

# Utilizing Large Twitter Corpora to Create Sentiment Lexica

Valerij Fredriksen, Brage Jahren, Björn Gambäck

Department of Computer Science

Norwegian University of Science and Technology

Trondheim, Norway

valerij92@gmail.com, brage.jahren@gmail.com, gamback@ntnu.no

## Abstract

The paper describes an automatic Twitter sentiment lexicon creator and a lexicon-based sentiment analysis system. The lexicon creator is based on a Pointwise Mutual Information approach, utilizing 6.25 million automatically labeled tweets and 103 million unlabeled, with the created lexicon consisting of about 3 000 entries. In a comparison experiment, this lexicon beat a manually annotated lexicon. A sentiment analysis system utilizing the created lexicon, and handling both negation and intensification, produces results almost on par with sophisticated machine learning-based systems, while significantly outperforming those in terms of run-time.

**Keywords:** Pointwise Mutual Information, Sentiment lexica, Lexicon-based sentiment analysis

## 1 Introduction

A popular social medium providing opinionated texts is the micro-blogging service Twitter. On Twitter, users can post textual entries of up to 140 characters, commonly called *tweets*. Each day, approximately 500 million new tweets are posted; a fraction of those are made available through Twitter's public API. Large datasets can therefore easily be acquired, making Sentiment Analysis (SA) of tweets particularly popular.

The present study shows that two features in particular stand out when building Twitter sentiment classification systems: the effect of using sentiment lexica in the classification process and the run-time performance. The use of sentiment lexica is shown to be the single most valuable system component in terms of overall system performance. Without using sentiment lexica, the performance dropped as much as 4%. In general, most present-day Twitter specific lexica only contain word unigrams and bigrams. Here, in contrast, the effect of including longer phrases in sentiment lexica is explored, as well as utilizing the long phrases in the classification process, while striving to create a lexicon-based SA system with real time performance. Section 2 gives an overview of the state-of-art in automatic creation of sentiment lexica and lexicon-based SA, while Section 3 describes the datasets used. In Section 4, the overall architecture of both the lexicon-based SA system and the lexicon creator system are detailed. Section 5 includes the tests conducted on the created lexicon and lexicon-based SA system, which are further discussed in Section 6. Finally, Section 7 concludes and outlines future work.

## 2 Related Work

Lexicon-based sentiment analysis is the task of performing SA solely based on sentiment lexica. The overall performance of a lexicon-based SA system is hence closely related to the quality of the sentiment lexica, but also to *how* the sentiment lexicon information is used. In recent years, most SA systems use a Machine Learning (ML) approach, with sentiment lexica as a feature among others. The number of new lexicon-based SA systems that have appeared is therefore few. Although the task of classification differs

between a lexicon-based and an ML-based SA system, the sentiment lexica features in ML systems are grounded in the same ideas as lexicon-based SA systems. The common approach consists of two main steps: sentence analysis and sentiment calculation.

In the first step, each sentence in a tweet is analysed in search of specific features. A state-of-the-art feature set was identified by Mohammad et al. (2013). It includes word and character  $n$ -grams, word clusters, part-of-speech tags, the number of emoticons and consecutive punctuation marks, and negation. In the second step, these features are used to adjust raw sentiment values for the  $n$ -grams in a sentence that are looked up in Prior Polarity Sentiment Lexica. To calculate the final sentiment value, the sentiments of all  $n$ -grams in the sentence are summed up, and adjusted in accordance to any negation and intensification detected during the analysis step.

Prior Polarity Sentiment Lexica can be created in several ways; most commonly by manual annotation. However, manual annotation of large amounts of data is costly. This also applies to the task of annotating sentiment lexica, where each  $n$ -gram should be assigned a sentimental strength. Before beginning the annotation process, the  $n$ -grams to include in the lexicon are collected, e.g., from a large corpus of texts capturing the language features the lexicon is aimed at. Mohammad et al. (2013) selected all unigrams and bigrams as candidate entries to include in the lexicon, while Velikovich et al. (2010) selected all  $n$ -grams up to length 10 before filtering out  $n$ -grams based on their frequency and mutual information.

When using a corpus of labeled documents to annotate candidate  $n$ -grams, the association measure Pointwise Mutual Information (PMI) can be used. PMI quantifies the information shared between events (Fano, 1961). The mutual information  $I(x, y)$  between two events  $x$  and  $y$  with probabilities  $P(x)$  and  $P(y)$ , and joint probability  $P(x, y)$  is defined as:

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

If there is an association between  $x$  and  $y$ , the joint probability will be higher than the product of the individual prob-

Name	Pos	Neut	Neg	Total
2013-train	3283	4175	1290	8748
2013-test	1258	1367	462	3087
2014-test	794	564	151	1509
2015-test	1038	987	365	2390
2016-test	7059	10342	3231	20632

Table 1: Overview of the SemEval datasets as used here

abilities ( $P(x, y) > P(x)P(y)$ ), while it will be equal to the product of the individual probabilities if the events are independent (i.e.,  $P(x, y) = P(x)P(y) \Rightarrow I(x, y) = 0$ ).

