Machine learning – final project
Kfir Bar

Sentiment Analysis of Movie Reviews and Twitter Statuses

Introduction

Sentiment analysis is the task of identifying whether the opinion expressed in a text is positive or negative in general, or about a given topic. For example: *"I am so happy today, good morning to everyone"*, is a general positive text, and the text: *"Django is such a good movie, highly recommends 10/10"*, expresses positive sentiment toward the movie, named Django, which is considered as the topic of this text. Sometimes, the task of identifying the exact sentiment is not so clear even for humans, for example in the text: *"I'm surprised so many people put Django in their favorite films ever list, I felt it was a good watch but definitely not that good"*, the sentiment expressed by the author toward the movie is probably positive, but surely not as good as in the message that was mentioned above. In many other cases, identifying the sentiment of a given text is very difficult for an algorithm, even when it looks easy from a human perspective, for example: *"if you haven't seen Django, you're not worth my time. if you plan to see it, there's hope for you yet."*
Nowadays, when the micro-blogging platforms, such as Twitter, are commonly used, the task of sentiment analysis becomes even more interesting. Micro-blogging introduces a new way of communication, where people are forced to use short texts to deliver their messages, hence containing new acronyms, abbreviations, and grammatical mistakes that were generated intentionally.
Although there are several known tasks related to sentiment analysis, in this project we will focus on the common binary problem of identifying the positive / negative sentiment that is expressed by a given text toward a specific topic. In other words, the texts that we deal with in this project, must express either positive or negative sentiment, and they cannot be neutral about the topic. There are other tasks that allow a text to be neutral about a specific topic, or even totally objective, i.e. expressing no interest in the topic at all. By narrowing down the problem like this, we get a classic binary classification case.
In this work, we will use two sets of manually annotated texts: 1) movie reviews from IMDB (www.imdb.com) – a world-known cinema portal, and 2) Twitter statuses. The second set is a collection of Twitter short messages (bounded by 140 characters), an annotated topic, and the sentiment that is expressed toward it. The annotation of the topic and the sentiment was done by human annotators. The first set contains reviews about various movies, unlimited in their text length, annotated by humans with their sentiment toward the movie being described. The topic in this case is not specifically mentioned, but it can be implied from the text. The main reason for using both sets in this project, was to show the difficulties of guessing the sentiment in short and often ungrammatical English texts, as opposed to a relatively long and well established English texts, as used in the second set. We used three classifiers: Naïve Bayes, KNN and SVM over a set of features that were extracted from the texts using techniques taken from the field of natural language processing (NLP).

Datasets

In this project, we used two datasets. Let us now describe the data sets in more details:

1) **IMDB–movie-review polarity dataset V 2.0**
   http://www.cs.cornell.edu/people/pabo/movie-review-data/

   This set is a collection of movie-review documents labeled with respect to their overall sentiment polarity (positive or negative). The set was released in June 2004 and it contains 1000 positive and 1000 negative processed reviews. The reviews were preprocessed by the dataset editors so that each review is formatted as a plain tokenized text, containing no structured information that can imply on the polarity of the text (e.g. stars rating – 0/5). The average review size (in words) is 746.3. Although this set is considered as a user-generated content the language is relatively grammatically correct in most cases, probably because users are not restricted with the size of the text.
2) **Sentiment Analysis in Twitter**
   http://www.cs.york.ac.uk/semeval-2013/task2/index.php?id=data

   This set is an ongoing shared task for sentiment analysis on Twitter statuses. Since our intention is to focus on the polarity variation of the problem (positive / negative), we use the dataset provided for Task B after removing tweets that were labeled either as "neutral" or as "objective". Each entry contains the tweet, the sentiment polarity (positive / negative) and the topic subject of the sentiment polarity. Table 1 shows an example such an entry.

Tweet	**Topic**	**Polarity**
Wow!! Lady Gaga is actually at the Britney Spears Femme Fatale Concert tonight!!! She still listens to her music!!!! WOW!!!	lady gaga	"positive"

Table 1 - Example of a single tweet entry

   In fact, due to legal constraints, the text of every tweet was not in the original corpus, but only the tweet ID. In order to get the text of tweet, one should use the Twitter API accordingly. Unfortunately, some of the tweets were already removed from the repository, so overall we left with 2,552 labeled examples, out of which 1814 are positives and 738 are negatives. In order to balance the set, we have only used 730 tweets of each polarity.
   The main challenges of processing tweets emerge because of the size restriction of 140 characters. This restriction makes people use shortcuts, omit prepositions and use emoticons (e.g. :D, _^_^_). Moreover, in Twitter, people are often use hash-tags attached to a single word expression, for referring to a trending topic (e.g. #FavouritePlayerFromEachPremierLeague) and the @ character when referring to another Twitter account (e.g. @kfirbar).

