Sentiment Analysis on reviews of Yelp Dataset

Nathan Mots, Prashant Chhabra, and Melina Sparks

Department of Computer Science, University of Victoria
nathanmots@live.ca, prashant.chhabra89@gmail.com, msparks@uvic.ca

Abstract

Natural Language Processing is currently an important area of research in the field of computer science already giving way to many powerful tools, with their own wide breadth of possible uses. One of these tools, sentiment analysis allows quick and accurate means to check the sentiment, positive or negative of some document by using a variety of well established data mining techniques. Although there is still a variety of current research in the field of sentiment analysis, the tools we already have to do simple sentiment analysis are already well flushed out, and are widely available to laymen who might need them. Yelp is a website that provides user recommendations and reviews for local businesses, restaurants, shopping, entertainment, and many other services. In this project, we use different sentiment analysis techniques to classify sentiment of reviews on yelp dataset.

Contents

1 Introduction

Natural Language Processing offers a powerful tool to derive positive or negative sentiment from a document using machine learning processes. The data yielded from this can provide a wide range of useful information in uncountable and often unexplored fields. As the field of Natural Language Processing has begun to be explored deeper. As well, simple uses for it have become more available to the average person with an interesting in Data Mining, however many are not aware of this. Our goal, going into this fairly naive about Natural Language Processing, what it was, or how to use it, was to glean some experience of the difficulty of use of Natural Language Processing with respect to a very applicable problem which was extracting sentiment from Yelp reviews.

Our own research into this question of ease of use is important as we are a group of non-experts in the area compared to a field where most people working in it are writing from a point of view of expertise already gathered, something we have yet to gain when starting out.

Though this is an explored area of interest it is still important as showing the ease of learning and use of a process such as Natural Language Processing can help us to understand if it needs to be made less or more available to non-experts. The results we gathered show how difficult it was for a group of non-experts to understand and implement some simple Natural Language Processing techniques to run accurate sentiment analysis on a real data set.

We took a two prong approach to the problem. The majority of our work was through using a simple tool ‘WEKA’ that is easy to learn and understand, what different methods could we use to apply sentiment analysis and what kind of accuracy would these simple to learn techniques give us on our data set. Additionally, we used an out of the box Java Linguistics library to implement a simple java program that would be able to classify sentiment on our data set, and test the accuracy of this, showing how easy it is for a lay person to utilize already existing packages to quickly implement a sentiment analysis tool, and how accurate such a tool could be.

2 Related Work

The idea of using sentiment analysis to draw conclusions on a users opinion is by no means a new concept. Other sources have taken similar steps to evaluate how accurate NLP can be. Studies have been done that have taken into account the noisiness of user reviews labeled by a discrete number value accompanying a review, and similar to what we have done, have attempted to negate this noise by just labelling the review as positive or negative instead of as a discrete numeric value. This has demonstrated greater accuracy than more supervised learning algorithms on the same data using the discrete numeric values [11]. Other research has again taken advantage of online rated reviews as a means of testing their classifiers, but used more complex means of classification: unsupervised and supervised techniques to learn word vectors capturing semantic term–document information and rich sentiment content. This allowed for a classifier that was able to use both continuous and multi-dimensional sentiment information [5]. Although we can also show again that the use of investigating simpler techniques such as bag-of-words model refined with choice of features based on the semantics and syntactic information from the text also yields us a simple classifier with a good rate of success [14].

Previous research has also shown that SVMs provide accurate tools for correlating user reviews to ratings, stronger than rule based classifiers [7].Furthermore we can note that examining well document classifiers and their effectiveness does not go without it's own merit. Other research shows us that even well documented classifiers such as Naive Bayes have a lot of potential on their own when tweaked, with techniques such as good negation handling, word n-grams and feature selection by mutual information, can get results of approximately eighty eight percent with just naive bayes alone [6]. It is also worth noting that one area we could focus more on, especially since our data was all based around restaurants is domain specific knowledge. Other research shows that sentiment analysis techniques are less effective without domain specific knowledge of the field, for example computer product reviews versus restaurants [3].

Other Research has addressed more specific qualities in the yelp data set. That is examining specific qualities in a yelp review, correlating the rating to either the service, quality of food, or other specific qualities of the experience instead of just the dining experience as a whole. This was done by using 'codewords' within the text. Classifying by this means showed to actually provide both good correlations with the specific quality being discussed but also better accuracy of the text to rating in general [4]. As well, other research has looked specifically at using SVM and learning word vectors to try and use individual words to correlate to the overall sentiment of a yelp review [2].