Turney and Littman (2002) proposed a method for creating sentiment lexica where the sentimental orientation of a word could be calculated from the PMI value of a word  $w$  in a positive context minus the value of the same word in a negative context. They used a seed set of positive and negative words, and decided the context of a word by its proximity to a seed word. Mohammad et al. (2013) similarly used a seed set, but one containing positive and negative hashtags and emoticons to create the *Sentiment140* and *HashtagSentiment* lexica.

Another method using unlabeled documents is the graph approach, where similarity between all candidate  $n$ -grams, and between those and a seed set of positive and negative words are calculated before each  $n$ -gram is initialized as a node in a graph. Two variants have been proposed: In *graph propagation* (Velikovich et al., 2010), the nodes representing the seed set are initialized with a sentiment score according to their orientation, before propagating their sentiment value to the nodes laying on a path of length  $k$  away. In *label propagation* (Zhu and Ghahramani, 2002), the sentiment value is calculated as the weighted average of an  $n$ -gram’s neighbours. Whereas each node in graph propagation only holds the max path from a seed word, the nodes in label propagation can possibly hold multiple paths to a specific seed word. In both variants, the edges between the nodes are weighted according to their similarity.

### 3 Datasets

To be able to create a sentiment lexicon based on raw tweets, about 400 million tweets were downloaded using the Twitter Streaming API<sup>1</sup> in the period 18/12/2015–19/03/2016 (92 days), with about 4 million tweets downloaded each day. The raw tweets were filtered, removing all tweets that contained “RT @” (retweets), a URL or the ° symbol (automated weather or GPS location services). In addition, tweets ending with a number were removed, since spammers often keep tweeting the same message with an incrementing number at the end, to combat Twitter’s spam detection. The resulting filtered dataset contained about 103 million tweets and was used to generate the  $n$ -grams. For comparison, five manually annotated Twitter sentiment dataset from International Workshop on Semantic Evaluation (SemEval) were utilized: the training and test sets from SemEval 2013, and test sets from SemEval 2014, 2015 and

<sup>1</sup><https://dev.twitter.com/streaming/reference/post/statuses/filter>

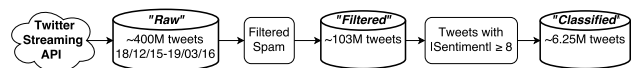


Figure 1: Datasets created using Twitter Streaming API

2016 (Nakov et al., 2016). Some of the tweets have been deleted since the datasets were originally created and are no longer available. Table 1 shows the class distributions of the datasets as used in this work. The 2013-train set was used for training, and the four test sets for testing only.

To be able to use the PMI approach when creating a sentiment lexicon, a large dataset of labeled tweets was also needed. For the approach to produce good and reliable results, the amount of labeled tweets need to be much higher than the tweets made available through SemEval. Hence our lexicon based sentiment analysis system (Section 4.3) was used together with the manually annotated sentiment lexicon *AFINN* (Nielsen, 2011) to automatically label the tweets. For the labeling to be as accurate as possible, only tweets with absolute sentiment score above a certain value were extracted. The value chosen was found using grid search and is a trade-off between precision and recall. Based on the 103 million unlabeled tweets, the labeling process yielded a labeled dataset containing 6.25 million classified tweets, of which 58.7% were labeled as positive and the rest as negative. The overall dataset creation process is shown in Figure 1.

## 4 Architecture

A PMI-based approach was used to build a system for automatic creation of a sentiment lexicon, and a lexicon based classifier was developed, utilizing the features of the lexicon created. Both systems use the same tweet preprocessing and vocabulary tokenization components.

The preprocessing includes handling of word elongation (e.g., ‘goddd’) using Levenshtein distance. The process consists of two parts: a four step dictionary creation part inspired by Brody and Diakopoulos (2011) and a word correction part using the created dictionary. The word is first reduced to its condensed form, before being looked up in the dictionary. If found, the dictionary returns the most likely spellings of the initial word. The Levenshtein distance is then calculated between the initial word and each of the returned words, before correcting the initial word to the closest match. This way, with a dictionary containing both ‘good’ and ‘god’, ‘goddd’ is 2 deletions away from ‘god’ or 2 deletions and 1 addition from ‘good’, so will be reduced to ‘god’.

