Sentiment Analysis of Restaurant Review

Do Hoang Giang* do0004ng@e.ntu.edu.sg

Nguyen Vu Tuan† nguy0113@e.ntu.edu.sg

April 21, 2013

Abstract

With the ever-growing availability and popularity of online media such as blogs and social networking sites, Internet is a valuable mine of information for product and service reviews. However, huge volume of opinionated text located in a large number of diverse sources makes the task of extracting and summarizing opinions become formidable for human readers. In this study, we investigate the efficacy of machine learning techniques in classifying opinion from text messages by semantic meaning. We use restaurant reviews from popular social network Yelp.com as our data set and classify text by positive/neutral/negative. We explore different approaches in extracting text features as well as machine learning methods to evaluate their effect on task accuracy. From the result of experiment, we construct a combined feature extractor and make use of Naive Bayes classifier for our final model. We conclude our study with explanation of observed trends in accuracy rates and providing directions for improvement.

1 Introduction

With the ever-growing of availability and popularity of online media, recent years, we became witnesses of a large number of websites that enable users to contribute, modify, and grade content. Users have an opportunity to express their personal opinions about special topics. Websites such as blogs, forums, product review sites, and social networks are valuable mines of information for product and service reviews. However, huge volume of opinionated text located in a large number of diverse sources makes the task of extracting and summarizing opinions become formidable for human readers. *Sentiment analysis* aims to automatically uncover the opinion of the author on a particular from the written text. It uses natural language processing and machine learning techniques to find statistical patterns in the text that reveal attitudes. It has gained popularity in recent years due to its immediate application in business intelligence, recommender systems [8], etc. Sentiment analysis has been widely studied for many years. Extensive research has been conducted on different aspects of this field including feature engineering as well as classification techniques.

*U0920115F, School of Computer Engineering, Nanyang Technological University

†U0920146H, School of Computer Engineering, Nanyang Technological University

Our project focuses on examining the effectiveness of supervised machine learning techniques to the sentiment classification problem. We use statistical methods to capture the elements of text polarity. Sentiment analysis can be conducted at various levels such as word level, sentence level, document level. Our statistical analysis is done on the document level in order to investigate the performance of different approaches in extracting text features and machine learning methods. We present the analysis on manually annotated copora from Yelp as well as describe the experiment and interpret results. Finally, from the obtained result, we report a combined feature extractor and make use of NaiveBayes classifier for our final model. We also produce a confidence score for each of prediction.

2 Data Sources

We chose *Yelp.com* as the domain for data sources.
Yelp.com founded in 2004 is a local directory service with social networking and user reviews. Yelp had an average of approximately 86 million monthly unique visitors as in late of 2012. Each Yelp business listing result contains a 5-point rating, reviews from website visitors, and details such as the business address, hours, accessibility, and parking. Yelp member have written over 36 million local reviews in the website.
We have chosen reviews about House of Prime Rib restaurant, 1906 Van Ness Ave, San Francisco, CA 94109-3008 (Nob Hill). The restaurant serves well-marbled Prime Rib in the English Tradition.
A small web crawler was implemented to obtain data from *Yelp.com*, and its design is described in the following subsection.

2.1 Web Crawler

We have written a small web crawler to retrieve all reviews for the restaurant and stored them in the XML format.
Web Crawler is written in Java with the support of *jsoup* library for parsing HTML. Our Web Crawler first receives an initial URL, retrieves the HTML content from this URL as well as extracts the next URL. Package *ir.sentiment.crawler* consists of 3 classes:

1. *WebCrawler.java* provides general methods for retrieving a web page from an URL address as well as parsing the content of that page based on HTML tag.

2. *YelpCrawler.java* (*extends WebCrawler*) provides specific methods to obtain the review content from *Yelp.com*. It contains two main methods:

   - *getAllReviews*(*String urlPath*) return a list of reviews in this URL (including date, time, rating, author, and review content).
   - *getNextPageUrl*(*Document document*) return a string which is the link for the next review page.