3 Data Preprocessing

Data pre-processing was the step taken to collect and prepare the input data for data mining. With this process the objective was to have a quality, resulting dataset. A quality data set has reduced irrelevant and redundant information as well as the reduction of noise and unreliable data [8]. The first task was deciding on a data set, and then using different collection techniques to retrieve the data.

3.1 Data Collection

The dataset chosen for this project was retrieved from the Yelp Academic Dataset challenge website [12]. To gain access to the dataset, a name and email had to be provided and the terms of use had to be accepted. After these steps were taken, a zip file, containing five different JSON files, was downloaded. The zip file was decompressed into a folder that contained the full dataset for the challenge. The files included in this folder were named business.json, review.json, user.json, checkin.json, and tip.json. For the analysis of the reviews, only the business and review files were needed. The data object schematic for each JSON file is as followed:

Business Object [13]

```
{
  'type': 'business',
  'business_id': (a unique identifier for this business),
  'name': (the full business name),
  'neighborhoods': (a list of neighborhood names, might be empty),
  'full_address': (localized address),
  'city': (city),
  'state': (state),
  'latitude': (latitude),fi
  'longitude': (longitude),
  'stars': (star rating, rounded to half-stars),
  'review_count': (review count),
  'photo_url': (photo url),
  'categories': [(localized category names)]
  'open': (is the business still open for business?),
  'schools': (nearby universities),
  'url': (yelp url)
}
```

Review Object [13]

```
{
  'type': 'review',
  'business_id': (the identifier of the reviewed business),
  'user_id': (the identifier of the authoring user),
  'stars': (star rating, integer 1-5),
  'text': (review text),
  'date': (date, formatted like '2011-04-19'),
  'votes': {
          'useful': (count of useful votes),
          'funny': (count of funny votes),
          'cool': (count of cool votes)
  }
}
```

The next phase was finding the most efficient way to access the data as the JSON files were very large and took a lot of time to process on a personal computer.

3.2 Data Transformation

The JSON files needed to be accessed in a way that was more efficient than using a text editor alone. MySQL offered the functionality that was needed to process the data at an efficient rate, but first the data needed to be in the right input format for MySQL.

3.2.1 Stage One - json to csv

Comma Separated Values (CSV) can be easily imported into MySQL. Hence, json files were converted into CSV files using a python script [1]. The code was downloaded from the GitHub repository and used as a black box to convert the files. Once this had been completed, MySQL could be used for the final data transformation.

