Sentiment Analysis of Movie Review Comments

6.863 Spring 2009 final project

Kuat Yessenov
kuat@csail.mit.edu

Saša Misailović
misailo@csail.mit.edu

May 17, 2009

Abstract

This paper presents an empirical study of efficacy of machine learning techniques in classifying text messages by semantic meaning. We use movie review comments from popular social network Digg as our data set and classify text by subjectivity/objectivity and negative/positive attitude. We propose different approaches in extracting text features such as bag-of-words model, using large movie reviews corpus, restricting to adjectives and adverbs, handling negations, bounding word frequencies by a threshold, and using WordNet synonyms knowledge. We evaluate their effect on accuracy of four machine learning methods - Naive Bayes, Decision Trees, Maximum-Entropy, and K-Means clustering. We conclude our study with explanation of observed trends in accuracy rates and providing directions for future work.

1 Introduction

Internet today contains a huge quantity of textual data, which is growing every day. The text is prevalent data format on the web, since it is easy to generate and publish. What is hard nowadays is not availability of *useful* information but rather extracting it in the proper *context* from the the vast ocean of content. It is now beyond human power and time to seed through it manually; therefore, the research problem of automatic categorization and organizing data is apparent.

Textual information can be divided into two main domains: facts and opinions. While facts focus on objective data transmission, the opinions express the sentiment of their authors. Initially, the research has mostly focused on the categorization of the factual data. Today, we have web search engines which enable search based on the keywords that describe the *topic* of the text. The search for one keyword can return a large number of pages. For example, Google search for the word "startrek" finds more than 2.3 million pages. These articles include both objective facts about the movie franchise (e.g. Wikipedia article) and subjective opinions from the users (e.g. review from critics).

In recent years, we became witnesses of a large number of websites that enable users to contribute, modify, and grade the content. Users have an opportunity to express their personal opinion about specific topics. The examples of such web sites include blogs, forums, product review sites, and social networks.

Opinion can be expressed in different forms. One example may be web sites for reviewing products, such as Amazon [1], or movie review sites such as RottenTomatoes [4] which

enable rating of products, usually on some fixed scale as well as leaving personal reviews. These reviews tend to be longer, usually consisting of a few paragraphs of text. With respect to their length and comprehensiveness they tend to resemble blog messages. Other type of web sites contain prevalently short comments, like status messages on social networks like Twitter [5], or article reviews on Digg [2]. Additionally many web sites allow rating the popularity of the messages (either binary thumbs up/thumbs down or finer grained star rating), which can be related to the opinion expressed by the author.

Sentiment analysis aims to uncover the attitude of the author on a particular topic from the written text. Other terms used to denote this research area include "opinion mining" and "subjectivity detection". It uses natural language processing and machine learning techniques to find statistical and/or linguistic patterns in the text that reveal attitudes. It has gained popularity in recent years due to its immediate applicability in business environment, such as summarizing feedback from the product reviews, discovering collaborative recommendations, or assisting in election campaigns.

The focus of our project is the analysis of the sentiments in the short web site comments. We expect the short comment to express succinctly and directly author's opinion on certain topic. We focus on two important properties of text:

1. subjectivity – whether the style of the sentence is subjective or objective;
2. polarity – whether the author expresses positive or negative opinion.

We use statistical methods to capture the elements of subjective style and the sentence polarity. Statistical analysis is done on the sentence level. We apply machine learning techniques to classify set of messages.

We are interested in the following questions:

1. To what extent can we extract the subjectivity and polarity from the short comments? What are the important features that can be extracted from the raw text that have the greatest influence on the classification?
2. What machine learning techniques are suitable for this purpose? We compare in total four techniques of supervised and unsupervised learning.
3. Are the properties of short messages important for sentiment analysis similar to the properties of some existing corpus? We compare our manually annotated corpora to the larger existing corpus

We present the analysis on manually annotated examples from Digg. We describe the experiments and interpret the results.

2 Methodology

Our method of sentiment analysis is based upon machine learning. We explain what sources of data we used in 2.1, how we selected features in 2.2, and how we performed classification in 2.3.

2.1 Sources

We chose the domain of social web site comment messages. We obtained the comments from articles posted on Digg. Digg [2] is a social networking web site which enables its users to submit links and recommend the content from other web sites. Digg has a voting system which allows users to vote for (+1) or against (-1) posted items and leave comments on posts. The total sum of *diggs*, that is the difference between thumbs up votes and thumbs down votes, represents the *popularity* of the post. Besides popularity, which is assigned by other users, there is no clue about the sentiment of the author of the messages.

We have chosen two relatively popular posts from Digg. Both articles share the theme; they are about movie reviews of recent blockbuster movies:

1. `http://digg.com/movies/Quantum_of_Solace_disappoints` : a review of new James Bond movie "Quantum of Solace" (684 diggs);

2. `http://digg.com/movies/Star_Trek_The_best_prequel_ever` : a review of "Star Trek" movie (669 diggs.)

We have retrieved all comments from these posts and stored them in the original format in files `qos.txt` and `startrek.txt`.

