Sentiment Analysis Using Natural Language Processing

PROJECT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Engineering)

TO

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

SUBMITTED BY

ADITYA GOURAV	(2K11/CO/007)
AMAN GUPTA	(2K11/CO/016)
ASHISH YADAV	(2K11/CO/029)
KARTIK SINGAL	(2K11/CO/052)

DEPARTMENT OF COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY,
SHAHBAD DAULATPUR, BAWANA ROAD - 110042 (DELHI)

Sentiment Analysis

Using Natural Language Processing

PROJECT

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(Computer Engineering)

TO

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

SUBMITTED BY

ADITYA GOURAV (2K11/CO/007)

AMAN GUPTA (2K11/CO/016)

ASHISH YADAV (2K11/CO/029)

KARTIK SINGAL (2K11/CO/052)

FACULTY ADVISOR

Mr. Manoj Kumar

Associate Professor, DTU

DEPARTMENT OF COMPUTER ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY
SHAHBAD DAULATPUR, BAWANA ROAD - 110042 (DELHI)

CERTIFICATE

This is to certify that this project report entitled "Sentiment Analysis Using Natural Language Processing" being submitted by Aditya Gourav, Aman Gupta, Ashish Yadav and Kartik Singal at Delhi Technological University in partial fulfilment of the requirement for the award of the Degree of Bachelor of Technology (Computer Engineering) is a record of the students own work carried out under my supervision and guidance. This project work is of desired standard and has not been submitted in any other University or Institution for the award of any other degree.

Mr. Manoj Kumar
Associate Professor
Computer Engineering Department
DTU, Shahbad Daulatpur, Delhi

ACKNOWLEDGEMENTS

- With pleasure, I acknowledge my deep sense of gratitude and indebtedness to my supervisor, Mr. Manoj Kumar, Associate Professor, Department of Computer Engineering, DTU, for his enlightening guidance, intelligent approach, constructive critique, whole hearted and ever available help, which has been the primary impetus behind the research. Without the wise advice and able guidance, it would have been impossible to complete the project in this manner.

- Last, but not least, We would like to thank my parents for their help, encouragement, prayers and cooperation during our dissertation work.

- Above all We owe a debt to “Almighty” with whose kind blessing we have been able to clear another chapter of our life.

ADITYA GOURAV (2K11/CO/007)
AMAN GUPTA (2K11/CO/016)
ASHISH YADAV (2K11/CO/029)
KARTIK SINGAL (2K11/CO/052)

ABSTRACT

In this project, we plan to work on product reviews of various product classes and analyze them for finding the product features and opinion of various customers about those features. Using this analysis we aim to identify feature-wise good and bad aspects of a given product. This can be a useful practical solution to allow customers to help decide how well a product satisfies his/her needs if they are only looking for few important features in a product and don't care about other features.
We plan to use the product information available on marketing sites e.g. www.imdb.com, twitter.com to learn common features for a given product type to help reduce redundancy. Also, we aim to analyze the performance of different feature extraction techniques on different product classes. Opinion mining for specific product features would require some level of semantic understanding to separate opinions about other features mentioned in the same review. Also same word can be used to express contrasting opinions, which must be taken into account to avoid incorrect sentiment classification if only a global polarity is used for each sentiment word.
Based on these challenges, we aim to achieve a robust solution for extracting features and opinion about them from the product reviews. We present the general trends and within we only focus on extracting reviews from online websites about a particular product. Additionally, we present some interesting emerging projects in the contemporary research and point out the key appliances of sentiment analysis.

This project 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.