3.2.2 Stage Two - data cleaning using MySQL

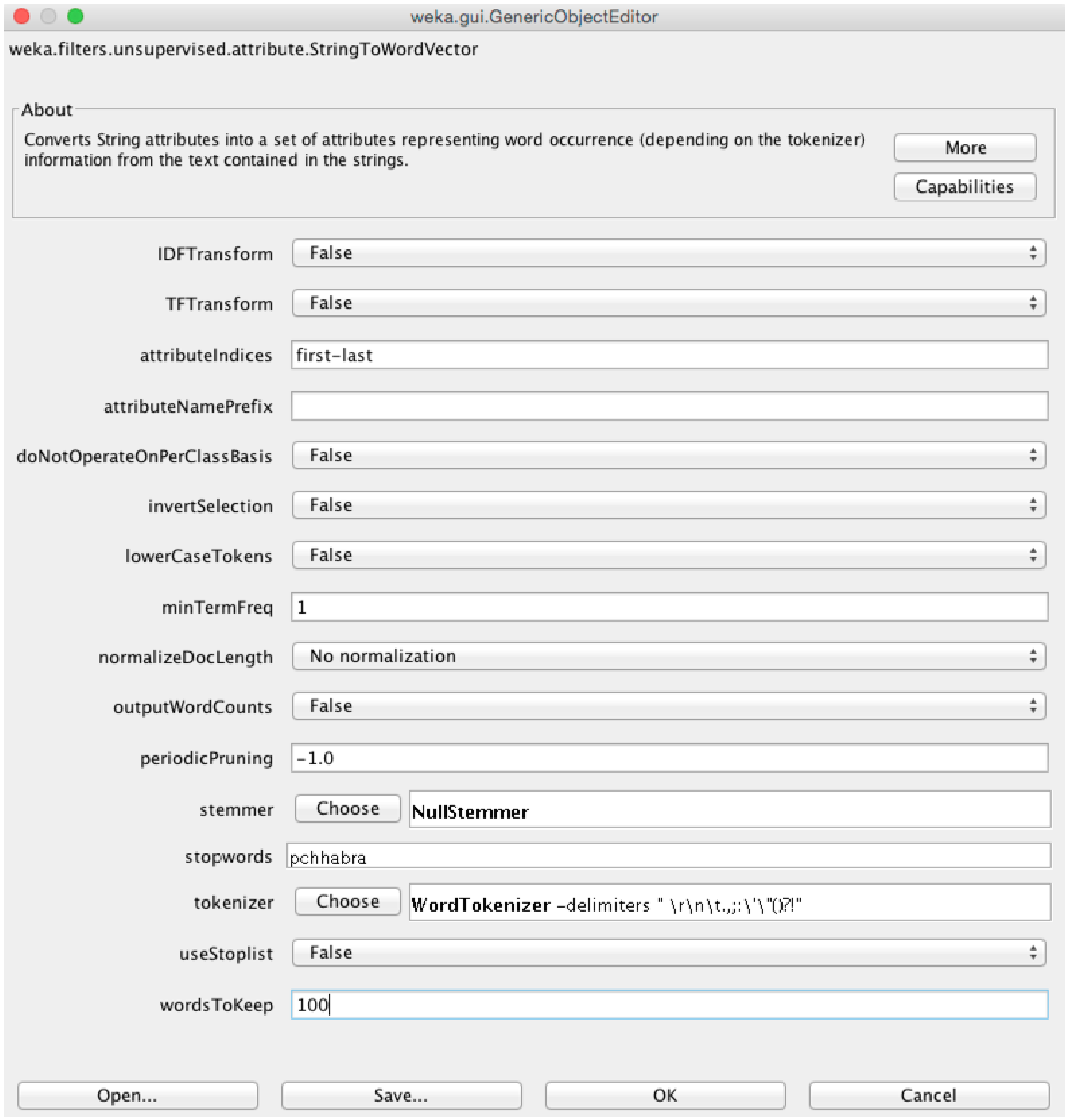
The next phase of the data-pre-processing was to clean the data using MySQL. First, SQL queries were used to remove unnecessary attributes from the dataset. For example, the longitude, latitude, and date were not relevant to this problem and could therefore be removed. The attributes that were used included the business id, categories, stars, and text. The business id is a unique identifier for each tuple and each business can have multiple categories. For example, a specific business could have the category of bar and restaurant. The stars and text represent the attributes associated with a given review for a specific business. The stars are integer values from 1 to 5 where 1 is bad and 5 is great. The written review is represented with the text attribute, which contains the text. After the irrelevant data was removed, the next step was to decide what data would produce promising results.

To ensure consistency throughout the data, a homogeneous dataset needed to be used. This was the reasoning behind using only restaurant reviews instead of a variety of different business reviews. Since this dataset was so large, steps needed to be taken to reduce the processing time. To accomplish this, the dataset was reduced to 10,000 reviews. 5000 of these reviews were five star reviews and 5000 were one star reviews. One and Five star reviews were selectively chosen, as well as the even number of tuples per class value, to reduce bias in our training data. Two to four star reviews were ignored and the data pre-processing was almost complete. One star was replaced by “no” indicating negative sentiment and Five star was replaced by “yes” indicating positive sentiment.

The prepared data was then exported from MySQL into a CSV file. File headers specifying attributes and class values were added to the CSV file and then it could be saved as an arff file. This is the input file type specific to Weka. Unfortunately Weka had issues reading the file due to the different special characters contained in the text portion of the data. Some of the special characters included “\n”, “\r”, “,”, ““, etc. To fix this issue, MySQL was used again and the special characters were replaced with spaces.

3.2.3 Stage three - Preprocessing using weka

Weka can’t handle string attributes hence, additional preprocessing is required before using classification algorithms of weka. String to work vector filter was used to convert string into attributes. This filter was used with default settings but “words to keep” was changed to 100 to speed up preprocessing. Figure 1 specifies settings used in String to word vector filter


weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.StringToWordVector

About

Converts String attributes into a set of attributes representing word occurrence (depending on the tokenizer) information from the text contained in the strings.