The reason we have chosen movie reviews is that they provide good material for analyzing subjectivity and opinions of the authors. Movie reviews have been used before for sentiment analysis. We expect that comments express the same range of opinions and subjectivity as the movie reviews. The main difference between the movie reviews and Digg comments is length of the text. Typical comment is only one or couple of sentences short, and is usually narrowly focused on a single claim made in the article. Movie reviews tend to be more focused on the plot and the impression about the movie. However, we would expect the subjective expression of both to be the same.

Sentiment analysis technique can be performed either at the document level, or sentence level [11]. In this project we assume that the sentiment of the whole message is expressed as the *sum of sentiments* of each individual sentence. This model proves to be correct in most of our examples. This model was successful due to the brevity of the messages. Indeed, many messages have only one or a couple of sentences. In some cases we experienced that the sentences in the message were of the opposite polarity.

Since the messages were not annotated by their authors, we manually graded them. The grades are discrete, denoting only the presence of the properties. Each sentence was graded for its subjectivity – either *subjective* or *objective*, and for its polarity – either *positive*, *negative*, or *neutral*. In this phase, we followed the guidelines outlined in [13]. We created a simple parser that converts text in the form that allows giving the grade to each sentence (`parsedig.py`.) The annotated sentences were checked independently by both authors, and the decisions were made by consensus. The graded sentences are stored in `qos.txt.out` and `startrek.txt.out`.

2.1.1 Corpora Properties

"Quantum of Solace" (QoS) corpus contains comments on a review of the new James Bond movie. The review assumes a critical position towards the movie. The participants in the discussion express both agreement and disagreement with the article and show both positive

and negative towards the movies itself. The discussion becomes tense at a moment, but without longer flames.

"StarTrek" (ST) article is a short neutral review of the movie, and commentary on its relation to the old franchise. Discussion revolves around the question whether the movie is up to the high standard of the old franchise. Considering there are many fans of the series among participants, many negative opinions about the changes in the plot from the old movie are expressed. Sometimes, the discussion goes off-topic and shifts towards prequels in general ("God Father" in particular), and failure of another recent blockbuster "Watchmen".

Text	Msgs	Sents	Sents/Msg	Subj/Obj	Pos/Neg/Neutr
Quantum of Solace (QoS)	110	262	2.38	110 / 152	62 / 84 / 116
Startrek (ST)	49	169	3.43	61 / 107	34 / 68 / 66

Table 1: The properties of the annotated documents. *Msgs* – number of messages in a corpus, *Sents* – number of sentences in a corpus, *Sents/Msg* – number of sentences per message, *Subj/Obj* – number of sentences annotated as subjective and objective, *Pos/Neg/Neutr* – number of sentences annotated as positive, negative or neutral, respectively

From table 2.1.1 we can see that the messages, indeed, are short. In QoS corpus, there are only around two sentences per message, while ST corpus is a bit more verbose having 3.5 sentences per one post. We can see the relative disbalance of sentences annotated as subjective and objective. This might seem counterintuitive. Our explanation is that many ambiguous sentences, for whose subjectiveness we didn't have a definite clue, were marked as objective. In addition to that, sentences were generally reviewed without regards to their context (by reading long comments backwards, for example.) Example ambiguous sentences include:

1. The same people who are like, "OMG THIS MOVIE ROCKS! I SEEN IT 14 TIMES!", are saying, "That movie was meh. It was borderline fail.", a few months later.

2. 1) Watch the documentary FLOW, out on DVD soon, see how serious the water supply situation can get thanks to actual companies doing what is described in this film, and 2) typing spoilers in your comment is an amateur move.

Even greater disbalance exists when considering sentences that show negative emotion versus positive emotion. In ST corpus, there are almost twice as many negative sentences as there are positive sentences. We expect that such disbalance may reflect unjustly on the classification performance. Partly, the reason is that both movies come as sequels to existing media product, and thus, cause disappointment from old fans.

2.2 Feature Selection and Extraction

In order to perform machine learning, it is necessary to extract clues from the text that may lead to correct classification. Clues about the original data are usually stored in the form of a *feature vector*, $\vec{F} = (f_1, f_2, \ldots f_n)$. Each coordinate of a feature vector represents one clue, also called a *feature*, f_i of the original text. The value of the coordinate may be a binary value,

indicating the presence or absence of the feature, an integer or decimal value, which may further express the intensity of the feature in the original text. In most machine learning approaches, features in a vector are considered statistically independent from each other.

The selection of features strongly influences the subsequent learning. The goal of selecting good features is to capture the desired properties of the original text in the numerical form. Ideally, we should select the properties of the original text that are relevant for the sentiment analysis task. Unfortunately, the exact algorithm for finding best features does not exist. It is thus required to rely on our intuition, the domain knowledge, and experimentation for choosing a good set of features.

In this section we discuss the possible candidates for good features that are applicable to sentiment analysis. In section 3 we present the evaluation of different selection techniques on our test examples.