3. *XMLDumper.java* provides method to convert a list of review into a file in pre-defined XML format.

2.2 Corpora Properties

We have retrieved 3328 reviews consists of different ratings (Fig. 1). While the rating

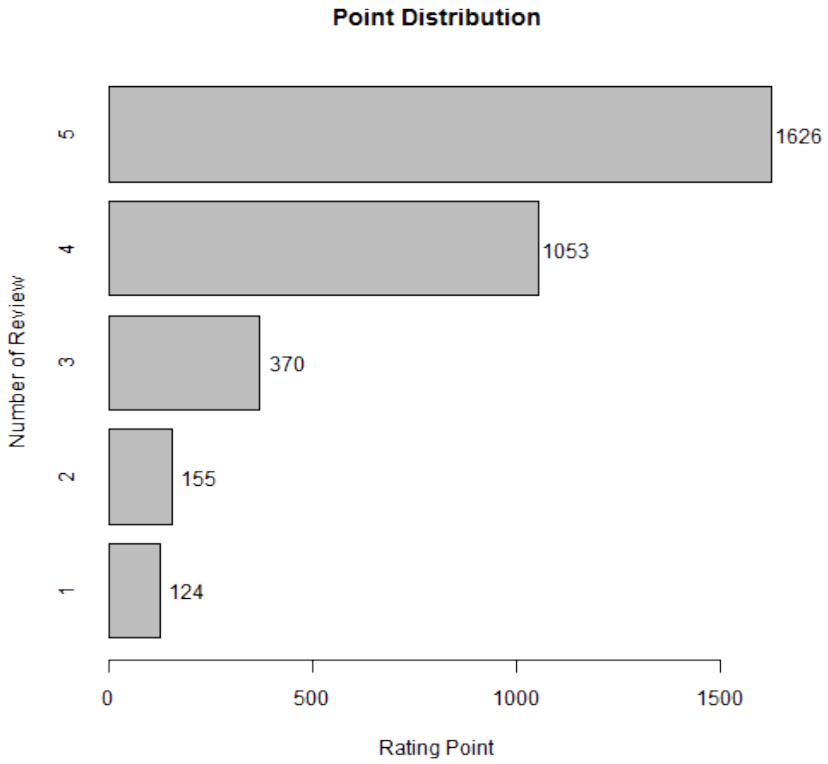

Figure 1: Review Point Distribution

given by the user may not be a perfect indicator for the polarity of the review (positive or negative), it often has a strong correlation with this review' s property. We manually explored some reviews and found that:

- 5-star reviews are mostly positive
- 4-star reviews can be often both positive and neutral
- 3-star reviews can be both positive, neutral and negative
- 2-star and 1-star reviews can be both neutral and negative

Based on that observation, in order to get a balance polarity set, we have filtered out the corpus to get 1000 reviews with: 114 one-star reviews, 122 two-star reviews, 197 three-star reviews, 219 four-star reviews, and 348 five-star reviews. For training purpose, we have labeled 300 reviews consist of positive, neutral and negative classes with 100 of each type.

We note that in our training data a neutral review is the one that contains a balance of positive and negative comments. Traditionally, a neutral one should not express any attitude toward the object. Our labeling method makes the task extremely challenging because of the fuzzy boundary between positive and neutral reviews as well as between neutral and negative ones.

3 Methodology

Sentiment analysis can be performed at different levels: document level, sentence level, attribute level. In this project, due to the constraint of limited training size, we only consider the statistical analysis on the document level.

3.1 Feature Extraction

In this project, we investigate the effect of different feature sets on the classification task. Various features are examined and discussed in the following subsections.

3.1.1 Unigrams

Our *Bag-of-words* model takes individual words in a sentence as features, assuming their appearances conditional independent. The text is represented as an unordered vector of words. All the words in the collection constitute the dictionary. Ideal *Bag-of-words* model would contain all the words that exist in the language. However, this model would not be practical due to model complexity as well as the dynamic change of the language. In our implementation, we enforced a threshold for unigram presence in which only words appearing in more than one review are considered.

