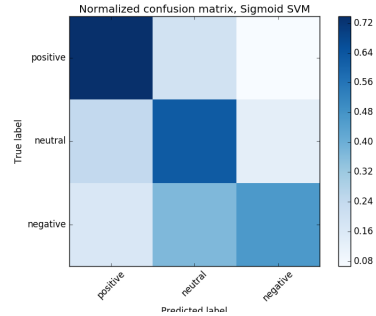


Figure 9.10.: Normalized Confusion Matrix for Sigmoid SVM

Scores on negative cases			
	Precision	**Recall**	**F1-Score**
Maxent	0.599	0.403	0.482
SGD	0.659	0.258	0.370
Bernoulli	0.489	0.464	0.476
Sigmoid	0.432	0.464	0.447
Average	0.545	0.397	0.444

Table 9.6.: Average scores on negative labels among the classifiers. Lowest scoring category in every classifier.

9.3. Putting it all together

The top four classifiers of the seven trained have been presented in this chapter. The grid search results from the other classifiers, as well as their confusion matrices, can be found in Appendix A and B. The greatest challenge for all the classifiers is in classifying negative documents correctly, as seen in Table 9.6. The training set contains around 8000 tweets, where 3000 tweets are positive, 1150 are negative and 3850 are neutral. This unequal partitioning of tweets' labels has an effect on the classifiers' behaviour in favoring neutral tweets, and to some degree positive, because of their greater share of the dataset. A quick look at the confusion matrices prediction of neutral and positive labels confirm this.

Results from experiments conducted on splitting the data set into new sets with different ratios of the three categories can be seen in Table 9.7. As expected, the recall of negative documents is much greater when the data set is evenly distributed on the three categories as seen in Table 9.8. However, these boosts in the negative category's score come at a cost of the two other as the overall F1-score observed in Table 9.7 is lower during these even distributions. On a few educated guesses, the custom distribution seen in these results have a significantly better performance than using the entire data set of 0.731. This shows the importance of distribution of categories in data sets, and the effect deviation away from the uniform distribution of categories has on the results of classification.

Table 9.9 displays the feature engineering tests done during the grid searches. We see that the overall highest score for the classifiers was not including all the features, but in fact excluding the negation count feature. The bold face scores show the respective classifiers' best F1-scores by feature combinations. It is worth noting that the distance between the three top scores (all features, exclusion of emoticons, and exclusion of negation

Results when limiting each category						
	500	**1000**	**1500**	**2000**	**Unlimited**	**Custom**
Precision	0.643	0.680	0.709	0.698	0.723	0.729
Recall	0.647	0.680	0.708	0.699	0.729	0.730
F1-score	0.642	0.678	0.708	0.697	0.722	0.731

Table 9.7.: Average scores when setting a maximum number of samples per category on the MaxEnt classifier. The iteration furthest to the right is on a custom setup of 1200 negative, 1500 positive and 2000 neutral.

Results on the negative category when limiting each category						
	500	**1000**	**1500**	**2000**	**Unlimited**	**Custom**
Precision	0.655	0.714	**0.724**	0.650	0.599	0.697
Recall	0.**780**	0.710	0.687	0.558	0.403	0.562
F1-Score	**0.712**	**0.712**	0.705	0.600	0.482	0.622

Table 9.8.: Scores for the negative category when setting a maximum number of samples per category on the MaxEnt classifier. The iteration furthest to the right is on a custom setup of 1200 negative, 1500 positive and 2000 neutral.

	Multinomial	Bernoulli	Linear	Maxent	SGD	Average
Baseline classifier	0.552	0.592	0.603	0.694	0.668	
All - emoticon feature	3.50E-02	6.40E-02	3.60E-02	2.40E-02	2.80E-02	3.74E-02
All - negation count	**3.90E-02**	6.60E-02	3.70E-02	**2.80E-02**	3.30E-02	**4.06E-02**
All - lexicon feature		6.20E-02	**3.80E-02**	1.20E-02	2.00E-02	3.30E-02
All - character feature	1.50E-02	5.30E-02	2.20E-02	8.00E-03	1.10E-02	2.18E-02
All classifiers	3.50E-02	**6.70E-02**	3.70E-02	2.00E-02	**3.50E-02**	3.88E-02

Table 9.9.: Comparison of the inclusion/exclusion of features. The notation "All - *feature*" should be read "All except *feature*". The baseline classifier scores are in F1-score, while the other scores are the gain in F1-score on that specific combination. The gain is generally highest when including all features except the negation score. The two empty slots in the first row and first column are not available.

MaxEnt results on different sizes of the dataset

	1000	**3000**	**5000**	**8000**
Precision	0.571	0.707	0.711	0.723
Recall	0.577	0.712	0.714	0.729
F1-score	0.563	0.705	0.705	0.722

Table 9.10.: Average scores on different sizes of the SemEval2015 dataset with the MaxEnt classifier. The largest dataset has the highest score on all metrics.

counts) are rather small. This suggests that the outcome was little affected by either exclusion or inclusion of the negation count and emoticon feature in the general case.