Features

Sentiment analysis may be considered as a text categorization problem (Pang and Lee, 2002) where the topic of a given textual document is guessed from a set of all the possible sentiment labels (e.g. in the polarity variation this set contains only the labels positive and negative). Based on this observation, we extract various features from the text, using NLP-based algorithms, and inject them into a classifier. Several related works (Pang and Lee, 2002; O'Keefe and Koprinska, 2009) show that using features extracted from the individual words (unigrams) get competitive results as of works that use longer *n*-gram phrases. In this project we will focus merely on unigrams. We used the open-source Java package OPEN-NLP (http://opennlp.apache.org/) to extract the relevant linguistic features from the texts of both corpora (IMDB and Twitter). OPEN-NLP is actually a set of classifiers that work on the word level, that is, every word is classified for various tasks. However, since the models that are provided with OPEN-NLP were trained on relatively long and well-structured texts, they do not perform as well on short and ungrammatical tweets. Still, since there are no other available tools known to us, we can live with this limitation.
We preprocessed the texts for removing all the unnecessary punctuations, except the emoticons, using a list of ~100 predefined emoticons. Also, only in tweets, where the topic is explicitly available, we remove the topic words from the text so we will not consider them as features. Table 2 summarizes the types of features, on the word level, that we used for classifications.

Feature	**Definition**	**Example**
Word	The word itself	hello, can't, @TIMENewsFeed, #thingsThatMatter,
Lemma	The dictionary entry of a word	thought = think, children = child, books = book
Part-of-speech (POS) tag	The part of speech of the word (e.g. Verb, Noun, Adjective), considering **it's context**	In the text - "I would like to **book** a ticket" the POS tag of the word "book" is **Verb**
Location	Indicates how far is the word from the topic (only in Twitter). The location is determined by the window of words that the specific word exists in. For example, window #1 contains all the words that their absolute distance (in word counts) from the topic is no far than 3 words. Overall, we used three windows:	In the text – "Wow!! Lady Gaga is **actually** at the Britney Spears Femme Fatale Concert tonight!!! She still listens to her music!!!! WOW!!!" the location window of the word "actually" is 1, and the location window

	1. up to 3 words 2. up to 6 words 3. more than 6 words	of the word “Britney” is 2

Table 2 - The features we used in our experiments

We experimented with different sets of the abovementioned features, as reported in the “experimental results” section. In general, every labeled example is represented by a vector of its features’ weights and its assigned label. In order to be aligned with other works in this field, we experimented with three different weighting functions:

1. Feature presence (FP) – a binary value, indicating whether the feature exists in the text or not (e.g. in the text – “hello world”, only the features “hello” and “world” are set to 1, and all the other words in the vocabulary are set to 0. The vocabulary is the set of all the words we see in the corpus).
2. Feature frequency (FF) – a real value, indicating the frequency of the feature in the given example, normalized by the size of the text (in words).
3. TF-IDF – a real value, indicating the frequency of the feature in the given text, divided by the logarithm of the number of examples from the corpus containing this feature. In other words, for feature f_i:

$$TF - IDF(f_i) = \frac{FF_i}{\log{(DF_i)}}$$

   where FF_i is the feature frequency of f_i as before, and DF_i is the document frequency of f_i, that is, the number of documents containing f_i. The intuition behind this weighting function is the give a larger weight to features that are seeing less in the corpus than the common ones. In other words, we increase the impact of rare words over common words.

Classifiers

We used three different classifiers: Naïve Bayes, kNN, and SVM. Intuitively, the features seem mutually dependent. Take for example, the text – “*Tomorrow shall consist of watching Will Ferrell films with my fave people and Chinese food, sweet*”. Here, the real sense of the word “watching” depends strongly on some other words (e.g. films, tomorrow), and this affects the sentiment polarity of the entire text toward the topic – Will Ferrell. Hence, Naïve Bayes may miss some of those dependencies, because by its nature, it assumes that the features are independent. SVM, on the other hand, does take those dependencies into consideration, and therefore, most state-of-the-art works in the field of text processing, use SVM. We use two kernels independently: linear and polynomial of degree 2. kNN was used with Euclidean distance, with and without feature normalization.

Environment

The entire program was written in Java. We used the OPEN-NLP Java library combined with WordNet 3.0 for text analytics and feature extraction. OPENNLP was used to extract the part-of-speech tags, and WordNet was used for determining the lemma of a given word. We then use WEKA, directly from the Java code, for running the classifiers on the training data. We test different features settings with the three classifiers and the results are reported and analyzed in the next section.

Experimental results and analysis

In our first set of experiments, we are trying to see how good each classifier performs on the datasets, with different features settings. All measurements are done on the average **accuracy** of the same 10-fold cross validation split. Table 3 elaborates on the different features settings we used in our experiments. We used the three weighting functions that were mentioned above with every features setting; the results are reported.