TABLE OF CONTENTS

INTRODUCTION

Background and Recent Research

Among the 2 main types of textual information - facts and opinions, a major portion of current information processing methods such as web search and text mining work with the former. Opinion Mining refers to the broad area of natural language processing, computational linguistics and text mining involving the computational study of opinions, sentiments and emotions expressed in text. A thought, view, or attitude based on emotion instead of reason is often referred to as a sentiment. Hence, lending an alternate term for Opinion Mining, namely, Sentiment Analysis. This field finds critical use in areas where organizations or individuals wish to know the general sentiment associated to a particular entity - be it a product, person, public policy, movie or even an institution. Opinion mining has many application domains including science and technology, entertainment, education, politics, marketing, accounting, law, research and development. In earlier days, with limited access
to user generated opinions, research in this field was minimal. But with the tremendous growth of the world wide web, huge volumes of opinionated texts in the form of blogs, reviews, discussion groups and forums are available for analysis making the world wide web the fastest, most comprehensive and easily accessible medium for sentiment analysis. However, finding opinion sources and monitoring them over the Web can be a formidable task because a large number of diverse sources exist on the Web and each source also contains a huge volume of information. From a humans perspective, it is both difficult and tiresome to find relevant sources, extract pertinent sentences, read them, summarize
them and organize them into usable form. An automated and faster opinion mining and summarizing 2 system is thus needed. Much before ARPAnet expanded into the World Wide Web, we often asked our friends to recommend a good orist/baker, requested reference letters regarding job applicants from colleagues or to explain who they were planning to vote for in the next local elections or consulted consumer reports to decide what refrigerator to buy. But the Internet and the Web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics that is, people one has never heard of. And conversely, more and more people are making their opinions available to strangers via the Internet. In fact, surveys conducted in the recent times indicate that a large percentage of internet users (bordering 81research for product purchase on the internet atleast once. Out of this number, close to a quarter of them perform this research daily. Among people that look up restaurant, hotel and spa reviews, close to 87% credit internet reviews to influencing their purchase decision. Several consumer surveys indicate that customers are willing to shell out almost 90 of the original price of current 5-star rated items. Apart from this apparently heavy reliance on reviews, a rather high percentage of internet users have posted comments and reviews on products, making the web an ideal place for consumer feedback. Sentiment analysis has created increased interest due to its promising and potential applications.

Related Work

Language Models

Unigrams
The easiest and most used approach. Pang et al. [2] reported an accuracy of 81.0%, 80.4%, and 82.9% for Naive Bayes, MaxEnt and SVM respectively in the movie-review domain. This was found to be closely similar to accuracies obtained in twitter classification which were 81.3%, 80.5%, and 82.2% respectively [4].

Bigrams
Bigrams were found to be a very sparse feature in the Twitter corpus as experienced by Go et al. and caused an overall drop in accuracy for MaxEnt and NB. MaxEnt gave equal probabilities to the positive and negative class for this case because there is not a bigram that tips the polarity in either direction. A better option is to combine bigrams and unigrams.

Unigram + Bigram
Both unigrams and bigrams are used as features. In the movie-review domain, there was a decline for Naive Bayes and SVM, but an improvement for MaxEnt.

POS tagging
POS tags would be a useful feature since usage for several words varied. For example, over as a verb has a negative connotation whereas over as the noun, would refer to the cricket over which by itself doesn't carry any negative or positive connotation. However past experiments with POS tagging in feature extraction for sentiment analysis have yield little improvements. The accuracy improves slightly for Naive Bayes but declines for SVMs, and the performance of MaxEnt is unchanged while classifying movie-reviews with their individual accuracies being 81.5%,81.9% 80.4% respectively. Limiting to adjectives, following up on previous successful efforts in sentiment detection, failed to produce any results while in fact reducing overall accuracies in all 3 methods despite the intuitive idea of expecting adjectives to carry a great deal of information regarding overall sentiment in a document. As is obvious from figure, the 2633 most frequent unigrams are a more optimal choice. These findings were found to be consistent in the area of twitter sentiment analysis as well.

Classifiers for Sentiment Analysis

Among the various machine learning algorithms that have been used for sentiment analysis Naïve Bayes, SVM and MaxEnt have shown promising results in movie-review classification and subsequently in recent Twitter sentiment analysis research. Here we look into detail into these three approaches and compare and contrast their efficiencies in the Twitter domain as found by previous work.

Naive Bayesian Classifier

The Naive Bayesian classifier is a straightforward and frequently used method for supervised learning. It provides a flexible way for dealing with any number of attributes or classes, and is based on probability theory.

Starting by giving a recipe for training a Naive Bayes classifier using just the words as features:

1. Estimate the probability $P(c)$ of each class $c \in C$ by dividing the number of words in documents in c by the total number of words in the corpus.
2. Estimate the probability distribution $P(w \mid c)$ for all words w and classes c. This can be done by dividing the number of tokens of w in documents in c by the total number of words in c.
3. To score a document d for class c, calculate

$$\textbf{score}(d,c) \overset{def}{=} P(c) * \prod_{i=1}^{n} P(w_i|c)$$

4. If you simply want to predict the most likely class label, then you can just pick the c with the highest **score** value. To get a probability distribution, calculate

$$P(c|d) \overset{def}{=} \frac{\textbf{score}(d,c)}{\sum_{c' \in C} \textbf{score}(d,c')}$$

The last step is important but often overlooked. The model predicts a full distribution over classes. Where the task is to predict a single label, one chooses the label with the highest probability. It should be recognized, though, that this means losing a lot of structure. For example, where the max label only narrowly beats the runner-up, we might want to know that.