A training set of 8000 tweets is not considered a large training set in the case of machine learning in sentiment analysis. As these are short microblog documents, they require a bigger training set than performing sentiment analysis on documents that are longer (e.g. movie reviews or news articles) to achieve top results. Simple tests on the MaxEnt classifier, seen in Table 9.10, with different training set sizes prove that size of the corpora trained upon has a big impact on classifier performance. There is a great spike in performance improvement when increasing the dataset from a sample size of 1000 to 3000, however we still see significant improvement from 5000 to 8000. This leads us to believe that a dataset comprised of 8000 tweets is might be insufficient, and working with an even bigger datasets could improve the performance of our classifiers even further.

10. Discussion

In this chapter we discuss the fulfillment of the project goals listed in Section 1.3, and give some conclusive words on the experimental results and what remains as future work.

10.1. Evaluation of project goals

G1: Establish the state-of-the-art

We conducted an extensive structured literature review, following the guidelines put forward by Kofod-Petersen [2012]. The review presents highly relevant studies that have designed and constructed some of the most effetive systems on the topics, and goes into depth in the methods utilized. It also considers the research and work executed by the top scoring teams from the annual SemEval workshop of 2015. This state-of-the-art review has been the foundation on which we built our sentiment analysis, topic modeling and visualization systems.

G2: Create a topic modeling system for microblogs

A topic modeling system for modeling tweet corpora was successfully created. We managed to utilize pooling techniques to improve the coherence and interpretability of standard topic models, our results indicating that techniques such as author aggregating and hashtag aggregation generate more coherent topics. Moreover, we show that the author-topic model proposed by Rosen-Zvi et al. [2004], which is not based on pooling, depicts Twitter users and their tweets accurately through coherent topics.

G3: Create a sentiment analysis system for microblogs

A sentiment analysis system targeted for Twitter microblogs was created, constructed on the most common findings among the selected systems from the state-of-the-art review. From a total of seven classifiers trained through extensive feature engineering and grid searches, our experiments show that the Maximum Entropy classifier performed best in terms of F1-score, rivaling the best teams of the 2015 SemEval workshop.

G4: Create a visualization system

A novel visualization application for topic modeling and sentiment analysis in Twitter was created. The only directly similar system found was Tweet Viz from the North Carolina State University.[1] However, similar systems tend to use word counts and co-occurrences for modeling topics. Based on an extensive search, TweetMoods is the first visualization of its kind in combining sentiment analysis and topic modeling by use of LDA based methods.

10.2. Conclusions

In this section, we present our conclusions based on the results retrieved from the conducted experiments. In summary, our topic modeling results show that applying pooling techniques to tweets increases the topic coherence significantly. On performing message polarity classification on tweets, the Maximum Entropy classifier yielded results outperforming most earlier submitted work to the International Workshop on Semantic Evaluation of 2015. This proves the importance of our extensive grid searches on optimizing the parameter space of the classifiers.

10.2.1. Topic modeling

Various methods for estimating the optimal number of topics for tweets were tested. The Elbow method almost exclusively suggested a k value between 3 and 5, no matter how large or diverse the corpus was. Even though we managed to show that different types of corpora produce highly varying clusters, we could not accurately deduce an optimal number of topics from this. The stability score [Greene et al., 2014] also produced rather poor scores for all number of topics when applied to a tweet corpus. The sparsity of tweets is likely the cause of this; the documents do not contain enough words to produce sufficient term co-occurrences. Hong and Davison [2010] found that 50 topics produced the optimal results for their author-topic model, although the optimal number of topics is dependent on the diversity of the corpora. We therefore used k values of 10 and 50 in our experiments, using 50 for large corpora where we could expect a diverse collection of documents.

In Section 6.2, we discussed the use of hashtag co-occurrences to divulge latent networks of topics and events in a collection of tweets. A hashtag co-occurrence graph for Super Bowl 2016 helped reveal information about this event; artists playing during the half-time show, which companies had

[1] https://www.csc.ncsu.edu/faculty/healey/tweet_viz/tweet_app/

commercials during the match, etc. A hashtag co-occurrence network was not directly combined with topic modeling in our experiments; however, we performed a hashtag aggregating technique that utilizes hashtags in similar ways.

We show that aggregating tweets to mitigate the disadvantages of sparse texts do indeed have an impact on the coherence of a topic model. The hashtag aggregation technique is especially interesting here, as it utilizes a specific metadata tag that is not present in standard documents. A hashtag aggregated topic model produced a much better coherence than the standard LDA variation for the same corpus; this is also consistent with recent research on topic models for microblogs. Two Author-topic models were used in our experiments, one using the Rosen-Zvi et al. [2004] topic model (referred to as AT1 in this thesis) and the aggregated author-topic model proposed by Hong and Davison [2010] (referred to as AT2), both seeming to produce interpretable topics. It is worth noting that there is no standardized methods for evaluating topic models, as most quantitative ways try to estimate human judgement. Moreover, there is no precise definition of a *gold standard* for topic models, which makes the task of comparing and ranking topic models difficult. A combination of a computational method and human judgement was therefore used in our evaluations.