2.2.1 Bag-of-words Model

Bag-of-words is a model that takes individual words in a sentence as features, assuming their conditional independence. The text is represented as an unordered collection of words. Each feature of the vector represents the existence of one word. This is effectively a *unigram* model, where each word is conditionally independent from the others. All the words (features) in the feature vector constitute the dictionary. The challenge with this approach is the choice of words that are appropriate to become features.

Using this model the sentence *This is a great event* may be represented by the following feature vector:

$$\vec{F_0}=\{\texttt{'a':1, 'event':1, 'great':1, 'is':1, 'this': 1}\}.$$

(Here we represent the feature vector as a python dictionary; NLTK, for example, uses this representation of a feature vector.) This would be a satisfactory representation if that single sentence was only one in the whole corpus. If we want to be able to represent other sentences sentences, for example *It is a great Startrek movie*, the previous feature vector would not be a good representative. It is thus required to extend the set of words, and incorporate them as the features in the feature vector. The set of features in this case would be

$$\{\texttt{'a', 'event','great', 'is', 'it', 'movie', 'Startrek', 'this'}\}.$$

Feature vectors that fully represent both sentences would be (for the first sentence; similar feature vector is created for the second sentence):

$$\vec{F_1} = \{\texttt{'a' : 1, 'event': 1,'great' : 1, 'is' : 1, 'it' :0,}$$
$$\texttt{'movie' : 0, 'Startrek': 0, 'this' : 1}\}$$

Only some of the words appear in both sentences, and they are used for expressing the *similarity* between the sentences. Obviously, for any real use, the feature vector would have to contain a much larger number of words. We will explore some of the choices for selection of words that are suitable for sentiment analysis.

It is possible to register either the presence of the word appearance in some text, or the frequency – the number of times the word appeared. The frequency in the feature vector for sentence *I really really enjoyed the movie* for word "really" would have value 2 (number of word appearances.) This may indicate the extent of the sentence polarity on a finer grained scale. However, since we compare single sentences, it not very common to have one word appearing multiple times. Furthermore, previous research have shown that for sentiment analysis the mere presence or absence of the word have the same performance as the more detailed frequency information [12]. For that reason, we have chosen the appearance of the word as feature vector values in subsequent experiments.

Ideal bag-of-words feature vector would contain all the words that exist in the language. It represents de facto a dictionary of the language. However, this model would not be practical for at least three reasons. One reason is model complexity, since the model would capture more information than required. It would represent training corpus of text ideally,but it would overfit to it as well and lead to bad performance when exposed to new examples. Additionally, computational complexity of the subsequent learning (for e.g. one million elements long vector) is tremendous. Finally, if the two previous obstacles were overcome, handling new words is still not possible. Languages are very dynamic, and new words are invented often, especially in the Internet community.

Sentiment of the author is often expressed via certain words and phrases. For example, in the sentence *This was a great event.* the word "great" is the best indicator of the author's opinion. One approach that could be imagined is to manually select the most important keywords (such as *great*, *excellent*, *terrible* when we want to express polarity of a sentence) and use them as the features. However, Pang et al. [12] show that manual keyword model is outperformed by statistical models, where a good set of words that represent features are selected by their occurrence in the existent training corpus. The quality of selection depends on the size of the corpus and the similarity of domains of training and test data. Using statistical trends from different domains, that don't have the same desired properties as the original domain, may lead to inaccurate results; for example, if subjectivity analysis is done on a set of sentences from newspaper reports, where most of the sentences were written in objective style.

It is important to create the comprehensive dictionary (feature vector) which will capture most important features both in training set and in previously unseen example. We evaluate two different approaches for selection of features in bag-of-word model. Both are based on the selection of the most frequent words in a text corpus. One approach is to use the elements from *the same domain*. We will divide each text into two pieces. One piece will be used as the known set, one for the training purposes. Note, however, that the variance may be large due to the small corpus that we have. Another approach assumes using set of features based on word frequencies in *existing corpus, which is similar by topic and by sentiment*. An example of such corpus is the movie review corpus described in [10]. Advantage of such choice of features is the ability for better comparison of new messages. Messages may belong to other articles on a given web site.

The selection of all words that appear in corpora may lead to overfitting, analogous to the case of selecting all words in a language, described previously. Other risks mentioned there apply as well. Certain bounds must be set to constrain the size of the feature vector. In our experiments we consider two approaches to constraining the size of the feature vector. Both consider the words that appear most number of times. In one scenario, we directly

bound the size of the feature vector, and *select the words that appeared most number of times.* In the second scenario, we select all words which have the frequency *over certain threshold.* Removing infrequent words may lead to improvement in the performance of the classification. The influence of the words that are not mapped in the feature vector may, however, be indirectly encoded by additional feature `UNKNOWN`, which would represent either the presence or the frequency of unknown words in a sentence. Additionally, it is possible to remove some of the existing frequent words via black listing. Common practice for search engines is to remove words such as *a*, *the*, *do*, ... which bring little useful information.

This model is simple, and it has several limitations. Limitations include the inability to capture the subjectivity/polarity relations between words, distinguishing between parts of speech, inability to handle the negation, and different meanings of one word.