Settings name	**Description**
Unigram (U)	Using the words, without the subject words (only in Twitter)
Lemma (L)	Using the lemmas; when a lemma does not exist (e.g. proper name), we use the word itself. Like in the previous settings, we do not use the subject words
Lemma+POS (L+P)	Same as previous, but here every lemma is concatenated with its part-of-speech tag (e.g. book_VERB). Subject words were not considered
SelectedLemna (SL)	Only lemmas that their part-of-speech tag is one of the following: adjective, adverb, or verb. The motivation for this comes from seeing some external works showing that the subjective sentiment of an author is reflected mostly on verbs, adjectives and adverbs. Subject words were not considered
SelectedLemma+POS (SL+P)	Same as previous, but here every selected lemma is concatenated with its part-of-speech tag. Subject words were not considered
LemmasAndPOS (LP)	Using the lemmas and their corresponding part-of-speech tags as individual features. Subject words were not considered
Lemma+Location (L+LOC)	Using the lemmas concatenated with their location, i.e. their distance from the subject (as mentioned above)

Table 3 - Features settings

The entire set of results is reported in Appendix A; here we analyze the results and provide some interesting insights. We begin by some general observations:

1. On both datasets, IMDB and Twitter, the best result was achieved by SVM with polynomial (degree 2) kernel. Interestingly, on IMDB, the best features set is the one that use lemmas weighted with the feature-

presence function. This observation was reported also by other related works. However, on Twitter, the best result was reported on lemmas and part-of-speech tags (as individual features), weighted with TF-IDF. This makes sense somehow; recall that Twitter message are very short and the part-of-speech tags add some more information that seem important for this task. Note that when using the feature presence weighting function, the accuracy was not increased when adding the part-of-speech tags to the lemmas (0.7654 only lemmas Vs. 0.7639 when using part-of-speech tags as well). This is probably because when using feature-presence, there is not enough information to know that a verb (for example) was used in the text; the frequency and document distribution of the verbs probably draw a better picture for the classifiers.

2. In general, all the classifiers perform better on IMDB than on Twitter (best average accuracy on IMDB is 83.85% while on Twitter is only 78.75%). This result meets our assumptions that Twitter texts are more difficult to process. Their language is relatively more free and short than the one used in IMDB.
3. In general, kNN performs relatively bad on both datasets.

Classifier analysis

Figure 1 shows the average accuracy of each classifier across all feature settings, on both datasets. It is clear to see that SVM with deg-2 polynomial kernel performs better than the others on both datasets. Having said that, Naïve Bayes achieves the second best results, which are only slightly worst than SVM-POLY, in spite the assumption of strong dependency of the features. In addition to that, kNN performs much better on Twitter than on IMDB. SVM with linear kernel performs worst than Naïve Bayes.

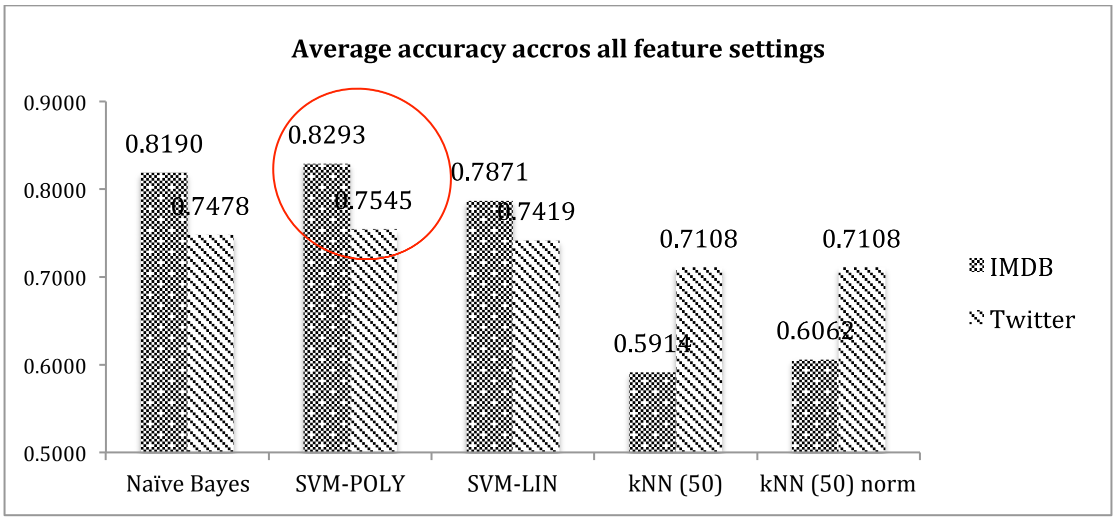

Figure 1 - Average accuracy across all feature settings. kNN is reported only with k=50 here. All the results are reported in Appendix A