10.2.2. Sentiment analysis

A total of seven different sentiment classifiers were created and experimented on. These machine learning classifiers were based on the algorithms presented in Section 2.2, the Näive Bayes (NB), Support Vector Machines (SVM), Maximum Entropy (MaxEnt) and Stochastic Gradient Descent (SGD). Based on the experiments conducted in Chapter 9 we conclude that the MaxEnt classifier performs best on classifying tweets into the categories of positive, neutral or negative based on the scoring defined by the classification metrics in Section 2.2.1. The experiments show that the SVM classifiers tend to favor feature sets of smaller size, and performs well with less information. However, the SVM classifiers were the only cases whose performance was negatively affected when both word and character n-grams were combined in the feature set. This allows us to hypothesize that the SVM classifiers do not deal with the curse of dimensionality well, whereas the MaxEnt classifer's performance was significantly boosted with increased information and feature sets of high dimensionality. SVM and NB achieved greatest performance when choosing parameters that resulted in small feature sets, generally working with only word unigrams and excluding character n-grams as seen in Appendix A. We predict that more informative feature sets with new and extended features can further increase the

performance of MaxEnt in text classification, that clearly benefited from the data inflation.

An extensive grid search split into a coarse and a fine stage was performed, which was key to the successful results of the classifiers. Similar systems as discussed in Chapters 4 and 7 also use grid search as a way of improving the parameters which are specific to the machine learning algorithm (e.g. alpha-, gamma- and C-values), but generally do not include a substantial grid search on the feature specific parameters. Our results show that the classifier specific parameters are most effective in improving the performance of each classifier, but that the additional grid search on the feature parameters gave the added edge resulting in our higher performance. Consequently, we speculate that many of the similar projects have underestimated the importance of optimizing the parameter space of the added features.

As seen in Table 9.9, using sentiment lexica to build a document sentiment score feature proved to be the most effective non n-gram feature. However, the lexicon transformer was based on lexica that are somewhat outdated and that do not include sentiment scores for much of the informal language of Twitter. This is especially true for slang language that is constantly changing on the web. The lexica used in this system are from 2005 to 2013, and though they perform quite well, updated versions would be preferred.

We introduced a naïve negation scope detection in Section 7.1.2. Adding negation tags to the respective tokens increases the vocabulary of the system. As only a minority of tweets contain negations, we're consequently extending the vocabulary with less informative features than we would without the tags. Similarly, we're currently mapping emoji-smileys to one of nine ASCII-smileys. This affects the information value of these tokens in the n-gram features, which would be more accurately valued if they were mapped to a larger set of ASCII-smileys. However, it is difficult to predict whether this would have a positive or negative performance outcome.

10.2.3. TweetMoods

Our visualization application can effectively infer a topic distribution over one or more documents, and it gives a good overview of which topics people talk about, given that the topic model already contains data related to the tweets retrieved by the search. This is one limitation of using a pre-trained model to infer a distribution over new documents; the topic distribution cannot give information about themes and topics in the documents if it is not already contained in the model. We therefore included a dropdown menu so that users can choose between several pre-trained topic models.

Our experiments show that this method can provide an informative overview of the topics covered by a corpus of tweets.

The visualization application also provides a great overview of the sentiments contained in a corpus of tweets retrieved by a specific query. Users can search for specific products or politicians and get an overview of the sentiments conveyed toward these products or politicians. The pie-chart shows an overview of these sentiments, and the list of tweets shows the actual tweets with the assigned sentiments. This lets the users verify the accuracy of any assigned sentiments themselves. The application gives extensive information about the contents of tweets, and visualizes the combined sentiment analysis and topic modeling information in a way (to our knowledge) not seen before.

10.3. Future Work

As the system built for this project was generally split into a topic modeling and a sentiment analysis part, there were several aspects of either part that fell outside the scope of this thesis.

10.3.1. Topic Modeling

One way to extend the topic modeling system would be to apply online analysis by implementing automatic updates of a topic model, continuously extended by new tweets retrieved from the Twitter Streaming API. This would help in detect emerging events, in a fashion similar to Lau et al. [2012]. Moreover, Dynamic Topic Models, as introduced in Section 2.3.2, should be considered to provide better temporal modeling of Twitter data. A limitation to topic modeling in general is the difficulties in evaluating the accuracy of the models. Computational methods try to simulate human judgement, which poses difficulties, as human judgement is not clearly defined. Further research could help provide better methods for evaluating topic models.

In this thesis, we aggregated tweets sharing authors and hashtags. Further work should look into other pooling schemes, and see how they compare to author and hashtag aggregation. One example would be to aggregate conversations on Twitter into individual documents. Tweets contain a lot of metadata that can aid the aggregation of tweets into individual documents.

10.3.2. Sentiment Analysis

For the Twitter sentiment classifier, the next step is including more features in the system. Features that might be of interest include among others part-

of-speech tagging and word clustering as proposed by Owoputi et al. [2012]. Part-of-speech tagging was not a prioritized feature for this thesis based on the results by Kouloumpis et al. [2011] that part-of-speech tagging does not have significant impact on TSA. Word clustering as a feature was found to have minor positive results on classification performance by Reitan et al. [2015].

There are several sentiment lexica that include emotion scores, such as fear and joy. It would be interesting to see how appending a feature that incorporates emotions would affect the estimation of sentiment. One could also look into deeper learning of links, pictures and other attachments' contents as features to improve sentiment classification.

Further work on features includes extending and improving the current features. The lexicon feature can be extended by constructing a new, updated sentiment lexicon aimed at the current Twitter language. One could extend also include sentiment scores that are more specific than incrementing the positive/negative score by one for each word occurrence. The negation count feature might be improved if it was not based on a naïve negation tagging algorithm, but rather a more advanced negation scope detection system. The emoticon feature can be extended by mapping emojis to more appropriate ASCII emoticons than the three that are currently being used.