Tokenization consists of splitting sentences into the longest non-overlapping  $n$ -grams also found in a given vocabulary.  $n$ -grams not found in the provided vocabulary are tokenized as unigrams. The reason for keeping the words not present in the vocabulary is that these words may be intensifiers or negators, which are excluded from the created lexicon.

### 4.1 PMI Lexicon

The PMI lexicon builder consists of vocabulary identification, counting vocabulary occurrences in a polarized dataset, and sentiment calculation as illustrated in Figure 2.

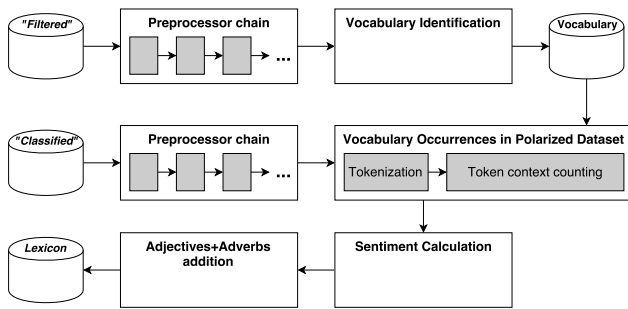


Figure 2: PMI lexicon creation architecture

The vocabulary identification step extracts and selects candidate PMI  $n$ -grams for a context vocabulary based on the large unlabeled, filtered dataset. The process of selection uses PMI  $n$ -grams (and is thus quite different from the graph propagation’s strictly  $n$ -gram frequency based selection described below). For an  $n$ -gram with  $n > 1$  to be selected, the PMI of the included words needs to be higher than a predefined threshold. This way only  $n$ -grams containing words that together mean something or form a common phrase are selected. In addition,  $n$ -grams ending on a stopword or containing intensifiers are filtered out. Unigrams are not selected as candidate entries, but are introduced later to the system.

In order to calculate sentiment values of  $n$ -grams in the context vocabulary, the automatically labeled (classified) dataset is used. The tokenizer splits the entries into the longest possible non-overlapping  $n$ -grams. Each individual token holds counters for occurrences in positive and negative contexts. For each  $n$ -gram in the context vocabulary, a sentiment value is calculated as  $\log_2 \frac{\text{freq}(w, \text{pos}) \cdot \text{freq}(\text{neg})}{\text{freq}(w, \text{neg}) \cdot \text{freq}(\text{pos})}$ , using the number of times the  $n$ -gram occurs in positive tweets and in negative tweets. The lexicon is created by adding all  $n$ -grams with absolute sentiment value above a defined threshold and an occurrence frequency in the labeled dataset above a set frequency.

With the created lexicon, all unigrams are run through an adjective and adverb addition algorithm, adding all missing adjective and adverb forms of the unigram to increase the coverage of the lexicon. The missing adverbs and adjective forms are derived based on a set of rules for forming comparatives and superlatives,<sup>2</sup> and for forming adverbs from adjectives.<sup>3</sup> These are added to the lexicon only if they were previously encountered in the tokenization process and assigned the same sentiment value as their related  $n$ -gram. The resulting lexicon forms the final PMI lexicon.

## 4.2 Graph Propagation Lexicon

For comparison, a lexicon was also created using graph propagation. The implementation of the graph propagation approach consists of four steps: vocabulary identification, context-vector creation, graph creation, and sentiment propagation. During the vocabulary identification step a set of candidate  $n$ -grams are identified and selected, forming

<sup>2</sup><http://www.eflnet.com/tutorials/adjcompsup.php>

<sup>3</sup><http://www.edufind.com/english-grammar/forming-adverbs-adjectives/>

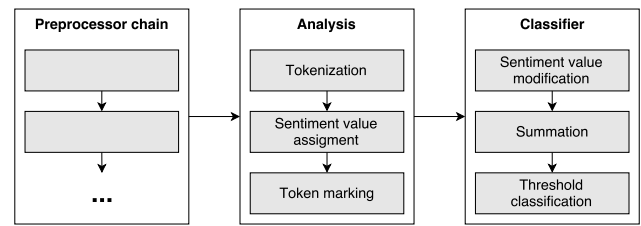


Figure 3: Lexicon classifier system architecture

a context vocabulary. Each tweet is processed into all possible  $n$ -grams with length up to  $n = 5$ , where candidate  $n$ -grams are selected based on their occurrence frequency in the large unlabeled tweet dataset. All  $n$ -grams below a frequency threshold, in addition to all  $n$ -grams ending with a stopword or containing an intensifier are filtered out.