An extension of the model that may appear natural and straightforward would be including pair of words (bigrams) as features instead of unigrams. However, we did not evaluate this model for two reasons. First reason is the sparsity of our hand-crafted corpus. Second reason is the result presented in [12] which doesn't show the advantage of bigrams over unigrams in sentiment analysis. The experiments in this paper used the movie review corpus that is now incorporated in NLTK.

2.2.2 Handling Syntactic and Semantic Properties

Bag-of-words model does not capture the relations between the words. For example, it will consider the two sentences *I saw a fantastic movie* and *I saw an excellent film* as two quite different sentences. The similarity between words *fantastic* and *great*, as well as words *movie* and *film* is obvious to the human reader. It is apparent that each of these pairs synonym words may be represented by a single feature. We modified the model so that feature vector has only one word representing a synonym cluster. The features then become semantic similarity rather than exact word match. For every word present in a sentence, we check whether there is a feature word that is synonymous to at least one sense of the word. We marked the presence of the feature if there is one.

WordNet [9] is a lexical database, which contains the relations between similar words. The relations include synonym, hyponym, hypernym, and so on. We used primarily synonymity of words, and looked at synonym closures of words in text. In addition to that, WordNet provides path similarity measure between sense which is a numerical value that tells how close two words are by their meaning. This way we could produce a numerical values for features telling how close words are to the feature word meanings. Unfortunately, WordNet does not include many words from the movie reviews, but given information from WordNet we can handle some interesting cases.

Some parts of the speech may give more information about polarity of the sentence. Adjectives and adverbs are often good clues about the opinion of the author. Examples include phrases as *a nifty plot* or *acted the role vehemently*. We evaluated the performance of the classifier when only adjectives and adverbs are considered as features. We manually checked whether the tagged words in our corpus get the correct part-of-speech tag. In most cases the tags were indeed correct. Additionally, filtering out the personal names (e.g. names of actors, movies...) may influence the classification, especially when making transition between multiple texts.

2.2.3 Handling Negation

Negation plays an important role in polarity analysis. One of the example sentences from our corpus *This is not a good movie* has the opposite polarity from the sentence *This is a good movie*, although the features of the original model would show that they are. Words that are influenced by the negation, especially adjectives and adverbs should be treated differently. This involves both the feature selection and the extraction from the new sentences. On a level of feature selection, a simple, yet effective way [8] for the support of the negation is to include additional feature *[word]-NOT*, for each adjective and adverb. On a level of extraction the feature values from new sentences, one way to support negation in the sentence is to perform full parsing of the sentence. This approach is both computationally expensive and may be inaccurate due to the lack of the tagged corpora for training. Alternative method is *chunking* the sentence according to some criterion. We applied basic chunking for our corpus. The results show that this technique can yield an improvement to the classification. Although at this moment we support only a small number of patterns, which handle adjectives and adverbs, it would be possible to create more extensive set of rules that would match nouns and verbs instead.

2.3 Classification

Classification algorithm predicts the label for a given input sentence. There are two main approaches for classification: supervised and unsupervised. In supervised classification, the classifier is *trained* on a labeled examples that are similar to the *test* examples. Contrary, unsupervised learning techniques assign labels based only on internal differences (distances) between the data points. In classification approach each sentence is considered independent from other sentences. The labels we are interested in this project are (1) subjectivity of the sentence and (2) polarity of the sentence.

We consider three supervised – Naive Bayes, Maximum Entropy and Decision Trees, and one unsupervised classification approach – K-Means clustering. All four algorithms are available in NLTK framework [3] [7].

2.3.1 Supervised Learning

Naive Bayes assumes that all features in the feature vector are independent, and applies Bayes' rule on the sentence. Naive Bayes calculates the prior probability frequency for each label in the training set. Each label is given a likelihood estimate from the contributions of all features, and the sentence is assigned the label with highest likelihood estimate.

Maximum Entropy classifiers compute parameters that maximize the likelihood of the training corpus. They represent the generalization of Naive Bayes classifiers. The classifier apples iterative optimizations, that find local maximum. The start state is initialized randomly. They are run multiple times during the training to find the best set of parameters.

Decision trees create a flowchart based classifier. At each level it utilizes decision stumps, a simple classifiers that check for the presence of a single feature. The label is assigned to the sentence at the leaf nodes of the tree.

Supervised learning techniques divide the data corpus into two groups – *training set* and *test set*. The training of the classifier is done on the sentences from the training set. The

quality of the training is later evaluated on the sentences from the test set. In order to decrease the bias of particular choice of training and test data, common practice is to perform the cross-validation. The corpus is divided into N groups (called *folds*). The classification process is repeated N times, where data from one group is used for testing, and other data is used for training. The result of classification is the mean of results for single folds. Classification has the greater confidence if the result from each fold give similar results, i.e. the resulting variance is small. We performed the 10-fold cross-validation on our examples.