Currently, the same preprocessing methods are being applied to each feature of the sentiment classification system. Customizing a set of preprocessor methods that are specific for each feature might also yield interesting results.

Final remarks for the sentiment classification system entails running experiments on other datasets. Some tests were run on the SemEval 2016 datasets that had lower performance but promising potential. However, it fell outside the scope of this thesis to customize the classifiers and evaluate the results.

10.3.3. TweetMoods

One way to extend the sentiment classification part of the visualization system would be to let the users choose whether or not they feel that the assigned sentiment is the correct one. This would require incrementally training the classifier with the new document-sentiment pairs, and would improve the classifier over time, given that the users' choices are accurate.

On May 24th, 2016, Twitter announced that they will change the way replies and attachment work on Twitter.[2] Media attachments and usernames at the beginning of a reply will no longer count toward the 140

[2]https://blog.twitter.com/2016/doing-more-with-140-characters

character-limit, which allows tweets to become longer. It will be interesting to see if this will have an effect on information retrieval on Twitter in future research.

Bibliography

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment Analysis of Twitter Data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-96-1. URL `http://dl.acm.org/citation.cfm?id=2021109.2021114`.

David M. Blei. Probabilistic Topic Models. In *Communications of Association for Computer Machinery*, volume 55, New York, NY, USA, April 2012. ACM.

David M. Blei and John D. Lafferty. Dynamic Topic Models. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 113–120, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143859. URL `http://doi.acm.org/10.1145/1143844.1143859`.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. In *the Journal of Machine Learning Research*, volume 3, pages 993–1022, MIT, Massachusetts, USA, 2003. JMLR. org.

William Boag, Peter Potash, and Anna Rumshisky. TwitterHawk: A Feature Bucket Based Approach to Sentiment Analysis. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 640–646, Denver, Colorado, June 2015. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/S15-2107`.

Fazli Can and Esen A. Ozkarahan. Concepts and Effectiveness of the Cover-coefficient-based Clustering Methodology for Text Databases. In *ACM Transitional Database Systems*, volume 15, pages 483–517, New York, NY, USA, December 1990. Association for Computer Machinery. doi: 10.1145/99935.99938. URL `http://doi.acm.org/10.1145/99935.99938`.

Jonathan Chang, Sean Gerrish, Chong Wang, Jordan L Boyd-Graber, and David M Blei. Reading tea leaves: How humans interpret topic models.

In *Advances in neural information processing systems*, pages 288–296, Vancouver, British Columbia, 2009.

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. In *Machine Learning*, volume 20, pages 273–297, Hingham, MA, USA, September 1995. Kluwer Academic Publishers. doi: 10.1023/A:1022627411411. URL `http://dx.doi.org/10.1023/A:1022627411411`.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. Part-of-speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL `http://dl.acm.org/citation.cfm?id=2002736.2002747`.

Derek Greene, Derek O'Callaghan, editor="Calders Toon Cunningham, Pádraig", Floriana Esposito, Eyke Hüllermeier, and Rosa Meo. *How Many Topics? Stability Analysis for Topic Models*, pages 498–513. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. ISBN 978-3-662-44848-9.

Matthew Hoffman, Francis R. Bach, and David M. Blei. Online Learning for Latent Dirichlet Allocation. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 856–864, Vancouver, Canada, 2010. Curran Associates, Inc.

Liangjie Hong and Brian D. Davison. Empirical Study of Topic Modeling in Twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0217-3. doi: 10.1145/1964858.1964870. URL `http://doi.acm.org/10.1145/1964858.1964870`.

Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent Twitter Sentiment Classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 151–160, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL `http://dl.acm.org/citation.cfm?id=2002472.2002492`.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. Sentiment Analysis of Short Informal Texts. In *J. Artif. Int. Res.*, volume 50, pages 723–762, USA, May 2014. AI Access Foundation.

Anders Kofod-Petersen. How to do a Structured Literature Review in Computer Science, 2012.

Olessia Koltsova and Sergei Koltcov. Mapping the public agenda with topic modeling: The case of the Russian LiveJournal. In *Policy & Internet*, volume 5, pages 207–227, Russia, 2013. doi: 10.1002/1944-2866.POI331. URL `http://dx.doi.org/10.1002/1944-2866.POI331`.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter Sentiment Analysis: The Good the Bad and the OMG! In *International Conference on Web and Social Media*, volume 11, pages 538–541, San Francisco, USA, 2011.

Jey Han Lau, Nigel Collier, and Timothy Baldwin. On-line Trend Analysis with Topic Models:\# Twitter Trends Detection Topic Model Online. In *Proceedings of COLING 2012: Technical Papers, pages 1519–1534*, pages 1519–1534, Mumbai, India, 2012.

Bing Liu. Sentiment Analysis and Subjectivity. In *Handbook of natural language processing*, volume 2, pages 627–666, Chicago, USA, 2010.

Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-13360-1.

Wes McKinney. Pandas, Python Data Analysis Library. 2015. *Reference Source*, 2014.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Sujian Li, and Houfeng Wang. Entity-centric Topic-oriented Opinion Summarization in Twitter. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 379–387, New York, NY, USA, 2012. Association for Computational Linguistics. ISBN 978-1-4503-1462-6. doi: 10.1145/2339530.2339592. URL `http://doi.acm.org/10.1145/2339530.2339592`.