To compare candidate  $n$ -grams, their context vectors are created using the COALS method (Rohde et al., 2006), i.e., by summing up the word frequencies of  $x$  words before and  $x$  words after the  $n$ -gram, over all mentions of the  $n$ -gram in the dataset, weighted by the distance from the  $n$ -gram using a ramped window of size 6 (words next to the  $n$ -gram on either side increase the frequency of that word in the context vector by 6 for that side, etc.). A matrix containing all  $n$ -grams is created, with rows containing the context vectors of all selected  $n$ -grams and columns containing the occurrence frequencies of words occurring in the vectors. The values are normalized using Pearson Correlation, and all negative values are then set to 0, while all positive values are squared.

Given the candidate entries and their context vectors, the graph is created, with nodes representing the candidate entries, and edges with weights representing node similarity. To create the edges, cosine similarity is calculated between all  $n$ -gram pairs, represented by their context vectors. As in Velikovich et al. (2010), an edge is created between two nodes if their similarity is greater than a set threshold. The seed nodes then propagate their sentiment values through the finished graph. In our implementation, each seed node can affect nodes that are connected to it via two or less other nodes. When all seed nodes have propagated their sentiment value, the final sentiment value of each node is calculated by subtracting the sum of all negative max paths from the sum of all positive max paths. The negative and positive max paths to a node are the maximum sentiment values each connected seed node affects the node with. A node with more paths to positive than negative seed nodes will most likely get a positive sentiment value. Finally, the lexicon is created by extracting all the  $n$ -grams and their sentiment values.

## 4.3 Lexicon Based Sentiment Analysis

The lexicon based Sentiment Analysis system accepts single tweets or a set of tweets, and outputs a predicted classification per tweet. The predicted classification is determined by running each tweet through three main stages: preprocessing, analysis and classification, as shown in Figure 3.

The analysis consists of detecting negation cues, intensifiers words, and punctuation marks, and assigning sentiment values. This is done by first applying preprocessing

System	2013	2014	2015	2016
Graph	0.4942	0.4844	0.4571	0.4744
PMI	0.6130	0.6170	0.5711	0.5685

Table 2: Graph approach vs. PMI approach

and tokenization, splitting the tweet into tokens that are looked up in the provided sentiment lexicon and assigned the lexicon value, if found, or a sentiment value of zero, if not found in the lexicon. If a token matches a negation cue (*‘not’*, *‘wouldn’t’*, etc.), a simple method proposed by Das et al. (2001) is used, whereby  $n$  consecutive words appearing after the negation cue are marked as negated. If a token matches an intensifier, the next token is marked as intensified. Finally, a sentence final *‘!’* or *‘?’* marks all tokens in the sentence as intensified. The sentiment value of each token found in the lexicon is then calculated as  $(L \cdot I) - N$ , where  $L$  is the token’s lexicon value,  $I$  the intensification value ( $I = 1$  if not intensified), and  $N$  the negation value.

The value of  $I$  for a token is dependent on what intensifier word or punctuation mark it is affected by. Some intensifier words such as *‘kind of’* or *‘hardly’* will work as dampeners with  $I$ -values between 0 and 1. Words like *‘incredibly’* or *‘extremely’* will on the other hand work as boosters with  $I$ -values above 1. A token can be intensified by both an intensifier and a punctuation mark. In such case,  $I$  is the product of the intensification constant of the intensifier and the intensification constant of the punctuation mark. The  $I$ -values of *‘!’* and *‘?’*, and the  $N$  value along with their influence ranges are determined through grid search.

The final sentiment score of a tweet is calculated as the sum of its tokens’ sentiment values, and it is classified as either positive, negative or neutral by comparing its sentiment value against two thresholds that set the lower and upper bounds for tweets to be classified as neutral. The thresholds are also determined through grid search.

## 5 Experiments

A series of experiments were conducted, both to determine how well the lexicon based classifier, utilizing the PMI lexicon, fares against more complex classifiers, and to investigate the effect of using its output as a feature in an ML classifier. For the system to perform as well as possible, the optimal parameters for both the lexicon creator and the classifier need to be identified. Rather than a complete grid search where all combinations of parameter values are explored, the search consisted of a series of iterative steps, with each step testing all predefined values of a single parameter, while keeping all the other variables constant.

The following parameters were optimised for the inclusion of an  $n$ -gram in the vocabulary: maximum length and minimum occurrence frequency; and for inclusion in the lexicon: minimum occurrence in polarized dataset, and minimum PMI and absolute sentiment values. For the lexicon classifier, the optimised parameters were: the negation constant  $N$  (Section 4.3) and the number of tokens after a negation cue to include in the scope, the intensification constants for tokens inside sentences ending with *‘!’* resp. *‘?’*, and

Lexicon	2013	2014	2015	2016
Sentiment140	0.5260	0.5415	0.4788	0.5186
AFINN	0.6259	0.6009	0.5801	0.5882
PMI	0.6748	0.6578	0.6201	0.6032

Table 3: F<sub>1</sub> scores of different lexica

the thresholds between *neutral* and *positive* resp. *negative* sentiment classification.