Accuracy is the simple measure for the evaluation of classifier. It is the relation between the sentences that were correctly classified and all the sentences in the test set. The accuracy, however, may give flawed results. If the number of e.g. objective sentences is much larger than the number of subjective, then if we skip training and assign all test data label "negative", the accuracy will still be very high, despite the obviously faulty classifier. We also used B^3 (B-Cubed) metric [6] that was initially used for evaluation of unsupervised classifiers. B^3 takes into consideration both precision and recall. Their combination is called F-score (it is in fact the harmonic mean or precision and recall). F-score can be weighted to give more penalty on either precision or recall. We used unweighted version, that treats both precision and recall equally. We observed the values of accuracy and and B^3, and found that both give similar results in our experiments for the classification, but B^3 imposes somewhat higher penalty on misclassified example sentences. Finally, we used the accuracy in our evaluation results, for the comparison with previous papers that used this measure.

2.3.2 Unsupervised Learning

K-Means tries to find the natural clusters in the data, by calculating the distance from the centers of the clusters. The position of centers is iteratively changed until the distances between all the points are minimal. The centers are initially randomly assigned. K-Means can find only local maximum, and the final label assignment can be suboptimal. Common practice is to repeat the algorithm on the same data multiple times, and to report the best result. We have repeated the procedure 10 times in our experiments. We have used Euclidean distance as dissimilarity metric between feature vectors. We use B^3 measure to evaluate the performance of the classifiers.

3 Evaluation

3.1 Experimental Set-up

We implemented a Python program that performs the experiments and handles the results, based on the description from section 2. The program is available here `http://mit.edu/~kuat/www/6.863/sentiment.py`. It uses NLTK 0.9.9 including its movie review corpus, machine learning facilities, and WordNet bindings. Specifically, we use packages `nltk.*`, `nltk.cluster`, `nltk.classify`, `nltk.corpus.wordnet`, and `nltk.corpus.movie_reviews`. We also used numerical library for python `numpy` and `optparse` for parsing of command line parameters. Additionally, we used `matplotlib` package for automatic drawing of plots.

All our experiments were run on Ubuntu Linux 9.04 installed on a commodity work station.

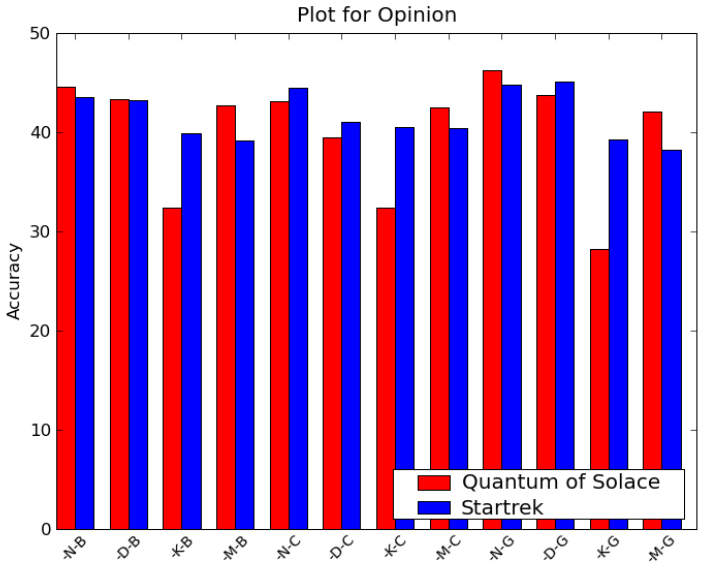

Figure 1: Classification by Polarity

3.2 Comparison of Machine Learning Algorithms and Feature Extractors

In our first experiment we consider the relation between supervised and unsupervised classification techniques as well as relation between different choices of the feature extractor. We measured accuracy of automated classification for each corpus and each label using 10-fold cross validation.

We considered four choices of classification technique – Naive Bayes (N), K-means Clustering (K), Decision Trees (D), and Maximum-Entropy using the default algorithm (M). For feature extractors, we used plain bag-of-words (B), bag-of-words using frequencies from the movie reviews corpus (C), and bag-of-words using only adjectives and adverbs and accounting for negation (G).

Figure 1 shows results for classification by the opinion – positive, negative, or neutral. Since there are three choices for the label, the base line is 33%. For QoS corpus, Naive Bayes and max-entropy perform the best at about 45% accuracy. For ST corpus, Naive Bayes is the winner at about 45% accuracy rate. Maximum-entropy does not perform as well as for the other corpus. In combination, Naive Bayes is the most effective classification technique with Max-Entropy and Decision Trees coming next close to it. These results lead us to the conclusion that the distinction of polarity of comments when there are neutral comments as well may be harder problem than simpler binary polarity analysis.