3.1.2 Bigram

An extension of the model that may appear natural and straightforward would be including higher order of n-grams such as bigrams or trigrams as features instead of unigrams. The performance of the two models appears to be a matter of some debate. In [5], the authors reported that unigrams outperform bigrams when classifying movie reviews by sentiment polarity, whilst Dave and his colleagues in [3] found that in some settings, bigrams and trigrams yield better product-review polarity classification.
In this experiment, we considered bigram as one of the features to be explored.

3.1.3 Stop Words

Articles such as 'a' and 'the' and pronouns such as 'he' 'they' and 'I' often provide little or no information about sentiment. Therefore, we initially constructed a small list of 174 words as a stop word list. Based on the result of preliminary results of experiments, we further refined the list to include more task-irrelevant words such as proper nouns (location names). All the words in the stop-words list are not included in our words vector space model. There is a only exception for bigram features because the phrases "I love" would have a very different meaning with "They love".

3.1.4 Stemming and Lemmatization

Stemming is a method for collapsing distinct word forms. This could help reduce the vocabulary size, thereby sharpening one's results, especially for small data sets. In our implementation, we used Porter algorithm to perform stemming. The Porter stemmer is one of the earliest and best-known stemming algorithms. It works by heuristically identifying word suffixes (endings) and stripping them off, with some regularization of

the endings.
However, one problem with Porter stemmer is that it often collapses sentiment distinctions, by mapping two words with different sentiment into the same stemmed form. For example, the two words 'objective' (positive) and 'objection' (negative) are mapped to the same form - 'object'.
Understanding the problem created by stemming, we also made use of lemmatization. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. The two may also differ in that stemming most commonly collapses derivationally related words, whereas lemmatization commonly only collapses the different inflectional forms of a lemma.

3.1.5 Part Of Speech

Part-of-speech (POS) information is commonly exploited in sentiment analysis and opinion mining. One simple reason holds for general textual analysis, not just opinion mining: part-of-speech tagging can be considered to be a crude form of word sense disambiguation. We look to the Stanford Log-Linear Part-of-Speech Tagger [7] for all our tagging needs. In our project, we considered POS tags as one of our features for the word vector space model.
Besides that, 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. Turney [9] proposed to detect document sentiment mostly based on adjectives or adverbs. In our implementation, we evaluated one set of feature including only adjectives and adverbs.

3.1.6 Sentiment Dictionary Words

In one of early work [5], the authors showed that a manual keyword model is outperformed by statistical model. Hence, it is important to create a comprehensive dictionary which will capture most relevant features both in training set and in previously unseen example. We evaluated this approach by filtering only words which appear in a predefined dictionary. We used a list of sentiment word which was compiled by Minqing Hu and Bing Liu starting from their previous work [4]. Other recommended word list is SentiWordNet [10] which contains 86994 words - each with three sentiment scores: positivity, negativity, objectivity.

3.1.7 Negation

Negation plays an important role in polarity analysis. One of the example sentences from our corpus: "This is not good" has the opposite polarity from the sentence "This is good". Words that are influenced by the negation should be treated differently. A simple, yet effective way [2] for the support of the negation is to include additional feature word"-[NOT]", for each negated unigram. A negated unigram is any word between a negation word (not, isn't, didn't, etc.) and the first punctuation mark following the negation word.

We implemented negation handling using regrex. A negation is any word matching the following regular expression:

$$
\begin{array}{l}
(?: \\
\quad \text{\^{}}(?: never|no|nothing|nowhere|noone|none|not| \\
\qquad havent|hasnt|hadnt|cant|couldnt|shouldnt| \\
\qquad wont|wouldnt|dont|doesnt|didnt|isnt|arent|aint \\
\quad)\$ \\
) \\
| \\
n't
\end{array}
$$