### 5.1 Graph-based Approach vs. PMI-based

Applying the graph propagation to lexicon creation encountered several problems. First with creating the seed set: a small seed produced smaller lexica with weaker sentiment values since fewer words were propagating the values, while a large seed set had problems setting good starting sentiment values. Second with creating the context vector: similarity between  $n$ -grams is calculated as similarity between their context vectors, but the relationship between multi-grams and their neighbouring words is less clear. A third problem is that calculating the cosine similarity for each word with every other word makes the computational complexity exponential.

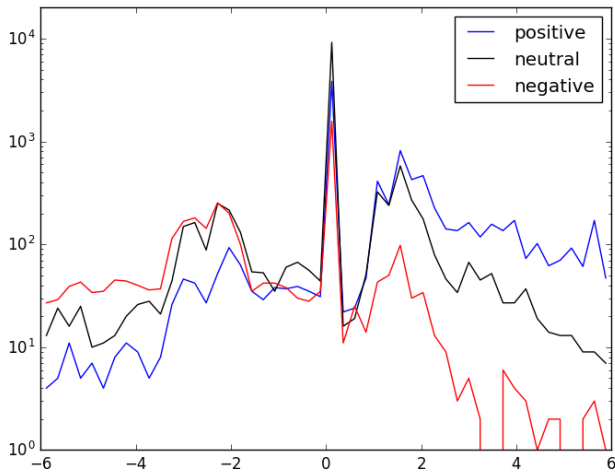
Most importantly, the PMI approach clearly outperformed graph propagation when tested on the 2013–2016 SemEval data, as shown in Table 2.

### 5.2 Lexicon Comparison

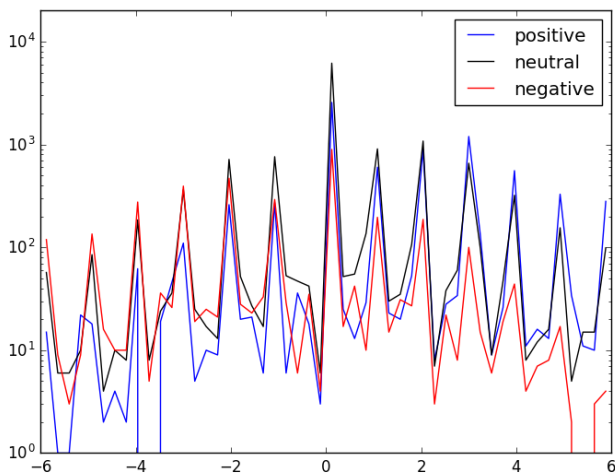
Table 3 compares the PMI lexicon against other previously created sentiment lexica. The *Sentiment140* lexicon (Mohammad et al., 2013) was also created using a PMI approach, making it an obvious choice for comparison, in addition to the *AFINN* lexicon (Nielsen, 2011) used in the creation of our *Labeled* dataset. The comparison was done by running our lexicon based classifier on the four SemEval datasets, using the PMI lexicon, *Sentiment140*, and *AFINN* as lexicon, respectively. As can be seen in the table, the PMI lexicon outperforms the other two sentiment lexica. *AFINN* is the closest, with the smallest difference of approximately 0.015 on the 2016 dataset.

Although the results seem to point to the PMI lexicon being the best, it should be noted that this lexicon was specifically tailored to work with the classifier. *AFINN* should not be affected by this, since it only contains words without special characters and has a single sentiment value per lexicon entry similar to the PMI lexicon. For *Sentiment140* on the other hand, the classifier is not able to utilize all features the lexicon provides: negation is handled differently and the preprocessor removes all non-alphanumerical characters except characters forming emoticons, so there are *Sentiment140* entries the classifier never is able to use. Hence no clear conclusion can be drawn for *Sentiment140*. We do, however, stipulate that the method of creating a labeled dataset for the lexicon creator is superior to the hashtag and emoticon approach used to create *Sentiment140*, and that the use of larger  $n$ -grams benefits the PMI lexicon.

Since the comparison between the PMI lexicon and *AFINN* is the most reasonable, Figure 4 depicts the distribution of predicted positive, negative and neutral tweets. In Figure 4a



(a) PMI lexicon



(b) AFINN lexicon

Figure 4: Sentiment value histograms

we can see a clear distinction between the classes, while in Figure 4b, the classes are much closer and overlapping. The zig-zagged pattern in Figure 4b is due to *AFINN* lexicon only using integer sentiment values. The separability of the different classes displayed in the graphs shows why the PMI lexicon will generally classify more tweets correctly.