The chief drawback to the Naive Bayes model is that it assumes each feature to be independent of all other features. This is the "naive" assumption seen in the multiplication of $P(w_i \mid c)$ in the definition of **score**. Thus, for example, if you had a feature best and another world's best, then their probabilities would be multiplied as though independent, even though the two are overlapping. The same issues arise for words that are highly correlated with other words (idioms, common titles, etc.).

Maximum Entropy Classifier

Maximum entropy classifiers are commonly used as alternatives to Naive Bayesian classifier because they do not require statistical independence of the features that serve as predictors. The Maximum Entropy (MaxEnt) classifier is closely related to a Naive Bayes classifier, except that, rather than allowing each feature to have its say independently, the model uses search-based optimization to find weights for the features that maximize the likelihood of the training data.

The features you define for a Naive Bayes classifier are easily ported to a MaxEnt setting, but the MaxEnt model can also handle mixtures of boolean, integer, and real-valued features.

I now briefly sketch a general recipe for building a MaxEnt classifier, assuming that the only features are word-level features:

1. For each word *w* and class $c \in C$, define a joint feature $f(w, c) = N$ where *N* is the number of times that *w* occurs in a document in class *c*. (*N* could also be boolean, registering presence vs. absence.)
2. Via iterative optimization, assign a weight to each joint feature so as to maximize the log-likelihood of the training data.
3. The probability of class *c* given a document *d* and weights λ is

$$P(c|d,\lambda) \overset{def}{=} \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c' \in C} \exp \sum_i \lambda_i f_i(c',d)}$$

Because of the search procedures involved in step 2, MaxEnt models are more difficult to implement than Naive Bayes model, but this needn't be an obstacle to using them, since there are excellent software packages available.

The features for a MaxEnt model can be correlated. The model will do a good job of distributing the weight between correlated features.

Support Vector Machines

An SVM is a kind of large-margin classifier: it is a vector space based machine learning method where the goal is to find a decision boundary between two classes that is maximally far from any point in the training data.

Problem Statement

In this project, given reviews for a product, we essentially identify its features/attributes and then extract opinion about those product features and classify them as positive or negative. We will analyze the opinion expressed for each product feature and observe the good and bad features from a customer's perspective. We will also obtain a list of commonly used features for a given product type.

Objectives

We plan to use the following objectives -

- Product Feature Extraction: We can use an n-gram model to extract noun phrases/words which are usually candidate features for a product. We can also add Wordnet sysnset data to expand on the list of candidate features and also put an appropriate frequency threshold to discard unimportant features.

- Sentiment Analysis: We can use a list of sentiment words already marked as positive and negative and then score each sentence as positive or negative or neutral based on presence of these words in that sentence.

Methodology

Our method of sentiment analysis is based upon machine learning.
We obtain our sources of data (movie reviews) we used from imdb site (www.imdb.com)

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, ~F = (f1; f2; : : : fn). Each coordinate of a feature vector represents one clue, also called a feature, fi 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.

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:

F0={’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, 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
{’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):

F1 ={’a’ : 1, ’event’: 1,’great’ : 1, ’is’ : 1, ’it’ :0, ’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 which doesn't show the advantage of bigrams over unigrams in sentiment analysis.

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.

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.

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.
Based on our literature surveys, we model a Multinomial Naïve Bayes Classifier with Mutual Information feature selection and add-1 smoothing coupled with a Unigram+Bigram approach for feature extraction.

Naive Bayesian Classifier

The Naive Bayesian classifier is a straightforward and frequently used method for supervised learning. It provides a flexible way for dealing with any number of attributes or classes, and is based on probability theory. It is the asymptotically fastest learning algorithm that examines all its training input. It has been demonstrated to perform surprisingly well in a very wide variety of problems in spite of the simplistic nature of the model. Furthermore, small amounts of bad data, or noise, do not perturb the results by much. The Naive Bayesian classification system is based on Bayes rule and works as follows.

$P(c|t) = P(c)*P(t|c)/P(t)$

It assumes each feature is conditional independent to other features given the class. That is, where c is a specific class and t is text we want to classify. P(c) and P(t) is the prior probabilities of this class and this text. And P(t|c) is the probability the text appears given this class. In our case, the value of class c might be POSITIVE or NEGATIVE, and t is just a sentence. The goal is choosing value of c to maximize P(c|t): Where P(wi|c) is the probability of the ith feature in text t appears given class c. We need to train parameters P(c) and P(wi|c). It is simple for getting these parameters in Naive Bayes model. They are just maximum likelihood estimation (MLE) of each one. When making prediction to a new sentence t, we calculate the log likelihood (log P(c) + log P(wi|c)) of different classes, and take the class with highest log likelihood as prediction. In practice, it needs smoothing to avoid zero probabilities. Otherwise, the likelihood will be 0 if there is an unseen word when it making prediction. Using add-1 smoothing is a solution to that problem.