A clause-level punctuation mark is any word matching the following regular expression:

$$\text{\^{}}[.:;!?-,]\$$$

3.2 Weighting Scheme

Weighting Scheme has traditionally been important in standard IR. The most common scheme used in traditional topic-based text classification is frequency counts which associate with the counts of individual words (unigram) or n-grams.
The TF-IDF weighting scheme from information retrieval may be applied too. As in traditional text classification, these features have been shown highly effective for sentiment classification as well.
In [5], the authors obtained better performance using presence rather than frequency. That is, binary-valued feature vectors in which the entries merely indicate whether a term occurs (value 1) or not (value 0) formed a more effective basis for review polarity classification than did real-valued feature vectors in which entry values increase with the occurrence frequency of the corresponding term. In our experiment, we explore both of these three weighting schemes.

3.3 Classification

Classification algorithm predicts the label for a given input review. There are two main approaches for classification: supervised and unsupervised. In supervised classification, the classifier is trained on labeled examples that are similar to the test examples. Our project is focusing on supervised techniques. We considered five algorithms: Naive Bayes, MaxEnt (Logistic Classification), Support Vector Machine, and Random Forest. All five algorithms are available in Weka package.
For Naive Bayes, we made use of Multinomial Model rather than Multivariate Bernoulli. In contrast to the multivariate Bernoulli event model, the multinomial model captures word frequency information in documents. In order to avoid the "zero frequency" problem, Weka implementation performs Laplace smoothing estimation technique.
In case of Support Vector Machine, we used LibLINEAR Weka implementation in which the kernel type is the original dot product. For Random Forest, we used 100 random trees for classification, whilst for MaxEnt algorithm, we used default Logistic Classification implementation of Weka.

4 Evaluation

We performed the 10-fold cross-validation on our experiment and take results to analysis. In 10-fold cross-validation, The corpus is divided into 10 groups (called folds). The classification process is repeated 10 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.
We implemented a Java program that extracts features for the copora we retrieved from Yelp.com as described in section 2. The features are extracted based on the description in section 3.1. We made use of Standford Parser library for tokenization and POS-tagging. Additionally, we used Jama library for implementing weighting scheme. Finally, Weka 3.7.9 are employed for machine learning facilities.
All our experiment were run on Window 8 Pro installed on Core(2) Duo CPU - 2.66GHz.

4.1 Comparison of Feature Extractors and Learning Techniques

Our first experiment considers the relation between different choices of feature extractor as well as several of supervised machine learning techniques. We considered 9 set of features consisting of bigram, unigram, dictionary words, unigram with POS-tagging, adjectives and adverbs (tagged word features) and the last four sets with negation handling. All of these feature sets are weighted by frequency count (other weighting schemes are compared in latter experiment). Table 1 shows the F-measure weight results for 300 labeled reviews (10-fold cross validation).

	Naive Bayes	MaxEnt	SVM	Random Forest
Unigram	0.700	0.615	0.675	0.563
Unigram with Negation	0.713	0.633	0.680	0.607
Bigram	0.707	0.627	0.674	0.580
Dict.Word	0.693	0.595	0.635	0.613
Dict.Word with Negation	0.718	0.702	0.659	0.619
POS	0.678	0.594	0.679	0.612
POS with Negation	0.702	0.633	0.680	0.615
Tagged Word (adj. & adv.)	0.691	0.594	0.663	0.646
Tagged Word with Negation	0.711	0.619	0.668	0.647

Table 1: F-Measure Result for Different Feature Sets and Learning Methods

For our particular corpus, Naive Bayes performs at the best with about 0.7 in F-measure score. Max-Entropy and Support Vector Machine (linear transformation) come next close to Naive Bayes. Our feature vector size ranges from 300 to 2000 (after filtering out stop words) which can help explain why Naive Bayes is the most efficient classifier. Our experiment demonstrates that recognition of negation has a big influence on the task. With negation handling, the result of classification is greatly improved. Moreover, other advanced features do not show any significant improvement compared to the simple unigram model. This fact can be explained by the spoken English language of

the corpus. Details about that problem are described in the section 6 .
Comparing to the previous works of Pang and Lee [5] where they obtained around 80% accuracy rate, we came to a conclusion that distinction of polarity of comments where there are three classes may be harder than simpler binary polarity analysis.