### 5.3 System Performance

To test the overall system performance, the lexicon based classifier, utilizing the PMI lexicon, was compared against a baseline system (Jahren et al., 2016) and VADER, Valence Aware Dictionary and sEntiment Reasoner (Hutto and Gilbert, 2014), a lexicon based sentiment analysis tool specifically tuned towards social media. VADER utilizes a Support Vector Machine (SVM) classifier and goes beyond bag-of-words by taking into consideration word order and degree modifiers.

The baseline system was created as a state-of-the-art Twitter Sentiment Analysis (TSA) system based on the most common approaches within the field. It was implemented in the Scikit-Learn (Pedregosa et al., 2011) machine learning framework, also using an SVM classifier, but trained on word and character  $n$ -grams, word clusters, part-of-

	System	Prec	Rec	F <sub>1</sub>	Time
2013	VADER	0.6540	0.6573	0.6508	1.29
	Baseline	0.7209	0.7120	0.7073	74.01
	Lexicon	0.6866	0.6803	0.6748	0.04
2014	VADER	0.6536	0.6421	0.6441	0.63
	Baseline	0.7091	0.6832	0.6847	41.38
	Lexicon	0.6818	0.6554	0.6578	0.02
2015	VADER	0.6310	0.6201	0.6197	0.99
	Baseline	0.6862	0.6548	0.6527	64.80
	Lexicon	0.6483	0.6213	0.6201	0.03
2016	VADER	0.5939	0.5919	0.5928	8.63
	Baseline	0.6461	0.6434	0.6431	538.88
	Lexicon	0.6085	0.6028	0.6032	0.19

Table 4: Sentiment classifier performance

System	2013	2014	2015	2016
Baseline	0.7073	0.6847	0.6527	0.6431
+ Lexicon	0.7216	0.6911	0.6586	0.6356

Table 5: Baseline system with PMI lexicon

speech tags, emoticons, punctuation, and sentiment information from VADER, from the automatically annotated lexica *Sentiment140* and *HashtagSentiment* (Mohammad et al., 2013), and from the manually annotated lexica *MPQA* (Wilson et al., 2005), *BingLiu* (Hu and Liu, 2004), *AFINN*, and *NRC Emoticon* (Mohammad et al., 2013). Note that this is a strong baseline: the system achieved the second highest tweet-level accuracy in SemEval’16 Task 4.

All systems were trained on the SemEval 2013 training data, and their performance compared across four SemEval test datasets is shown in Table 4. The *VADER Sentiment* system scores below the lexicon based classifier across all performance measures on all four datasets, while the state-of-the-art baseline system consistently scores above it. However, the lexicon based system significantly outperforms the other systems when it comes to run-time. Even on the smallest dataset (2014), it executes approximately 31.5 times faster than VADER and approximately 2 000 times faster than the baseline system. With an execution time of 0.19 seconds on the 2016 dataset, it classifies tweets with a speed of 108 600 tweets per second, against the speed of the baseline system with 38 tweets per second.

Experiments were also run using the lexicon based classifier as a feature in the baseline system. Table 5 shows a clear increase in performance in terms of F<sub>1</sub> score from 0.7073 to 0.7216 on the 2013 dataset when adding the lexicon based classifier as a feature. There is also an increase on the 2014 and 2015 datasets, although not as apparent. On the 2016 dataset on the other hand, the performance actually drops, which might be due to that dataset containing more noise and annotation errors than the other datasets.

Size	2013	2014	2015	2016
500	0.3387	0.2736	0.3029	0.3936
1 500	0.6533	0.6343	0.6049	0.6016
<b>3 000</b>	0.6748	0.6578	0.6201	0.6032
10 000	0.6674	0.6507	0.6201	0.6038
25 000	0.6328	0.6203	0.6020	0.6011
50 000	0.6052	0.6092	0.5650	0.5976
100 000	0.6040	0.6090	0.5648	0.5851
200 000	0.5965	0.6020	0.5626	0.5938

Table 6: Comparison of different sized PMI lexica

## 5.4 Lexicon Size Comparison

To explore how the size of the created sentiment lexicon affects the performance, eight sentiment lexica of sizes ranging from 500 to 200 000 entries were created and tested on the four SemEval datasets. As we can see from Table 6, as long as the lexicon size is above a certain threshold, the overall performance remains acceptable. However, there is a continuous drop in performance when the lexicon size passes 10 000 entries. The best performing lexicon from the above test has 3 000 entries, which is quite interesting looking at the *Sentiment140* lexicon with approximately 300 000 entries in comparison. More words and phrases do not necessary lead to better results.