IDFTransform	False
TFTransform	False
attributeIndices	first-last
attributeNamePrefix	
doNotOperateOnPerClassBasis	False
invertSelection	False
lowerCaseTokens	False
minTermFreq	1
normalizeDocLength	No normalization
outputWordCounts	False
periodicPruning	-1.0
stemmer	Choose NullStemmer
stopwords	pchhabra
tokenizer	Choose **WordTokenizer** -delimiters " \r\n\t.,;:\'\"()?!"
useStoplist	False
wordsToKeep	100

Figure 1. String to word vector filter.

4 Data Mining

Data mining is the process in which algorithms are executed, on the processed data, with the overall goal of transforming the data into a more understandable format [9]. With this specific problem, different classification algorithms were implemented to predict if the given review is positive or negative. Before running the classification algorithms, sentiment analysis needed to be implemented to identify the important keywords.

4.1 Weka

After completing preprocessing, different classifiers were used to figure out the best one for text classification. Hence, all these algorithms were run using default settings. Youtube tutorial was referred to accomplish sentiment analysis using weka[16]. Table 1 summarises results of running six different classifiers.

Algorithm	Accuracy	Time to build Model
Naive Bayes	72.32%	.57 seconds
Multinomial Naive Bayes	80.52%	.11 seconds
K- Nearest Neighbour(5)	73.16	.01 seconds
J48	76.86	9.33 seconds
Random Forest (20 features 100 trees)	82.89%	78.06%
SMO(Support Vector Machine)	83.79%	413.96%

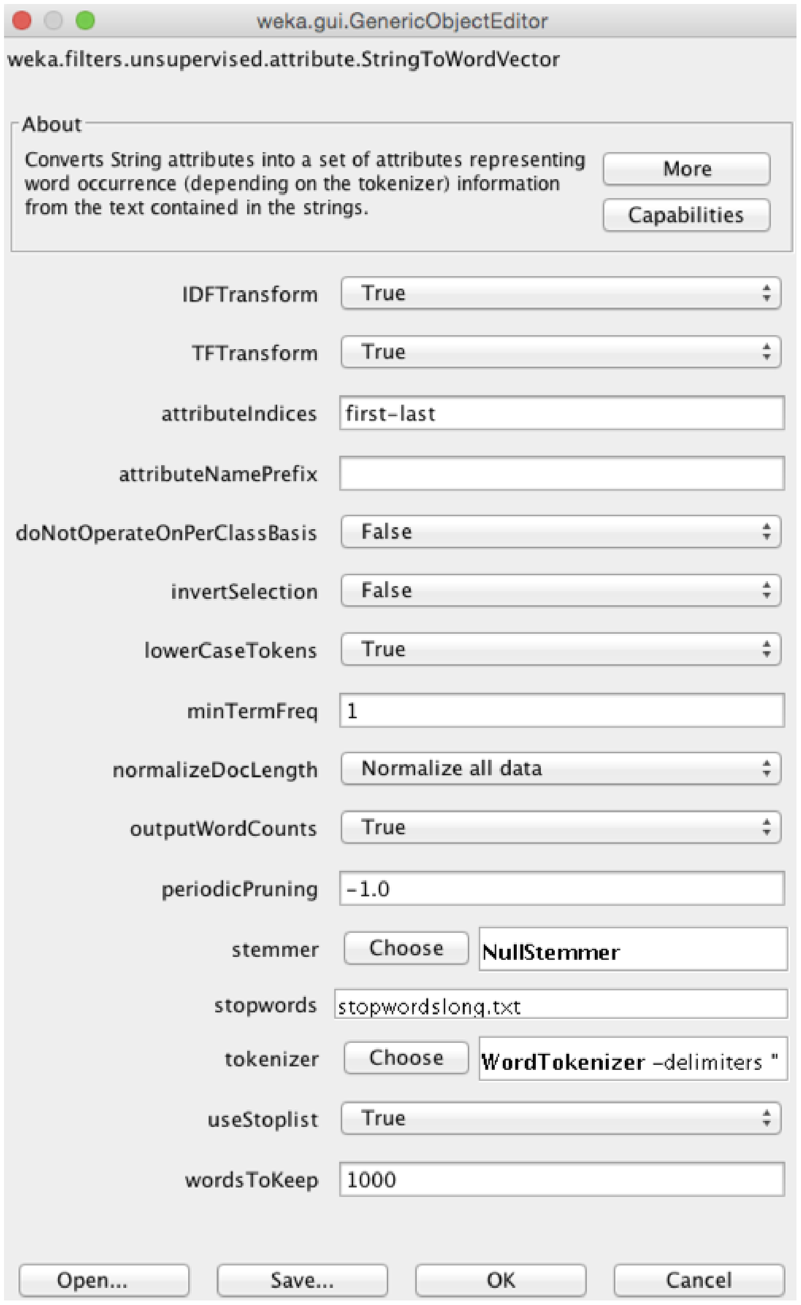
Table 1. Results of six different classifier using weka