4.2 Effect of Weighting Scheme

In this analysis, we compared the effectiveness of three weighting scheme according four learning methods. The feature vector for this experiment consists of unigrams stemmed by Porter algorithm. Only stemmed unigrams that appear at least 3 times in the training corpus are included. Figure 2 shows the effects of different schemes on the classification task measured by F-measure score.

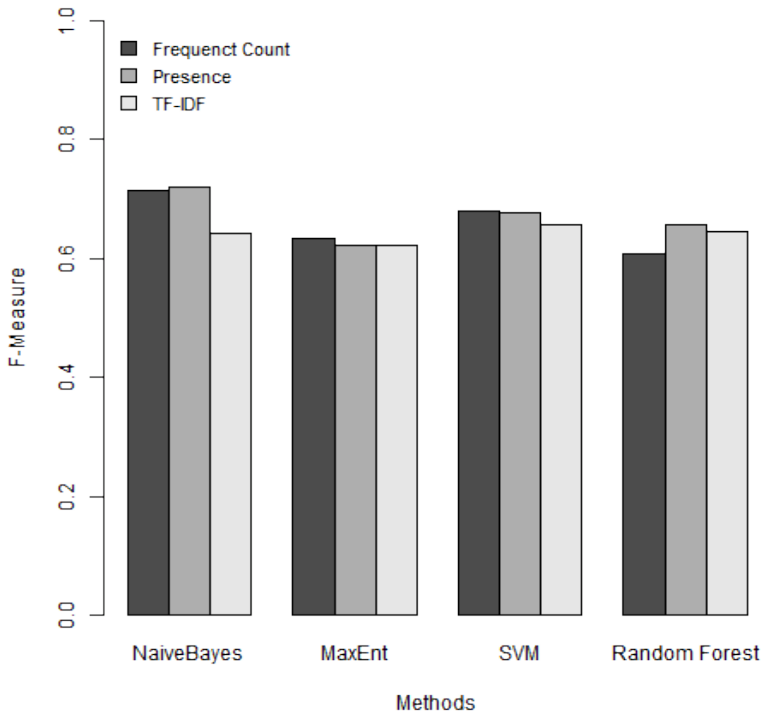

Figure 2: Weighting Scheme Comparison

Our result shows that there are not too much different between presence (binary) weighting scheme and frequency. Besides that, TF-IDF performs worse compared with the other two methods. The reason can be explained by the simple language of corpus. After filtering out the common stop words, most sentiment words (good, best, horrible, etc.) frequently appears in the remaining short content of the reviews. Hence, the TF-IDF scheme gives small weights for such sentiment words, whilst other noisy words (misspelled words) have much bigger weights. This fact leads to the low performance of TF-IDF.

4.3 Effect of Word Presence 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 bag-of-words. We evaluated 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.
In this experiment, we used lemma unigram and NaiveBayes classifier to compare the effect of different word presence threshold on the classification tasks. Figure 3 shows the result the experiment.