In addition to the  $F_1$ -score, the difference in coverage between the different sized lexica was explored. Four different coverage measures were used:

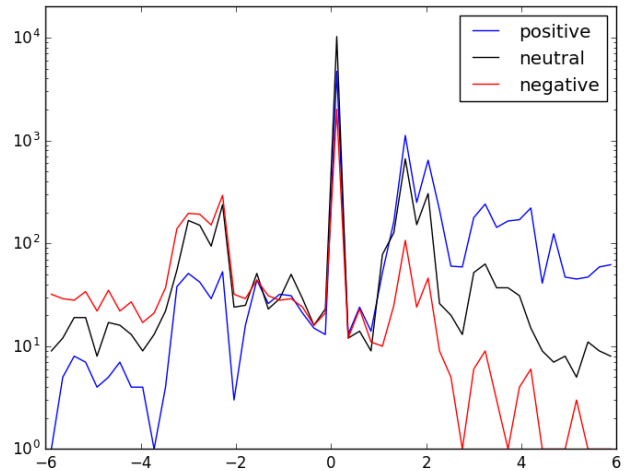
- **NZS** (Tweets with Non-Zero Sentiment): Ratio of tweets where final predicted sentiment value was  $\neq 0$ .
- **TwM** (Tweets with Multi-grams): Ratio of tweets that included at least one multi-gram from the lexicon.
- **WiL** (Words in Lexicon): Ratio of words in tweets found in the lexicon.
- **FWiL** (Frequent Words in Lexicon): Ratio of top 1 000 most common, non-stop-word words found in the lexicon.

Table 7 shows that the best performing lexicon with a size of 3 000 only contains words found in 47% of the tweets. That means that almost half of the tweets end up with a sentiment score of zero and are classified as neutral. In addition, only approximately 4% of the tweets contain multi-grams ( $n$ -grams with  $n > 2$ ) found in that lexicon. Although the scores for the different coverage measures increase with the lexicon size, meaning larger lexicon lead to higher coverage, it does not look like higher coverage means better classification performance.

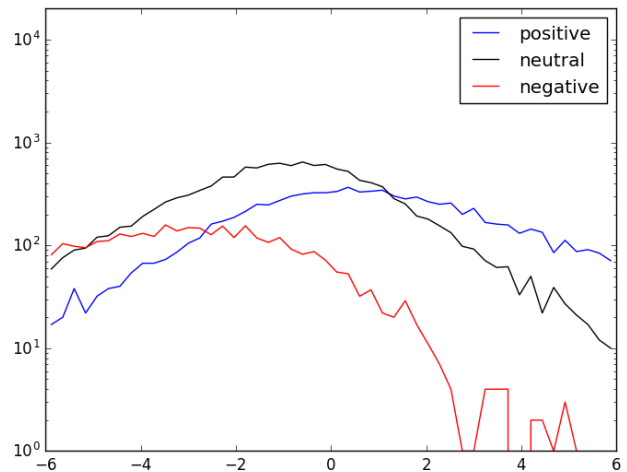
Figure 5 shows the distribution of positive, negative and neutral tweets given their sentiment value predicted by the classifier on a PMI lexicon of size 1 500 and of size 200 000. In comparison to Figure 4a, showing the same distribution for the best PMI lexicon, the difference in performance is even further visualized. Compared to the distribution of our best PMI lexicon, the distribution of the 200 000 lexicon, shown in Figure 5b is almost a uniform distribution, the large spike around 0 is gone, but the differences between classes are smaller. The other extreme is the distribution of the small lexicon, shown in Figure 5a. It

Size	NZS	TwM	WiL	FWiL
500	0.0435	0.0066	0.3567	0.0081
1 500	0.3902	0.0086	0.3793	0.0655
<b>3 000</b>	0.4726	0.0412	0.3632	0.0826
10 000	0.4922	0.0702	0.3647	0.0846
25 000	0.8421	0.1788	0.4012	0.1712
50 000	0.9968	0.3322	0.5229	0.4371
100 000	0.9992	0.4703	0.6122	0.7019
200 000	0.9996	0.6012	0.6692	0.9889

Table 7: Lexicon size vs. coverage



(a) Size 1 500 PMI lexicon



(b) Size 200 000 PMI lexicon

Figure 5: Lexicon size sentiment value histograms

classifies some of the positive and negative tweets well, but most tweets are left with a score of 0.