It is interesting to consider only two-value classification for opinions. Figure 2 shows the plot of each technique applied only to sentences that are either negative or positive, but not neutral. The results here is that both Naive Bayes and Decision Tree perform almost equally well on both corpora at around 67% accuracy rate. It is remarkable that unsupervised K-Means clustering algorithms and Max-Entropy classification perform almost as well and better on ST corpus as the winners. One experiment (KC) did not complete because of an

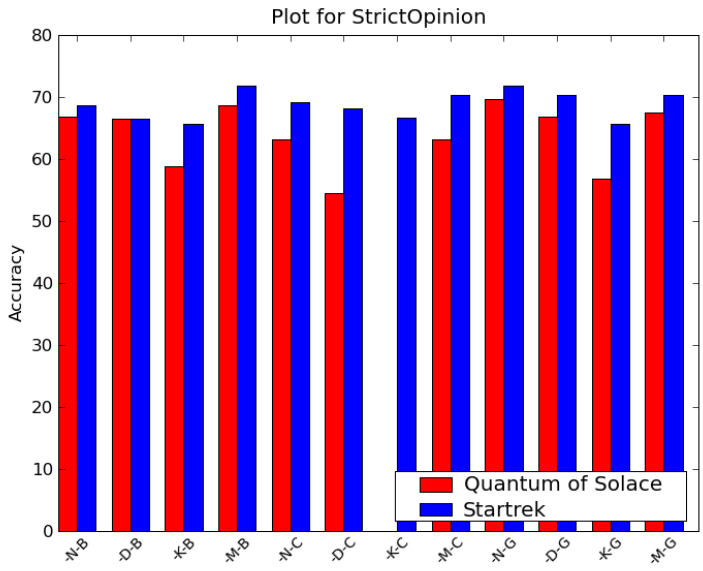

Figure 2: Classification by Strict Polarity

assertion violation in K-Means clustering algorithm in NLTK.

Figure 3 presents results for subjectivity analysis. Accuracy on QoS is noticeably better than on ST on all combinations except for clustering algorithm. For ST there is very little difference between the technique and the feature extractor accuracy rates, and they all are around 60%. For QoS, Decision Trees, Naive Bayes, and Max-Entropy perform very well in range 70-80%.

Our experiments demonstrate that our three choices of feature extractor influence the outcome for the techniques. For example, selecting words from our little domains, may give better results than features selected from word frequencies of the `movie_reviews` corpus. We can see that by comparing bag-of-words (B) with bag-of-words from movie reviews (C) on all three plots. In some cases, such as strict opinion on QoS, the results of having features selected from most popular words from movies reviews are significantly worse. This indicates that the words that are good clues about subjectivity and polarity of the messages appear even in corpora that are very small in size.

The most effective classification technique is Naive-Bayes combined with bag-of-words feature extractor that uses negated words. This combination performs consistently well on all experiments. It is surprising that unsupervised clustering algorithm performs excellent on subjectivity analysis on ST domain.

It is interesting to note the influence of recognition of negation and inclusion of such features the feature vector on the classification results. We can see the improvement in the case of strict polarity classification (figure 2.) In this case, it helps classifiers achieve the best accuracy rates. We would expect greater influence of negation in larger corpus, especially together with related word recognition (that we evaluated in section 3.4.) Negation features perform worse when it is required to recognize neutral sequences. They also don't help in subjectivity analysis. This is even expected – negation is not the property that is required

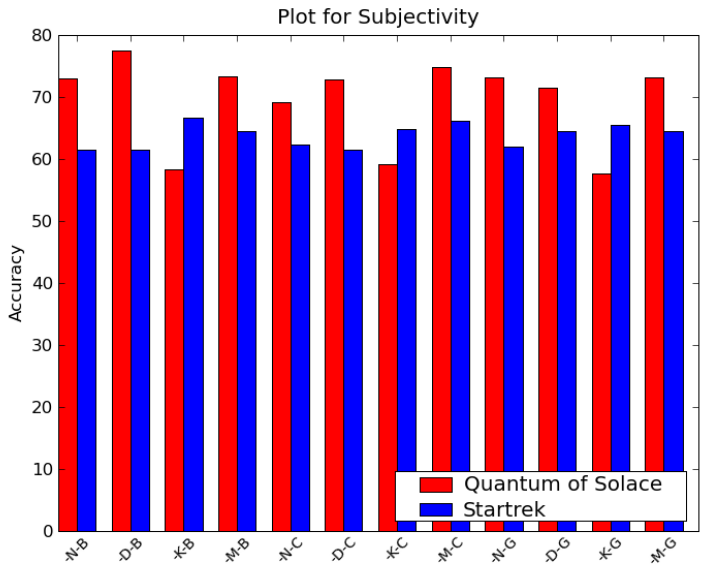

Figure 3: Classification by Subjectivity

for detection of subjective opinion. Its inclusion in feature vectors just increases noise, and decreases the quality of the classifier.

We also performed experiments with features such as inclusion of special field for unknown words. The results don't show significant improvement, but also don't decrease the classification accuracy. We haven't noticed any difference when we used frequency count instead of presence notification. We observed a relatively small decrease in accuracy (1-2%) when we apply removal of usual blacklisted words. We noted that using only certain parts of speech (attributes) in polarity analysis yields almost the same result for Naive Bayes classifier.