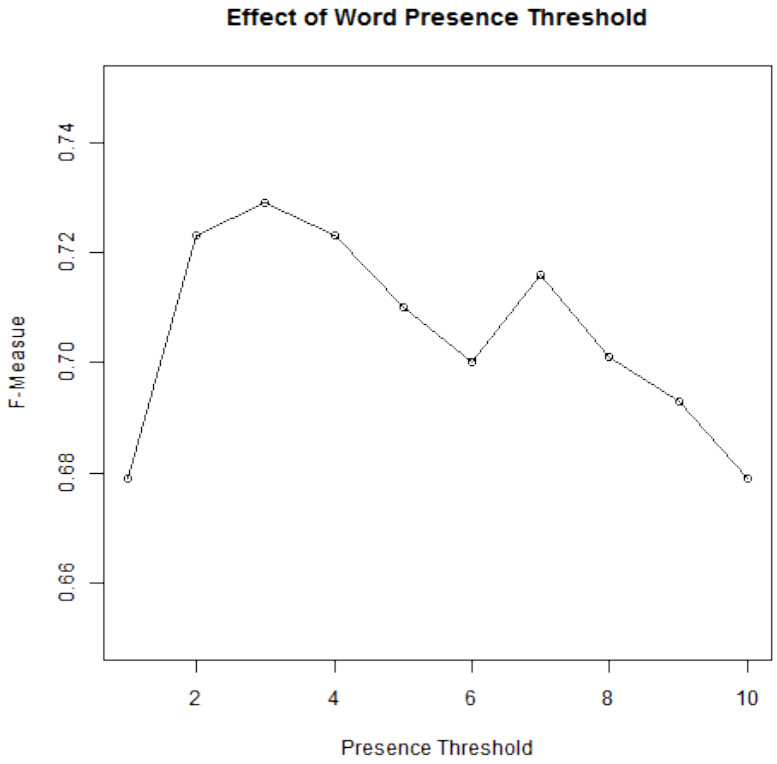

Figure 3: Effect of Word Presence Threshold

The result shows that the acceptable range for word presence threshold is from 2 to 8.

5 Final Model

After examining the combination of different sets of feature as well as various thresholds for word appearance, we decided our final feature set is unigram with negation handling for at least 6 times presence in different training documents. We also introduced two more relevant-task features which are the numbers of positive words and negative words in a review. The positive and negative words are taken from a pre-defined dictionary, in particular case, we used the word list of Minqing Hu and Bing Liu [5]. These features

briefly capture the information that a sentiment dictionary provides. Exploring the word space of unigram, we found that most of the obtained words are adjectives and adverbs. Moreover, by using unigram only, we can avoid the problem produced by incorrectly POS tagging for spoken English. Finally, we manually modify our stop word list in order to include more task-irrelevant words such as the restaurant names and the location names.
Table 2 shows the F-measure scores of different classification techniques on our final feature selection.

Naive Bayes	MaxEnt	SVM	Random Forest
0.774	0.714	0.701	0.669

Table 2: F-Measure Result

In order to reinforce our claim about neutral class, we explored the confusion matrix created by NaiveBayes (10-fold cross validation), as shown in table 3. The results show that the neutral class is the most misclassified class. In our submitted version,

	Negative	Neutral	Positive
Negative	74	23	3
Neutral	14	67	19
Positive	1	7	92

Table 3: Confusion Matrix for Naive Bayes

we made use of NaiveBayes multinomial classifier as our classification algorithms. For each unlabeled review, we reported one of three numbers: -1(negative), 0 (neutral), or +1(positive) as well as a *confidence score* for that prediction. In case of Naive Bayes, we computed *confidence score* for each prediction base on the following formula:

$$conf(c_j|d) = \frac{p(c_j|d)}{\sum_{k=1}^{|C|} p(c_k|d)}$$

However, we note that using the posterior probabilities $p(c_j|d)$ computed by Naive Bayes for *confidence scores* is inappropriate because they are usually tend to go to zero or one exponentially with document length. In [6], the author replaced the posterior scores with the *KL-divergence scores* to construct *confidence score* for NaiveBayes classification.
In order to assess the performance of our model in practice, we random examined some test examples and found that our classifier's accuracy is very promising.

6 Discussion

We have shown the result for our experiment. The result shows that for simple feature extraction, 10-fold cross validation obtained around 0.68%-0.7% F-measure score. Our final model performed quite better than that. Moreover, we also found that neutral class is the most difficult class for classifying. In this section, we first try to explain our obtained results as well as provide suggestion for improvements.