David Mimno, Hanna M. Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 262–272, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4.

David Nadeau and Satoshi Sekine. A survey of Named Entity Recognition and Classification. In *Lingvisticae Investigationes*, volume 30, pages 3–26, Barcelona, Spain, 2007. John Benjamins publishing company.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *International Conference on Web and Social Media*, volume 11, pages 1–2, Washington DC, USA, 2010.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, and Nathan Schneider. Part-of-speech tagging for Twitter: Word clusters and other advances. In *School of Computer Science, Carnegie Mellon University, Tech. Rep*, Pennsylvania, USA, 2012.

Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *Language Resources and Evaluation Conference*, volume 10, pages 1320–1326, Orsay Cedex, France, 2010. Universite de Paris-Sud.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. In *The Journal of Machine Learning Research*, volume 12, pages 2825–2830. JMLR.org, 2011.

Nataliia Plotnikova, Micha Kohl, Kevin Volkert, Andreas Lerner, Natalie Dykes, Heiko Emer, and Stefan Evert. KLUEless: Polarity Classification and Association. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Erlangen, Germany, 2015. Friedrich-Alexander-Universitat Erlangen-Nurnberg.

Xiaojun Quan, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. Short and sparse text topic modeling via self-aggregation. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 2270–2276. AAAI Press, 2015. ISBN 978-1-57735-738-4. URL `http://dl.acm.org/citation.cfm?id=2832415.2832564`.

Juan Ramos. Using TF-IDF to Determine Word Relevance in Document Queries. In *Proceedings of the first instructional conference on machine learning*, New Jersey, USA, 2003.

Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the Language Resources and Evaluation Conference 2010 Workshop on New Challenges for Natural Lan-*

guage Processing Frameworks, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. Negation scope detection for twitter sentiment analysis. In *6TH WORKSHOP ON COMPUTATIONAL APPROACHES TO SUBJECTIVITY, SENTIMENT AND SOCIAL MEDIA ANALYSIS WASSA 2015*, page 99, Lisboa, Portugal, 2015. Norwegian University of Science and Technology (NTNU).

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. Named Entity Recognition in Tweets: an Experimental Study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-937284-11-4. URL http://dl.acm.org/citation.cfm?id=2145432.2145595.

Michal Rosen-Zvi, Thomas Griffiths, Mark Steyvers, and Padhraic Smyth. The Author-topic Model for Authors and Documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494, Arlington, Virginia, United States, 2004. AUAI Press. ISBN 0-9749039-0-6. URL http://dl.acm.org/citation.cfm?id=1036843.1036902.

Helmut Schmid. Probabilistic Part-Of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on new methods in Language Processing*, volume 12, pages 44–49, Stuttgart, Germany, 1994. Citeseer.

Jianfeng Si, Arjun Mukherjee, Bing Liu, Qing Li, Huayi Li, and Xiaotie Deng. Exploiting Topic based Twitter Sentiment for Stock Prediction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 24–29, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

Dionisios N Sotiropoulos, Chris D Kounavis, Panos Kourouthanassis, and George M Giaglis. What drives social sentiment? An entropic measure-based clustering approach towards identifying factors that influence social sentiment polarity. In *Information, Intelligence, Systems and Applications, IISA 2014, The 5th International Conference*, pages 361–373, Chania Crete, Greece, 2014. IEEE.

Karen Sparck Jones. Document Retrieval Systems. In *A Statistical Interpretation of Term Specificity and Its Application in Retrieval*, pages 132–

142, London, UK, UK, 1988. Taylor Graham Publishing. ISBN 0-947568-21-2. URL http://dl.acm.org/citation.cfm?id=106765.106782.

Pranav Waila, VK Singh, and Manish K Singh. Blog text analysis using topic modeling, named entity recognition and sentiment classifier combine. In *Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on*, pages 1166–1171, Mysore, India, 2013. IEEE.

Chong Wang, David Blei, and David Heckerman. Continuous time dynamic topic models. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, volume abs/1206.3298, Helsinki, Finland, 2012. URL http://arxiv.org/abs/1206.3298.

Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in Twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th Association for Computational Linguistics International Conference on Information and Knowledge Management*, CIKM '11, pages 1031–1040, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063726. URL http://doi.acm.org/10.1145/2063576.2063726.

Y. Wang, J. Liu, J. Qu, Y. Huang, J. Chen, and X. Feng. Hashtag Graph Based Topic Model for Tweet Mining. In *2014 IEEE International Conference on Data Mining*, pages 1025–1030, Shenzhen, China, Dec 2014. doi: 10.1109/ICDM.2014.60.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. TwitterRank: Finding Topic-sensitive Influential Twitterers. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-889-6. doi: 10.1145/1718487.1718520. URL http://doi.acm.org/10.1145/1718487.1718520.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354. Association for Computational Linguistics, 2005.

Yiming Yang and Jan O. Pedersen. A Comparative Study on Feature Selection in Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*,

ICML '97, pages 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-486-3. URL `http://dl.acm.org/citation.cfm?id=645526.657137`.

Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: A statistical framework. In *International Journal of Machine Learning and Cybernetics*, volume 1, pages 43–52, Nanjing, China, 2010. Nanjing University, Springer.