Naïve Bayes

Surprisingly, Naïve Bayes performs almost as good as SVM on both datasets. This is a surprise because there is a strong assumption that the features, which are based on words in this case, are dependent somehow and since Naïve Bayes assumes otherwise we expected it would not be as competitive. We used Naïve Bayes by treating the distribution of each feature over every class (i.e. positive and negative) as a multinomial distribution because this was proved to work well with countable data (like words, in our case). To be more specific, given a text d we want to calculate the probability $p(c|d)$ for every class c (in our case: positive and negative) and choose the one that maximizes:

$$argmax_c p(c|d) = argmax_c \frac{p(c)p(d|c)}{p(d)}$$

Now, since $p(d)$ is similar in all classes, we ignore it. d is represented here by a set of features, also known as, "bag of words". The value of $p(d|c)$ is then determined based on a multinomial distribution, so that we get:

$$p(d|c) = \frac{(\sum_i w_i)!}{(\prod_i w_i)!} \prod_i p(w_i, c)^{w_i}$$

where w_i is the weight of feature (e.g. word, lemma) *i*, and $p(w_i, c)$ is the probability that feature *i* occurs in class *c*.
Figure 2 summarizes the accuracies (on the *y* axis) when running on both datasets, with all the different feature settings. On IMDB, Naïve Bayes works best on the words themselves, without further information. However, on Twitter, there is a slight improvement when adding part-of-speech tags as individual features.

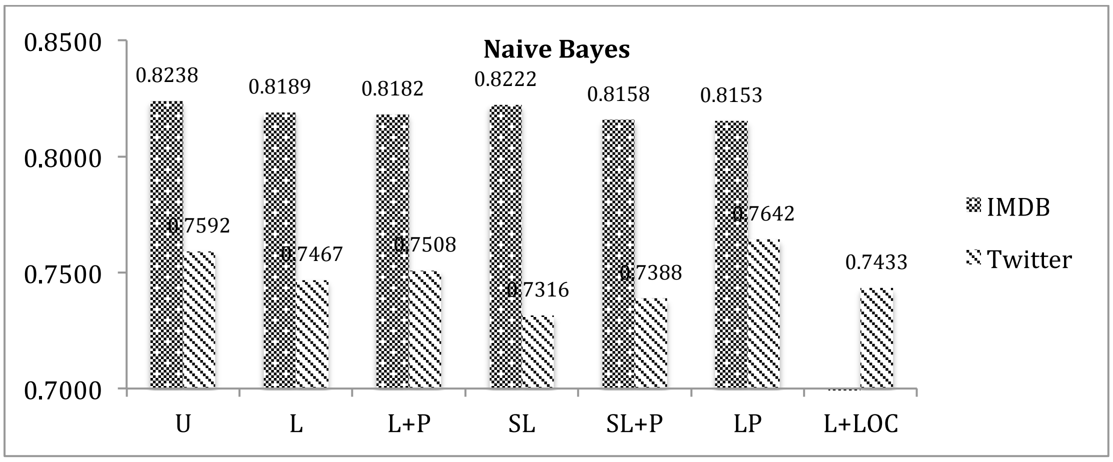

Figure 2 - Naive Bayes accuracy: accuracy on the *x* axis; feature settings on the *y* axis

SVM

When experimenting with SVM we tried two different kernels. The first is a simple linear kernel, multiplying the two vectors directly. The second one is a degree 2 polynomial with additional coefficient 2, i.e. $\left(\frac{1}{N} \cdot \bar{u}^T \cdot \bar{v} + 2\right)^2$, where N is the number of features and $\bar{u}$ and $\bar{v}$ are two vectors. In general SVM with the linear kernel performs a little bit worst than Naïve Bayes on both datasets. On the other hand, when using polynomial kernel, we get the best results on both datasets. This result stands in line with other previous results on text categorization and sentiment analysis. Figure 3 summarizes all SVM results.

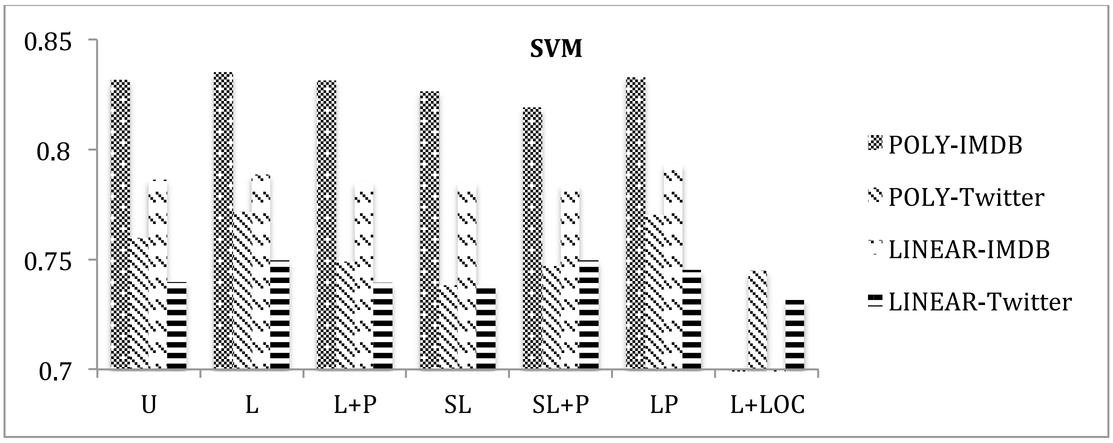

Figure 3 - SVM 10-fold cross validation average accuracy: accuracy on the *x* axis; feature settings on the *y* axis

KNN

kNN was used with Euclidean distance with and without normalization (norm indicates normalization). Figure 4 and 5 summarize the average accuracy results, across all feature settings, over a growing number of *k*, running on Twitter and IMDB respectively. As you can see, in general, the accuracy gets better as *k* grows larger. However, at some point, the accuracy remains the same. Interestingly, as opposed to SVM and Naïve Bayes, in this case the classifier performs better on Twitter rather than on IMDB. Another observation is that most of the curves look actually similar and hence hidden by one another.