However, we also need to say a word of caution. The accuracies between single folds in the cross-validation varied greatly in most of the cases. The intensity of single rounds was diverging up to 15% percents from the mean value. This results implies that although the results are encouraging, it is necessary to have larger corpus that would help produce more accurate classifiers. In section 3.5 we describe the experiment where we use the subset of sentences from the existing `movie_reviews` corpus.

3.3 Effect of Word Frequency Threshold

This experiment is designed to demonstrate the effect of feature vector size on the accuracy rates. It shows how the word frequency in training data can influence the selection of features for the dictionary. We evaluated Naive-Bayes technique on different feature vector selections by imposing a threshold on the feature occurrence. If we consider only the words that appear most frequently in the corpus, we might get better results by avoiding over-fitting as well as adjusting to the corpus style.

Figure 4 shows the effect of maximum threshold on the accuracy rate of Naive Bayes

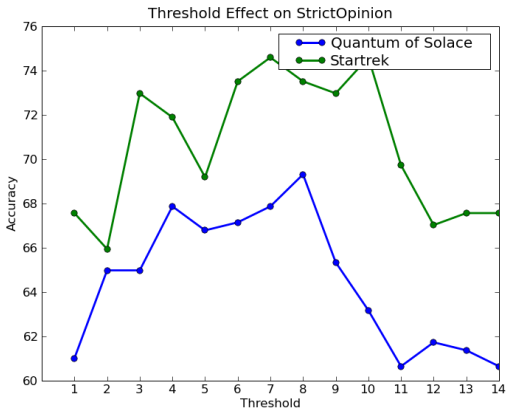

Figure 4: Effect of Threshold on Strict Opinion Classification

algorithm with the simple bag-of-words feature extractor on both inputs in strict opinion analysis. As we can see, there is a certain pick in for which the threshold is optimum. For small corpora, such as our annotated ones, the satisfactory frequency is between 3 and 10. As a comparison, the size of feature vectors in these two cases are 216 and 57 respectively, for corpus QoS. The whole corpus contains 993 different words.

The small threshold means more words in the feature vector. We can see that if we include all words, a noise is introduced, which reduces the accuracy rate. In fact, in case of small threshold, the classifiers overfit to the training data, and perform worse on test data. However, by picking too few words (only the most frequent ones), we may lose the text properties that are required for proper classification. For example, if we use only words that appear more than 10 times in a corpus, the accuracy drops. For larger corpora, like `movie_reviews`, the best accuracy is for the threshold is between 8 and 13 appearances of the word.

3.4 WordNet Analysis

In this analysis, the feature vector consisted of all words that appeared at least 3 times in the corpus. Feature selector was simple bag-of-words selector, and the classifying algorithm was Naive Bayes. The results are shown in table 2.

Text	Naive Bayes without WordNet	Naive Bayes with Wordnet
QoS opinion	64.982 %	66.065 %
ST opinion	72.122 %	69.189 %

Table 2: WordNet analysis

We have observed the increase of accuracy for the QoS corpus, but it drops with ST corpus. We think that the reason for such divergent results may be due to the size of the corpora and relatively small number of words that could be replaced by others in the corpora. Additionally, it is possible for one word to have multiple meanings, being synonym to other

words only in certain cases. All these variations are hard to catch on small training set. It would be necessary to analyze a larger corpus in order to make a definite conclusion.

3.5 Using existing movie_review Corpus for Training

In this experiment we have used the existing `movie_review` corpus to train the classifiers. Classifiers are then applied on the whole corpora QoS and ST. `Movie_review` corpus contains tags for positive and negative sentences. This corpus makes experiment suitable only for evaluation of strict polarity analysis, and only for supervised learning algorithms. We used a subset of 1000 positive and 1000 negative sentences. The feature vector contained 200 most used words.

Text	C+N	C+D	C+M
Quantum of Solace (QoS)	58.219 %	52.740 %	58.904 %
Startrek (ST)	61.765 %	60.784 %	64.706 %

Table 3: Using Movie Review Corpus for Training

The results of the classification are shown in table 3. We initially expected this classification to be better than the one on the small corpora. The performance of the classifiers is not very satisfactory. They achieve at most around 58% of accuracy. This is for more than 10% lower than the training done on our corpora. The number of training sentences is quite larger than the number of test sentence. The inspection of feature vectors reveal the existence of many words that don't appear in the test corpora.

3.6 Threads to Validity

We have shown experimental evaluation of feature selection and classification of sentiments in natural language texts. We have tried to objectively assess the contribution of different factors. However, the conclusions of this project must be taken with precaution. Potential threats to validity include following:

1. size of corpus – we have worked with very small corpora. Number of sentences in each corpus doesn't exceed 300 sentences. Research corpora include multiple thousands of polarized sentences.

2. manually annotated corpora – two corpora were manually annotated by the authors. It is possible that there exist random or systematic error induced in the annotations. This may be especially applicable to the results of the experiment in section 3.5.

3. relation between messages and their sentences – we have assumed that the attitude of the message is represented by the attitudes of its sentences. In case of short messages we believe this is true. However, additional experiments would be necessary to confirm or rule out this hypothesis.