6.1 Result Interpretation

Experiment results reveal that negation handling improves the classification result significantly. Unigram features appear to be the best choice for feature selection in our specific data. The reason for the performance of other features sets can be explained by the used language of the reviews. Exploring the content of the reviews, we found that most of the authors use spoken English instead of formal written form. Some sentences which provide sentiment information contain just one word e.g. "Delicious!"; the bigram features cannot capture that interesting information. Moreover, due to the grammatical issues of the spoken English, POS tagging cannot perform correctly leading to the problem for POS tagging and tagged words features. For example, the sentence "Tender, juicy, cuts like butter.... mmmm!": two words "tender" and "juicy" are tagged as nouns. After all, as explained above, because we used a general dictionary for sentiment dictionary word features, we cannot obtain some interesting features for our particular type of sentiment analysis e.g. "juicy, "deliciousness, etc. We also lost the information from the word not in dictionary like "wooow" which also attributes a lot of relevant information.
Considering the weighting scheme, presence and frequency weighting scheme outperforms TF-IDF. Explaining for the bad performance of TF-IDF scheme, looking at the bag-of-word, we found that most of reviews (after filtered out stop words) only contain common sentiment words like "good", "bad", etc. This fact leads to the low weight for such sentiment words in TF-IDF weighting scheme.
Finally, the result reveals that NaiveBayes is the best classifier for our scenario. This result clearly comes from out small corpora in which the whole size of unigram bag-of-word is about 2000 words stemmed token. NaiveBayes often performs well for such small feature space.

6.2 Threads to Validity

We have shown experiment evaluation of feature selection and classification of sentiment in restaurant review domain. However, the conclusions of this project must be taken with precaution as there are potential threads of validity:

1. size of corpus - we have worked with very small data. Our labeled set consists of only 300 reviews, whilst the research corpora should include multiple thousands of pieces of text.

2. manually annotated corpus - the data including labeling and stop word list was manually annotated by the authors. It is possible that there exists random or systematic errors which may be applicable to the results of the experiment.

3. domain - we have explored only one domain of reviews: restaurant review. Moreover, only one restaurant was examined. The language for this type of review may be specific. In order to confirm that the same technique can be applied for different topics, it would be necessary to perform experiments on a broader set. After all, we used data set from only one web site (Yelp.com), although unlikely the community of this web site may be different from others.

6.3 Improvements

There are a number of things that may be done in the future to improve classification. Below we discuss three methods can improve our results.

6.3.1 Subjective Analysis

One important tool in improving sentiment analysis is subjectivity analysis. Textual information can be divided into two main domains: facts and opinions. While facts focus on objective data transmission, describe that which can be observed by other individual, the opinions express the sentiment of the authors. Subjective statements (factual data) often do not provide actual personal opinions and sentiments; however, including them might decrease the classification results.

6.3.2 Spelling Correction

Searching through restaurant reviews, we very often found misspelled words (e.g. "It is soooo gooood"). These errors can be made unintentionally or intentionally. Since some of these misspelled words are critical in analyzing the correct sentiments from the reviews, fixing misspelled unigrams in the data set will hopefully yield meaningful improvements. Several methods for auto correcting such misspelling issues are to combine a dictionary with Levenstein edit distance or keyboard distance. More advanced techniques can be applied in particular context such as making use of n-grams, and pos tagging.

6.3.3 Learning Techniques

In our experiment, we investigated the effectiveness of several supervised machine learning algorithms with specific parameters. Different setup parameters or learning techniques may produce better results.
As in our settings, sentiment classes: negative, neutral, and positive can be order by the opinions of the authors. Regression techniques such as linear regression, support vector regression can be applied with different threshold boundaries.
Another option is unsupervised learning techniques such as clustering. Three clusters can be formed from three classes of sentiment.

7 Conclusions

In this report, we have analyzed the sentiment of restaurant reviews. We used the House of Prime Rib's reviews on Yelp.com as our text corpora. We evaluated the effectiveness of different feature selection and supervised learning techniques on the classification of reviews according to three classes of polarity (positive/neutral/negative). The results showed that for our specific simple unigram bag-of-words model can perform relatively well compared to other more advanced features such as bigram, adjectives and adverbs, sentiment dictionary, etc. It also revealed that our negation handling helped improved greatly the accuracy of the classification task. Moreover, NaiveBayes appeared to be the most efficient classification algorithms for our particular small corpus, and it was

the classifier in our latest version. Based on the result of experiments, we constructed a combined feature selection for our final model. The F-measure score of our final model is much higher compared to other simple feature selections. We also explained the obtained results as well as provided suggestions for future work.