Zhihua Zhang, Guoshun Wu, and Man Lan. East China Normal University, ECNU: Multi-level Sentiment Analysis on Twitter Using Traditional Linguistic Features and Word Embedding Features. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, Shanghai, China, 2015. East China Normal University Shanghai.

A. Tables of each classifier's grid search

Below you will find the full results from the most significant iterations of the extensive grid search (as described in Section 9) done for every classifier, except the MaxEnt presented in Figure 9.2.

Bernoulli Naive Bayes parameters				
Tfidf transformer		**Searched space**		**Result**
	ngram range	[(1,1),(1,2),(1,3),(1,4),(1,5)]		(1,1)
	sublinear tf	[True, False]		TRUE
	use idf	True, False		TRUE
	smooth idf	True, False		TRUE
	min df		0	0
	max df	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9		0.5
	negate	True, False		FALSE
Score	**0.46**	**Avg F1_score**		0.592

Tfidf Char transformer		**Searched space**		**Result**
	ngram range	[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]		(1,4)
	sublinear tf	True, False		FALSE
	use idf	True, False		TRUE
	smooth idf	True, False		FALSE
	min df		0	0
	max df	[0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.4
Score	**0.583**	**Avg F1_score**		0.643

All but Emoticons			
Score	**0.61**	**Avg F1_score**	0.656

All but NegCount			
Score	**0.605**	**Avg F1_score**	0.658

All but Lexicon			
Score	**0.609**	**Avg F1_score**	0.654

Combined (all)		**Searched space**	**Result**
	clf_alpha	[100, 50, 25, 10, 5, 2, 1, 0.1, 0.01, 0.001, 0.0001, 0]	**0.18**
	Lexicon(norm)	True, False	TRUE
	Negcount(norm)	True, False	TRUE
	Emoticon(Norm)	True, False	TRUE
Score	**0.61**	**Avg F1_score**	**0.659**

Combined (tfidf + char)			
Score	**0.603**	**Avg F1_score**	0.65

Figure A.1.: Grid search results for the Bernoulli NB classifier

SVM: Linear				
Tfidf transformer		Searched space		**Result**
	ngram range	[(1,1),(1,2),(1,3),(1,4),(1,5)]		(1,1)
	sublinear tf	True, False		TRUE
	use idf	True, False		TRUE
	smooth idf	True, False		FALSE
	min df		0	0
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.5
	negate	True, False		TRUE
Score	**0.537**	**Avg F1_score**		**0.603**

Tfidf Char transformer		Searched space		**Result**
	ngram range	[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]		(1,4)
	sublinear tf	True, False		TRUE
	use idf	True, False		TRUE
	smooth idf	True, False		FALSE
	min df		0	0
	max df	[0.1,0.2,0.3,0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.9
Score		**Avg F1_score**		**0.619**

All but emoticons			
Score	**0.572**	**Avg F1_score**	**0.639**

All but negcount			
Score	**0.573**	**Avg F1_score**	**0.64**

All but lexicon			
Score	**0.579**	**Avg F1_score**	**0.641**

Combined (all)		**Searched space**	**Result**
	C	[1,10,100]	**1**
	Gamma	[0.1, 0.01, 0.001, 0.0001]	**0.1**
	Negcount (norm)	True, False	**TRUE**
	Emoticon (norm)	True, False	**TRUE**
	Lexicon (norm)	True, False	**TRUE**
Score	**0.569**	**Avg F1_score**	**0.64**

Combined (only tfidf + char)			
Score	**0.571**	**Avg F1_score**	**0.638**

Combined (neg, lex, emoticons)			
Score	**0.541**	**Avg F1_score**	**0.589**

Figure A.2.: Grid search results for the Linear SVM classifier

Multinomial Naive Bayes parameters			
Tfidf transformer		**Searched space**	**Result**
	ngram range	(1,1),(1,2),(1,3),(1,4),(1,5)	(1,3)
	sublinear tf	True, False	FALSE
	use idf	True, False	FALSE
	smooth idf	True, False	TRUE
	min df	0	0
	max df	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	0.2
	negate	True, False	TRUE
Score	**0.51**	**Avg F1_score**	**0.552**

Tfidf Char transformer		**Searched space**	**Result**
	ngram range	(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)	(1,6)
	sublinear tf	True, False	TRUE
	use idf	True, False	FALSE
	smooth idf	True, False	TRUE
	min df	0	0
	max df	0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1	0.7
Score	**0.503**	**Avg F1_score**	**0.564**

All but emoticons			
Score	**0.546**	**Avg F1_score**	**0.587**

All but negcount			
Score	**0.544**	**Avg F1_score**	**0.591**

Combined (all)			
	clf_alpha	[100, 50, 25, 10, 5, 2, 1, 0.1, 0.01, 0.001, 0.0001, 0]	**25**
	negcount (norm)	True, False	TRUE
	emoticons(norm)	True, False	TRUE
Score	**0.546**	**Avg F1_score**	**0.587**

Combined (only tfidf + char)			
Score	**0.533**	**Avg F1_score**	**0.591**

Figure A.3.: Grid search results for the Multinomial NB classifier

SVM: poly				
Word feature		**Searched space**		**Result**
	ngram range	[(1,1),(1,2),(1,3),(1,4),(1,5)]		1,1
	sublinear tf	True, False		TRUE
	use idf	True, False		TRUE
	smooth idf	True, False		TRUE
	min df		0	0
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.4
	negate	True, False		TRUE
Score	**0.318**	**Avg F1_score**		**0.49**