4. topic – we have explored only one type of comments. These are comments on movies. The language in this comments may be specific. We believe that the same technique

may be applied to other kinds of comments, such as product announcements, political debates, etc. To confirm this conjecture, it would be necessary to perform experiments on a broader set of topically distinct messages

5. community – we used comments from only one web site (`Digg.com`). Although unlikely, the community of this web site may be different from the community of some other web sites.

4 Conclusion

In this report, we have analyzed the sentiment of social network comments. We used the comments on articles from Digg as our text corpora. We evaluated the fitness of different feature selection and learning algorithms (supervised and unsupervised) on the classification of comments according to their subjectivity (subjective/objective) and their polarity (positive/negative). The results show that simple bag-of-words model can perform relatively good, and it can be further refined by the choice of features based on syntactic and semantic information from the text. We looked into the influence of feature vector on the classification accuracy. We also observed that existing corpus from apparently similar corpus which contains sentences from movie reviews. Our results show that such corpus, although contains similar polarity of the words, as well as the common topic, may not perform classification well.

The future work includes:

1. performing larger scale experiments using our techniques; we could benefit from having larger data set but unfortunately it requires manual work of tagging sentences with labels;

2. using WordNet path similarity for obtaining numerical features of how close the sentence is to the selected features;

3. analyzing correspondence between number of Diggs (measure of popularity) and the comments;

4. using off-topic and on-topic as labels.

5 Appendix A – The list of command line options

In order to perform the analyses, we have created a Python program `sentiment.py`. This section all options that it supports.

```
Usage: sentiment.py [options] file1 [file2]*

Options:
  -h, --help            show this help message and exit
  -q, --quiet           print status messages to stdout
  -V, --verbose-all     print status messages to stdout
  -s, --subjectivity    Classify sentences based on subjectivity
                        (subjective/objective)
  -o, --opinion         Classify sentences based on opinion (thumbs up/down or
                        neutral)
  -f, --count-frequency
                        Count the frequency of word occurrence in sentence
  -p O_TESTPERCENTAGE, --test-percentage=O_TESTPERCENTAGE
                        Percentage of initial data saved for testing
  -c O_CROSSVALIDATE, --crossvalidation=O_CROSSVALIDATE
                        Crossvalidate (add % of cv. )
  -w, --wordnet         Use wordnet to select synonyms (or near synonyms)
  -x, --strict-opinion  Use only +1 -1 labels for opinions
  -y O_TAG_WORDS_FILTER, --tag-words-filter=O_TAG_WORDS_FILTER
                        Put P-O-S tags that filter feature vectors (e.g.
                        JJ;NN;NNP)
  -t, --tag-sentences   Tag sentences with corresponding P-O-S
  -b, --use-blacklist   Exclude some common words such as I, a, the from the
                        vectors
  -u, --use-unknown     Add a field(s) for unknown words encountered in test
                        sentences
  -i, --use-impersonal  Use personal names in feature vectors
  -m O_MAX_FV_SIZE, --max-feature-vector-size=O_MAX_FV_SIZE
                        Maximum feature vector size (if less than 15 it is
                        threshold for word appearance)
  -B, --bag-of-words    Feature extractor is bag of words
  -C, --corpus          Feature extractor is bag of words, features are
                        selected from existing corpus
  -G, --negation        Feature extractor is bag of words, features are both
                        adjective positive and negations
  -N, --naive-bayes     Perform Naive Bayes classification
  -M CL_MAX_ENTROPY, --max-entropy=CL_MAX_ENTROPY
                        Perform Maximum Entropy classification (provide
                        algorithm: default, GIS, IIS, CG, BFGS, Powell,
                        LBFGSB, Nelder-Mead)
  -D, --dec-tree        Perform decision tree classification
  -K, --k-means         Perform K-Means clustering classification
  -A, --agglomerative   Perform Agglomerative clustering classification
```

References

[1] Amazon online retailer web site. `http://www.amazon.com`.

[2] Digg social networking site. `http://www.digg.com`.

[3] Natural language toolkit. `http://www.nltk.org`.

[4] Rottentomatoes movie review site. `http://www.rottentomatoes.com`.

[5] Twitter social networking site. `http://www.twitter.com`.

[6] A. Bagga and B. Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, 1998.

[7] S. Bird, E. Klein, and E. Loper. *Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit*. O'Reilly Media, 2009.

[8] S. R. Das and M. Y. Chen. Extracting market sentiment from stock message boards. *SSRN*, 2001.

[9] C. Fellbaum. *WordNet: An Electronical Lexical Database*. The MIT Press, Cambridge, MA, 1998.

[10] B. Pang and L. Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *In Proceedings of the ACL*, pages 271–278, 2004.

[11] B. Pang and L. Lee. *Opinion Mining and Sentiment Analysis*. Now Publishers Inc, July 2008.

[12] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. 2002.

[13] J. Wiebe, T. Wilson, and C. Cardie. Annotating expressions of opinions and emotions in language. *Computers and the Humanities*, 39(2-3):165–210, May 2005.