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 B3 (B-Cubed) metric [6] that was initially used for evaluation of unsupervised classifiers. B3 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 B3, and found that both give similar results in our experiments for the classification, but B3 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.

Unsupervised Learning

K-Means tries to find the natural clusters in the data, by calculating the distance from the centres 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 B3 measure to evaluate the performance of the classifiers.

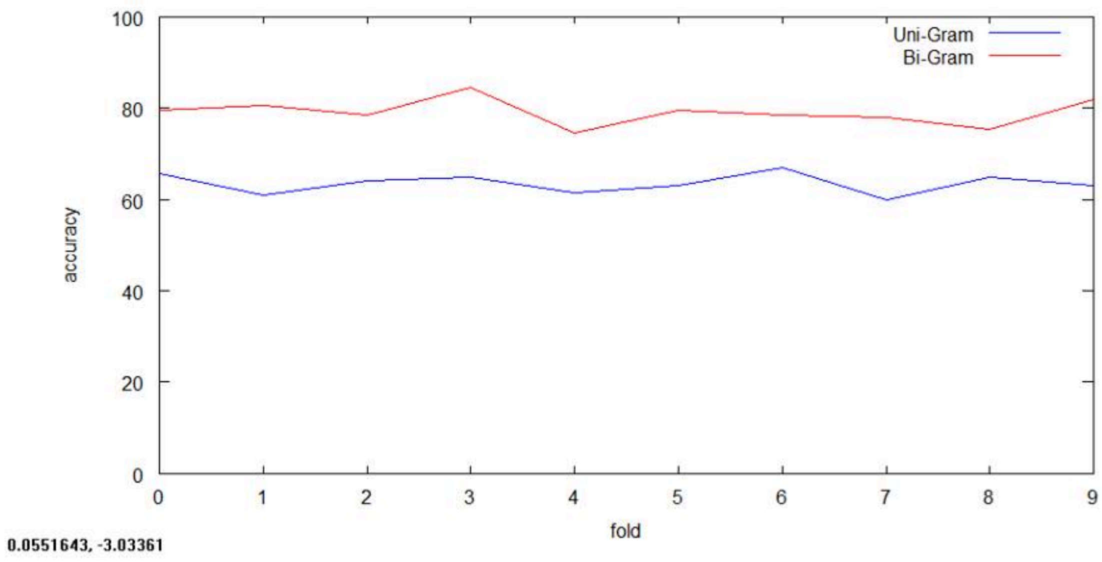
Evaluation

- Evaluation of product feature extraction can be done in comparison with another system such as Hu and Liu's (2004) feature extraction system on the same dataset. Feature extraction can also be evaluated qualitatively to observe whether the most useful features of a product are obtained or not. For this we can manually create list of useful features for target products.

- The datasets have an annotated test set for sentiment analysis which can be used for evaluation of opinion mining task. We can also compare the performance with other systems (Related Papers) using the same dataset.

Results

For Unigram Language model	Average accuracy recorded on used corpus was 63%
For Bigram Language model	Average accuracy recorded was 81%



Comparison between Unigram and Bigram models using Supervised Approach

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.

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.

Sentiment Analysis Applications

This section presents some selected and most prominent use cases for opinion mining techniques. Some have been already related in various research experiments previously mentioned while others still remain a goal to achieve in the future.

- ► Product benchmarking and market intelligence
- ► Advertisement placement
- ► Opinion search and retrieval
- ► Opinion spam detection

Conclusion

In this report, we have analysed the sentiment of social network comments. We used the comments on articles from imdb 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.

Challenges

- Some of the product classes do not have well-defined features like movies, books etc. For such classes, we need to identify implicit features based on what customers liked/disliked about it, which could be something like the movie plot, specific actor's performance etc.
- Currently available datasets usually label an entire review as positive or negative opinion about a product. But even a positive-labeled review can contain negative opinion about one or more product features. This needs to be considered when mining opinion for individual features and evaluating results.
- It would be interesting to identify the phrases within a sentence with positive or negative opinion rather than tagging the whole sentence as positive or negative.

BIBLIOGRAPHY

- ***Google scholar articles on sentiment analysis***
- ***Video Lectures by Christopher Manning and Dan Jurafsky***
- ***Wikipedia***
- ***Apache SVN OpenNLP by ViewVC***
- ***Sentiment Symposium Tutorial on Classifiers by Christopher Potts, Stanford Linguistics***