## 6 Discussion

After identifying the best performing PMI lexicon, a few statistics were gathered in order to see whether or not the resulting lexicon contained the features we expected. It also provides a general insight into the lexicon created. From Table 8, we can clearly identify all of the top 10 positive lexicon entries as actual positive phrases and the top 10 negative as negative phrases. In addition we do not see any uni-



<i>n</i> -gram	Positive		Negative	
	<i>n</i> -gram	Value	<i>n</i> -gram	Value
you have a great day		5.00	a bad bitch	-5.00
you have an amazing day		4.48	bitch ass nigga	-4.95
hope its a good one		4.33	dumb ass	-4.82
happy birthday i hope		4.21	fuck that bitch	-4.75
you had a great day		4.10	fuck a bitch	-4.75
you have a good day		4.08	bad right now	-4.73
you have a wonderful day		4.02	weak ass	-4.72
hope you have a great		4.01	fuck this shit	-4.68
you have a good one		3.99	fuck fuck fuck	-4.66
i love love love		3.96	fool me twice	-4.63

Table 8: Top 10 positive and negative entries











Emoji	Positive		Emoji	Negative	
	Alias	Value		Alias	Value
	Birthday Cake	2.76		Pouting Face	-2.64
	Wrapped Present	2.60		Parking Sign	-2.41
	Balloon	2.51		Angry Face	-2.41
	Confetti Ball	2.47		Triumph Face	-2.40
	Party Popper	2.46		Litterbox	-2.34

Table 9: Top 5 positive and negative emojis

grams, meaning that the scoring of longer *n*-grams higher than shorter *n*-grams seems to be working. The detected emojis were treated the same way as the phrases, as seen in Table 9. All emojis listed as positive are commonly used to further express positive sentiment in positive sentences, while all of the emojis listed as negative are commonly used in negative sentences.

To detect the impact each feature of the lexicon creator and classifier imposes on the performance, an ablation study was conducted. Table 10 shows that the impact of each feature is very subtle, with the single most important feature being the adjective and adverb addition which increases the  $F_1$ -score on the 2013 dataset by 0.01. This means that the classifier performance almost entirely depends on the quality of the PMI lexicon, with the additional features only contributing a small amount.

## 7 Conclusion and Future Work

The paper has shown that the Pointwise Mutual Information (PMI) lexicon creation approach works well: Lexicon comparison experiments showed a fully automatically created PMI lexicon beating the manually annotated *AFINN* lexicon on all datasets and across all performance measures. In addition to verifying the quality of the lexicon, this proves that creating sentiment lexica automatically is highly viable. While previous work building PMI lexica have been based on rather small sets of tweets, the present work is based on over 100 million tweets, while still being highly efficient.

However, for a classifier to utilize a sentiment lexicon’s full potential, the classifier must be specifically tailored to work with that specific lexicon. This is a consequence of the different sentiment lexica creation methods, where the

Features	2013	2014	2015	2016
All	0.6748	0.6578	0.6201	0.6032
- Adj/Adv	0.6653	0.6525	0.6127	0.6032
- Negation	0.6732	0.6561	0.6180	0.6030
- Intensification	0.6733	0.6548	0.6193	0.6036
- PMI <i>n</i> -grams	0.6714	0.6503	0.6161	0.6010

Table 10: Feature ablation results ( $F_1$ -scores)

creators of the different available lexica apply different features specifically meant to work well in another system or classifier. This is more important than lexicon size: Larger lexica did lead to a better coverage as shown in Table 7, but classification performance was not improved accordingly. Although the results point to that larger lexica with high coverage would not perform better than relatively small lexica with medium coverage, no definite conclusions can be drawn. The fact that larger lexica created with the lexicon creator did not lead to better performance might also be caused by the quality of the labeled dataset used.

Regarding the run-time performance of lexicon based Sentiment Analysis (SA) systems compared to more sophisticated SA systems, the results clearly suggest that lexicon based systems are the most viable SA systems to use in real-time classification applications. The lexicon based classifier achieves a classification speed of 108 600 tweets per second, meaning that our system could have classified all of the 500 million tweets posted on Twitter each day in real-time 19 times over. With this result it would be possible to add more advanced features to the classifier, trading off run-time performance for better classification performance and still classify fast enough to be a real-time classifier.

The source of most misclassifications are the tweets with sentiment value of 0. In the current implementation, all these tweets are classified as neutral by default. It should be possible to extract several other values for each *n*-gram to the lexicon that can actually help classify a tweet as neutral instead of classifying all tweets with sentiment value of 0 as neutral. For example, each word could have a sentiment score and an objectivity score. The sentiment score would be calculated as here, while the objectivity score would be calculated also using the PMI approach, but on a dataset labeled as subjective/objective.

Where the PMI lexicon creation approach is only concerned with finding the sentiment values of *n*-grams, the graph propagation approach is most concerned with the relationship between the different *n*-grams and strives to find *n*-grams used in similar contexts. Since one of the problems of the graph propagation approach is to find a good seed set with appropriate weights, it could be interesting to explore a lexicon creation combining the PMI and graph propagation approaches. The PMI-approach could be used to identify a seed set with sentiment values, while the graph propagation could be used to find *n*-grams similar to the ones already present in the seed set. With more appropriate seed set values, the most similar *n*-grams found during graph propagation would tentatively be assigned more appropriate sentiment values.