We also see that the normalization does not change the performance of the classifier when using feature presence (FP) and feature frequency (FF). However when using TF-IDF the normalization does changes the accuracy. We observe that TF-IDF running without normalization performs worst than all other options.

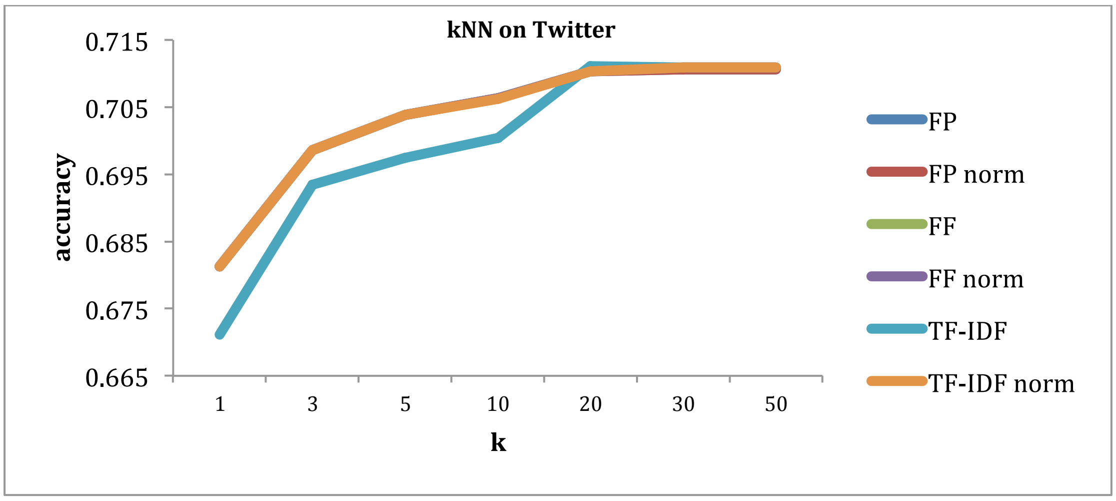

Figure 4 – kNN accuracy over all feature settings, running on **Twitter**: accuracy on the *y* axis; *k* on the *x* axis

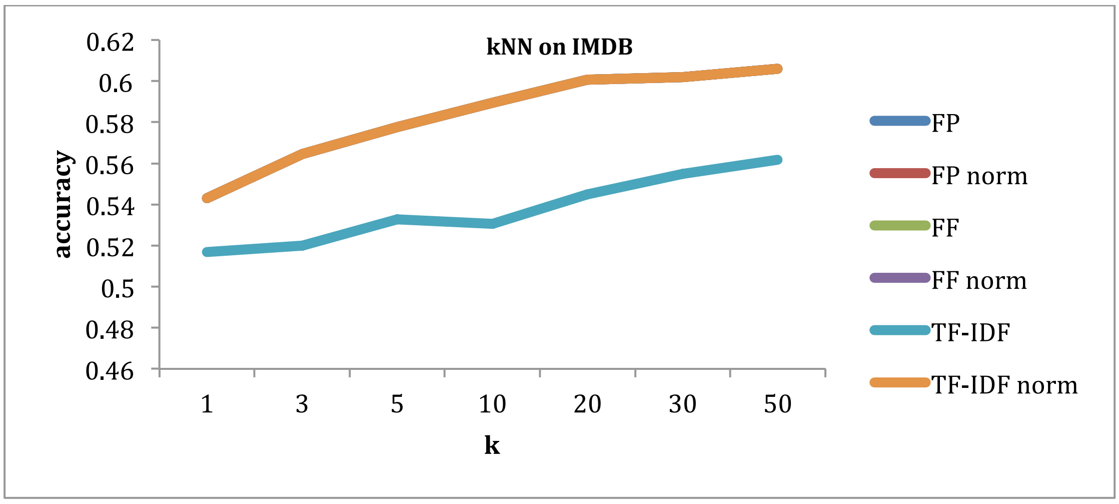

Figure 5 – kNN accuracy over all feature settings, running on **IMDB**: accuracy on the *y* axis; *k* on the *x* axis

We also tested the classifier on the training data. Figures 6 shows the average accuracy results running on the training data on both datasets IMDB and Twitter. Like before, some of the curves look similar and hence are hidden by one another. In general, we see that as expected, the accuracy decreases, as the number of neighbors grows larger. However, with TF-IDF, the accuracy starts relatively low, even when using only one neighbor. It indicates that when using TF-IDF there is a better chance for a specific example to match other examples as neighbors, increasing generalization and hence the accuracy on the test set.