Character feature		**Searched space**		**Result**
	ngram range	[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]		1,4
	sublinear tf	True, False		FALSE
	use idf	True, False		FALSE
	smooth idf	True, False		TRUE
	min df		0	0
	max df	[0.1,0.2,0.3,0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.3
Score	**0.278**	**Avg F1_score**		**0.405**

All / emoticon feature			
Score	**0.367**	**Avg F1_score**	**0.493**

All / neg. count feature			
Score	**0.351**	**Avg F1_score**	**0.493**

All / lexicon feature			
Score	**0.361**	**Avg F1_score**	**0.492**

All features		**Searched space**	**Result**
	C	[1,10,100]	1
	Gamma	[0.1, 0.01, 0.001, 0.0001]	0.01
	Negcount (norm)	True, False	**TRUE**
	Emoticon (norm)	True, False	TRUE
	Lexicon (norm)	True, False	TRUE
Score	**0.268**	**Avg F1_score**	**0.396**

Combined (word + character)			
Score	**0.294**	**Avg F1_score**	**0.396**

All but tfidf-character			
Score	**0.36**	**Avg F1_score**	**0.493**

Figure A.4.: Grid search results for the Poly SVM classifier

SVM: rbf				
Word feature		**Searched space**		**Result**
	ngram range	[(1,1),(1,2),(1,3),(1,4),(1,5)]		(1,1)
	sublinear tf	True, False		FALSE
	use idf	True, False		FALSE
	smooth idf	True, False		TRUE
	min df		0	0
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.5
	negate	True, False		TRUE
CV Score	**0.421**	**Avg F1_score**		**0.57**

Character feature		**Searched space**		**Result**
	ngram range	[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]		(1,4)
	sublinear tf	True, False		FALSE
	use idf	True, False		FALSE
	smooth idf	True, False		TRUE
	min df		0	0
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]		0.6
CV Score	**0.243**	**Avg F1_score**		**0.364**

All / emoticon feature			
CV Score	**0.432**	**Avg F1_score**	**0.578**

All / neg. count feature			
CV Score	**0.436**	**Avg F1_score**	**0.575**

All / lexicon feature			
CV Score	**0.42**	**Avg F1_score**	**0.575**

All features		**Searched space**	**Result**
	C	[1,10,100]	1
	Gamma	[0.1, 0.01, 0.001, 0.0001]	0.0001
	Negcount (norm)	True, False	TRUE
	Emoticon (norm)	True, False	TRUE
	Lexicon (norm)	True, False	TRUE
CV Score	**0.255**	**Avg F1_score**	**0.364**

Combined (word + character)			
CV Score	**0.255**	**Avg F1_score**	**0.364**

All but tfidf-character			
CV Score	0.432	**Avg F1_score**	**0.576**

Figure A.5.: Grid search results for the RBF SVM classifier

SGD				
Word feature			**Searched space**	**Result**
	ngram range		[(1,1),(1,2),(1,3),(1,4),(1,5)]	(1,3)
	sublinear tf		True, False	TRUE
	use idf		True, False	TRUE
	smooth idf		True, False	FALSE
	min df		0	0
	max df		[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]	0.3
	negate		True, False	TRUE
Score		**0.586**	**Avg F1_score**	**0.668**

Character feature			**Searched space**	**Result**
	ngram range		[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]	(1,6)
	sublinear tf		True, False	FALSE
	use idf		True, False	TRUE
	smooth idf		True, False	FALSE
	min df		0	0
	max df		[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]	0.5
Score		**0.588**	**Avg F1_score**	**0.673**

All / emoticon feature			
Score	**0.61**	**Avg F1_score**	**0.696**

All / neg. count feature			
Score	**0.623**	**Avg F1_score**	**0.701**

All / lexicon feature			
Score	**0.614**	**Avg F1_score**	0.688

All features			**Searched space**	**Result**
	loss		['hinge', 'log', 'squared_hinge']	hinge'
	alpha		[100, 10, 1, 0.1, 0.01, 0.001, 0.0001]	0.1
	penalty		['none', 'l1, 'l2', 'elasticnet']	elasticnet'
	Negcount (norm)		True, False	FALSE
	Emoticon (norm)		True, False	TRUE
	Lexicon (norm)		True, False	TRUE
Score		**0.617**	**Avg F1_score**	**0.703**

Combined (word + character)			
Score	**0.622**	**Avg F1_score**	**0.693**

All / character			**Searched space**
Score	**0.579**	**Avg F1_score**	**0.679**

Figure A.6.: Grid search results for the SGD classifier

SVM: Sigmoid			
Tfidf transformer		**Searched space**	**Result**
	ngram range	[(1,1),(1,2),(1,3),(1,4),(1,5)]	(1,5)
	sublinear tf	True, False	TRUE
	use idf	True, False	TRUE
	smooth idf	True, False	TRUE
	min df	0	0
	max df	[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]	0.3
	negate	True, False	TRUE
Score	**0.573**	**Avg F1_score**	**0.64**