References

[1] Bennett, P.N.: Assessing the calibration of Naive Bayes posterior estimates. Technical Report CMU-CS-00-155, School of Computer Science, Carnegie Mellon University (2000)

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

[3] Kushal Dave, Steve Lawrence, and David M. Pennock. Mining the peanut gallery: Opinion extraction and semantic classication of product reviews. In Proceedings of WWW, 2003.

[4] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04), 2004.

[5] Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-2002), 2002.

[6] Karl-Michael Schneider: Techniques for Improving the Performance of Naive Bayes for Text Classification. Sixth International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2005), 2005

[7] Toutanova, Kristina, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In Proceedings of HLT-NAACL, 2003.

[8] Junichi Tatemura. Virtual reviewers for collaborative exploration of movie reviews. In Proceedings of the 5th international conference on Intelligent user interfaces, 2000.

[9] Peter Turney. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the Association for Computational Linguistics (ACL), 2002.

[10] http://sentiwordnet.isti.cnr.it/

Appendix A - List of library

In order to implement crawler as well as classifier, we have made uses of different open-source java package:

- Jsoup: is a Java library for working with real-world HTML. It provides a very convenient API for extracting and manipulating data, using the best of DOM, CSS, and jquery-like methods. We made use of it to implement Yelp crawler.
- Jama: is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real, dense matrices. We made use of it to implement weighting scheme
- Apache Lucene: is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. We made use of it to implement stem-mer and lemmatization.
- Standford Parser: A natural language parser is a program that works out the grammatical structure of sentences. We made use of it to implement tokenization and POS tagger.
- Weka: is a collection of machine learning algorithms for data mining tasks implemented in Java. We made use of it to implement classifiers.

Appendix B - List of File

Our program consists of 5 java package. In this section we will describe briefly about them:

ir.sentiment.crawler contains file that support crawling tasks.

- *WebCrawler.java*: contains general methods for fetching HTML from an URL and retrieve information in specific HTML tags.
- *YelpCrawler.java*: contains methods to obtain structured information from Yelp.com.
- *XMLDumper.java*: contains methods to write obtained information to XML file.

ir.sentiment.model contains file that support general text functions and review sentiment functions.

- *Review.java*: describes a review object according to a pre-defined structure.
- *TextUtils.java*: contains general text utility functions such as: tokenziation, POS-tagging, negation regrex matching, stemming, lemmatization, etc.
- *StopWordList.java*: contain methods for stop word filtering.

ir.sentiment.extractor contains methods for feature extraction.

- *ReviewReader.java*: extracts review information from XML format.
- *UnigramExtractor.java*: extract unigram and unigram with negation handling features as well as document term matrix for these features.
- *BigramExtractor.java*: extract bigram feature as well as document term matrix for that feature.
- *DictionaryWordExtractor.java*: extract sentiment dictionary words and sentiment dictionary word with negation handling features as well as document term matrix for these features.
- *POSExtractor.java*: extract unigram (with POS tags) and unigram (with POS tags) with negation handling features as well as document term matrix for these features.
- *TaggedWordExtractor.java*: extract adjectives, adverbs and adjectives, adverbs with negation handling features as well as document term matrix for these features.
- *FinalExtractor.java*: extract features describes in section 5 as well as document term matrix for that feature.

ir.sentiment.weight contains methods for weighting scheme transformation.

- *WeightingScheme.java*: contains methods for presence, TF-IDF transformation.

ir.sentiment.classifier contains methods for classification phase.

- *FeatureGeneration.java* extracts feature vector describes in section 5 for test set.
- *NaiveBayesClassifier.java*: multinomial NaiveBayes classifier.
- *ReviewWriter.java*: writes the result to XML file.

Main.java : Main program for classification.