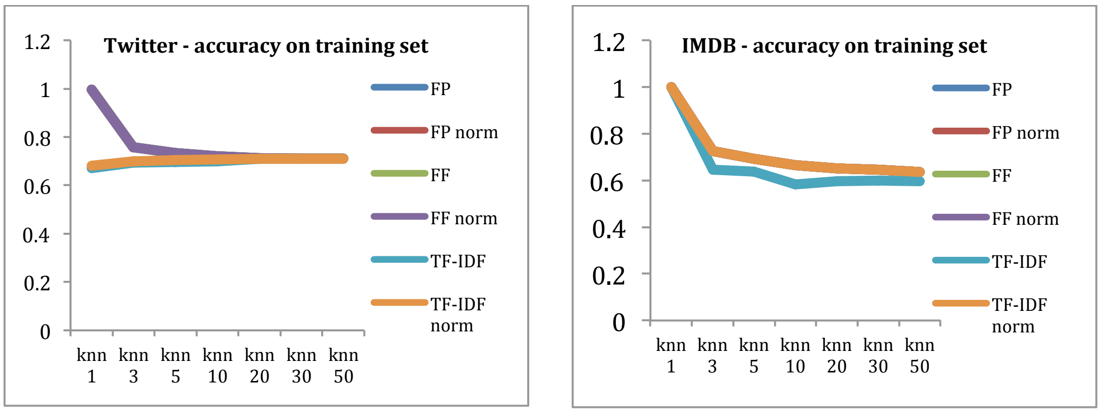

Figure 6 – kNN average accuracy on training set, over all feature settings, running on Twitter (left) and IMDB (right): accuracy the *y* axis; *k* on the *x* axis

Features settings analysis

Here we try to check which of the feature settings contribute most overall. In order to do that we calculate the average of every feature among all the three weighting functions and check how well it performs comparing to the unigram setting in which we consider only the words themselves. Figure 7 summarizes the results on IMDB only. Each setting is represented by one chart, compared to the unigram setting, with all the classifiers we used in our experiments. We marked the places where the feature contributes to the overall accuracy with a circle. We can clearly see that by using lemmas, even only by themselves, we get better accuracy than using the words. However, when we add more information to the lemmas (e.g. part-of-speech tags) or artificially select only some of the lemmas (in the SelectedLemma settings), the accuracy decreases all over the board. When we add part-of-speech tags as individual features, the accuracy remains better than using only unigrams.