Tfidf Char transformer		**Searched space**	**Result**
	ngram range	[(1,6), (2,6), (3, 6), (1,5), (2,5), (3,5), (1,4), (2,4)]	(1,4)
	sublinear tf	True, False	FALSE
	use idf	True, False	TRUE
	smooth idf	True, False	TRUE
	min df	0	0
	max df	[0.1,0.2,0.3,0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]	0.4
Score	**0.535**	**Avg F1_score**	**0.597**

All but emoticons			
Score	**0.558**	**Avg F1_score**	**0.628**

All but negcount			
Score	**0.56**	**Avg F1_score**	**0.633**

All but lexicon			
Score	**0.554**	**Avg F1_score**	**0.625**

Combined (all)		**Searched space**	**Result**
	C	[1,10,100]	1
	Gamma	[0.1, 0.01, 0.001, 0.0001]	0.001
	Negcount (norm)	True, False	TRUE
	Emoticon (norm)	True, False	TRUE
	Lexicon (norm)	True, False	TRUE
Score		**Avg F1_score**	**0.638**

Combined (only tfidf + char)			
Score	**0.552**	**Avg F1_score**	**0.61**

All but tfidf-character		**Searched space**	
Score	**0.593**	**Avg F1_score**	**0.645**

Figure A.7.: Grid search results for the Sigmoid SVM classifier

B. List of confusion matrices from all classifier experiments

Below are the confusion matrices from the experiments for all the classifiers not listed in Chapter 9.

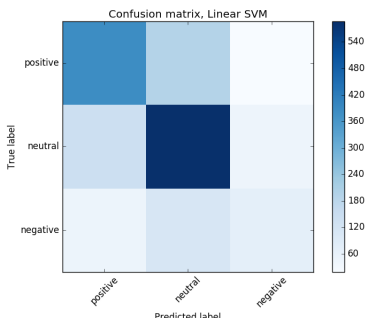


Figure B.1.: Confusion Matrix for Linear SVM

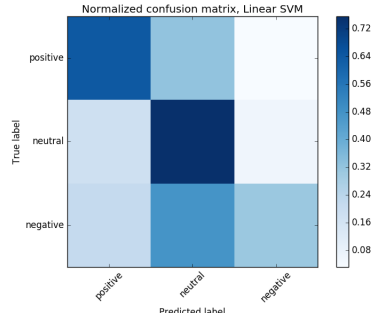


Figure B.2.: Normalized Confusion Matrix for Linear SVM

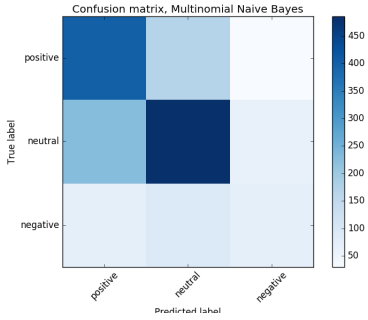


Figure B.3.: Confusion Matrix for Multinomial NB

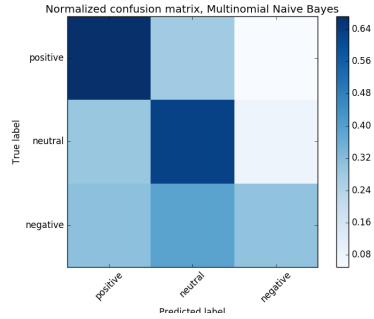


Figure B.4.: Normalized Confusion Matrix for Multinomial NB

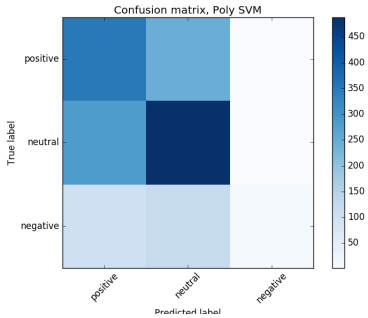


Figure B.5.: Confusion Matrix for Poly SVM

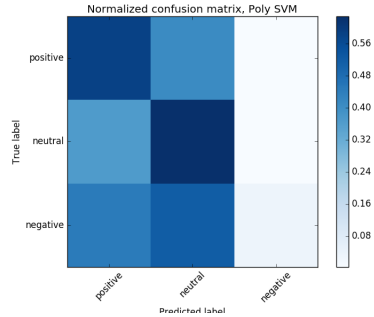


Figure B.6.: Normalized Confusion Matrix for Poly SVM

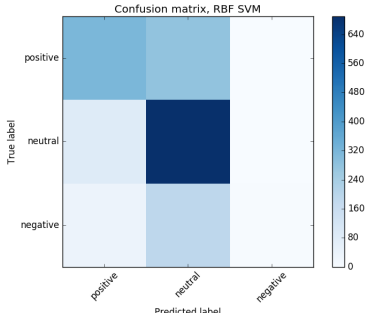


Figure B.7.: Confusion Matrix for RBF SVM

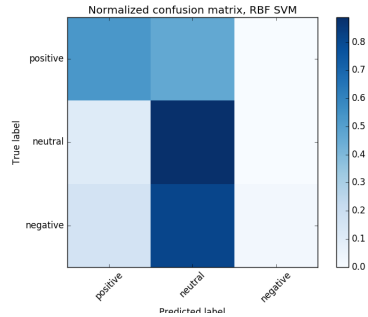


Figure B.8.: Normalized Confusion Matrix for RBF SVM