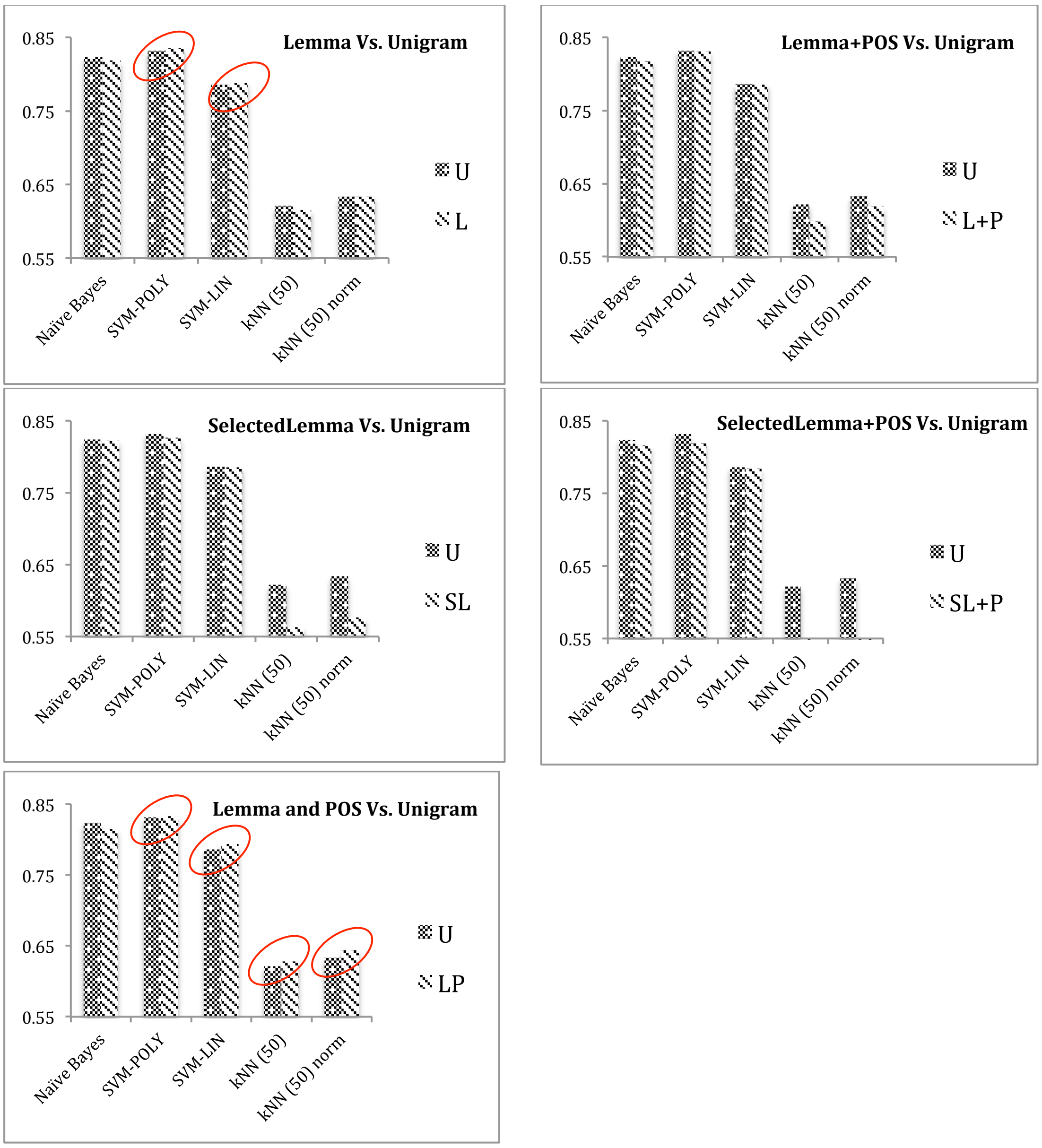

Figure 7 – The contribution of each feature-setting to the overall accuracy. Every chart shows the contribution of one setting compared to the unigram setting, running on the IMDB dataset

Conclusions

In this project we have investigated the task of sentiment analysis as a classification problem. We have used two different datasets: IMDB movie reviews and Twitter statuses that differ mostly in their typical length of each document, and the language style. We believe we have shown that using simple word-based features, it is harder to classify Twitter statuses, comparing to IMDB

reviews. Some other previous published works reported quite similar results on IMDB, for instance Pang and Lee (2002) reported on 82.9% of accuracy using 3-fold cross-validation SVM based on lemmas.
We believe that more work has to be done in order to learn other reliable features (synonyms, negations, named entities, etc.) so that one can use them to improve the accuracy when running on Twitter statuses.

Appendix A

Tables 4, 5, 6 show the results of running on IMDB, 10-fold cross-validation with weighting functions FP, FF, and TF-IDF respectively.

Features setting	**Naïve Bayes**	**SVM-POLY**	**SVM-LIN**	**kNN (3) norm**	**kNN (5) norm**	**kNN (10) norm**	**kNN (20) norm**	**kNN (30) norm**	**kNN (50) norm**
U	0.823	0.8275	0.7855	0.5625	0.582	0.6029	0.6275	0.6264	0.6249
L	0.8229	**0.8385**	0.7855	0.5445	0.5725	0.5889	0.6060	0.6245	0.6390
L+P	0.82	0.837	0.784	0.5549	0.5870	0.6040	0.6054	0.6120	0.6155
SL	0.825	0.834	0.7885	0.5285	0.5385	0.5410	0.5469	0.5710	0.5770
SL+P	0.8174	0.8275	0.7849	0.5155	0.5225	0.5355	0.5360	0.5315	0.5250
LP	0.817	0.837	0.7905	0.5524	0.5850	0.5940	0.6150	0.6390	0.6300

Table 4 - Average accuracy results on IMDB, using 10-fold cross validation, with feature-presence (FP)

Features setting	**Naïve Bayes**	**SVM-POLY**	**SVM-LIN**	**kNN (3) norm**	**kNN (5) norm**	**kNN (10) norm**	**kNN (20) norm**	**kNN (30) norm**	**kNN (50) norm**
U	0.8234	0.8325	0.7844	0.5625	0.582	0.6029	0.6275	0.6264	0.6249
L	0.8225	0.8344	0.7815	0.5445	0.5725	0.5889	0.606	0.6245	0.639
L+P	0.821	0.8345	0.782	0.5549	0.5870	0.6040	0.6054	0.6120	0.6155
SL	0.824	0.8244	0.7885	0.5285	0.5385	0.5410	0.5469	0.5710	0.5770
SL+P	0.8174	0.822	0.7875	0.5155	0.5225	0.5355	0.5360	0.5315	0.5250
LP	0.817	0.8315	0.792	0.5524	0.5850	0.5940	0.6150	0.6390	0.6300

Table 5 - Average accuracy results on IMDB, using 10-fold cross validation, with feature-frequency (FF)

Features setting	**Naïve Bayes**	**SVM-POLY**	**SVM-LIN**	**kNN (3) norm**	**kNN (5) norm**	**kNN (10) norm**	**kNN (20) norm**	**kNN (30) norm**	**kNN (50) norm**
U	0.825	0.835	0.789	0.5625	0.582	0.6029	0.6275	0.6264	0.6249
L	0.8114	0.8324	0.799	0.5445	0.5725	0.5889	0.606	0.6245	0.639
L+P	0.8135	0.823	0.79	0.5549	0.5870	0.6040	0.6054	0.6120	0.6155
SL	0.8175	0.821	0.778	0.5285	0.5385	0.5410	0.5469	0.5710	0.5770
SL+P	0.8125	0.8085	0.7795	0.5155	0.5225	0.5355	0.5360	0.5315	0.5250
LP	0.8119	0.8305	0.798	0.5524	0.5850	0.5940	0.6150	0.6390	0.6300

Table 6 - Average accuracy results on IMDB, using 10-fold cross validation, with TD-IDF

Tables 7, 8, 9 show the results of running on Twitter, 10-fold cross-validation with weighting functions FP, FF, and TF-IDF respectively.

Features setting	**Naïve Bayes**	**SVM-POLY**	**SVM-LIN**	**kNN (3) norm**	**kNN (5) norm**	**kNN (10) norm**	**kNN (20) norm**	**kNN (30) norm**	**kNN (50) norm**
U	0.7635	0.7564	0.7482	0.6709	0.6972	0.7039	0.7031	0.7098	0.7109
L	0.7556	0.7654	0.7533	0.6952	0.7043	0.7047	0.7066	0.7105	0.7109
L+P	0.7635	0.7462	0.7415	0.6721	0.6988	0.7098	0.7113	0.7105	0.7109
SL	0.7501	0.736	0.7427	0.6964	0.7058	0.7066	0.7117	0.7109	0.7109
SL+P	0.7533	0.7431	0.7549	0.6925	0.7062	0.7133	0.7113	0.7109	0.7109
LP	0.7847	0.7639	0.7517	0.6796	0.6941	0.7023	0.7109	0.7098	0.7098
L+LOC	0.7545	0.7427	0.7305	0.6623	0.6839	0.6862	0.6894	0.7101	0.7101

Table 7 - Average accuracy results on Twitter, using 10-fold cross validation, with feature-presence (FP)

Features setting	**Naïve Bayes**	**SVM-POLY**	**SVM-LIN**	**kNN (3) norm**	**kNN (5) norm**	**kNN (10) norm**	**kNN (20) norm**	**kNN (30) norm**	**kNN (50) norm**
U	0.7631	0.7552	0.7521	0.6709	0.6972	0.7039	0.7031	0.7098	0.7109
L	0.7501	0.7674	0.7631	0.6952	0.7043	0.7047	0.7066	0.7105	0.7109
L+P	0.7494	0.7462	0.749	0.6721	0.6988	0.7098	0.7113	0.7105	0.7109
SL	0.7317	0.7345	0.7454	0.6964	0.7058	0.7066	0.7117	0.7109	0.7109
SL+P	0.7431	0.7439	0.7592	0.6925	0.7062	0.7133	0.7113	0.7109	0.7109
LP	0.7682	0.7635	0.76	0.6796	0.6941	0.7023	0.7109	0.7098	0.7109
L+LOC	0.7454	0.7396	0.7356	0.6623	0.6839	0.6862	0.6894	0.7101	0.7109

Table 8 - Average accuracy results on Twitter, using 10-fold cross validation, with feature-frequency (FF)

Features setting	**Naïve Bayes**	**SVM-POLY**	**SVM-LIN**	**kNN (3) norm**	**kNN (5) norm**	**kNN (10) norm**	**kNN (20) norm**	**kNN (30) norm**	**kNN (50) norm**
U	0.7509	0.7682	0.7192	0.6709	0.6972	0.7039	0.7031	0.7098	0.7109
L	0.7345	0.7827	0.7321	0.6952	0.7043	0.7047	0.7066	0.7105	0.7109
L+P	0.7396	0.7545	0.7274	0.6721	0.6988	0.7098	0.7113	0.7105	0.7109
SL	0.7129	0.7443	0.7266	0.6964	0.7058	0.7066	0.7110	0.7109	0.7109
SL+P	0.72	0.7541	0.7345	0.6925	0.7062	0.7133	0.7113	0.7109	0.7109
LP	0.7396	**0.7875**	0.7243	0.6796	0.6941	0.7023	0.7109	0.7098	0.7109
L+LOC	0.7301	0.7525	0.7286	0.6623	0.6839	0.6862	0.6894	0.7101	0.7109

Table 9 - Average accuracy results on Twitter, using 10-fold cross validation, with TF-IDF

References

Pang B., L. Lee and S. Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques". In *EMNLP '02: Proc. of the ACL-02 conf. on Empirical methods in natural language processing*, pages 79–86. ACL, 2002

Russell Stuart J. and Peter Norvig. 2003. "Artificial Intelligence: A Modern Approach (2 ed.). Pearson Education". *p. 499 for reference to "idiot Bayes" as well as the general definition of the Naive Bayes model and its independence assumptions*

Jason Rennie, Lawrence Shih, Jaime Teevan, David Karger , Tackling the Poor Assumptions of Naïve Bayes Text Classifiers, presentation slides(http://cseweb.ucsd.edu/~elkan/254/NaiveBayesForText.pdf)