We can clearly see from above table that Support vector machine provided best accuracy and Multinomial naive bayes gave a good balance of speed and accuracy. Hence, we used support vector machine for rest of analysis and tried to play around with some of the settings of string to word vector filter to improve accuracy of SMO. We played around with following settings of String to Word Filter [15] :-

- IDFTransform - Transform each word frequency into:fij*log(num of Documents/num of documents containing word i) where fij if frequency of word i in jth document(instance).
- TFTransform - Transform the word frequencies into log(1+fij) where fij is the frequency of word i in jth document(instance).
- LowerCaseTokens - Make words case insensitive.
- NormalizeDocLength - Normalize word frequencies for a document.
- OutputWordCounts - Output frequency of word rather than just presence and absence of a word.
- Stemmer - To count words having same root as one word.

- Stopwords - Removal of stopwords which don't really tell anything about sentiment of a document ex - of, them etc.
- Tokenizer - Try different tokenizer algorithms
- WordsToKeep - Number of words to be used as attributes.

Figure 2 shows settings of String to Word Vector which increased accuracy of SMO from 83.79% to 94.64%.



weka.filters.unsupervised.attribute.StringToWordVector

About

Converts String attributes into a set of attributes representing word occurrence (depending on the tokenizer) information from the text contained in the strings.

IDFTransform	True
TFTransform	True
attributeIndices	first-last
attributeNamePrefix	
doNotOperateOnPerClassBasis	False
invertSelection	False
lowerCaseTokens	True
minTermFreq	1
normalizeDocLength	Normalize all data
outputWordCounts	True
periodicPruning	-1.0
stemmer	Choose NullStemmer
stopwords	stopwordslong.txt
tokenizer	Choose WordTokenizer -delimiters "
useStoplist	True
wordsToKeep	1000

Figure 2. String to word vector settings to improve accuracy of SMO

4.2 Java LingPipe

Sentiment analysis is a fairly well developed field with lots of pre-existing tools, often packaged with other linguistics libraries. We wanted to show how easy it could be to use one of these out of the box libraries to perform Sentiment Analysis. After browsing a few different packages for Python and Ruby, we settled on Java LingPipe library, due to it's good documentation and range of tutorials.

Because of the way the data was pre-processed, we had already obtained a CSV that had three columns, the review ID, the review, and a 'yes' or 'no' value for if it was a good review. Furthermore the review text had already had special characters, commas and quotations already parsed out, to make process it easier. Using an already existing tutorial from LingPipe on how to create a classifier and train it on text files, it was trivial rewrite this code to instead parse a CSV file into an array to use instead of text files. From here it was just necessary to define categories, as mentioned above either 'yes' or 'no', create a classifier, train it, then evaluate its performance, all using functions available from the LingPipe library. The code really just needed to identify if the current review was a training tuple or not to train or evaluate on.

Originally training tuples where defined by setting a global variable to a factor, any tuple's array position where the modulus of the array position would be trained on, this created a quick way to set training and non training data, that would be distributed throughout the data set. Once this was confirmed to work we wanted to add k-fold validation so that the numbers would be comparable with the accuracy counts we had gotten doing the WEKA tests.

Since the list was split into the first half being 'yes' and the second half, 'no', evaluating folds by splitting the list into *k* sections and evaluating on fold *i* of *k* would yield a wide range of accuracies due to the the data training on skewed mixtures of 'yes' and 'no' categories. This problem was remedied by evaluating on some *i* section of tuples in *k*/*m* tuples of each *k* fold. Where *k* is the count of specified folds, and *m* is the number of tuples in each fold and *i* is the current portion of that fold being evaluated on. This distributed the training and evaluating data much more evenly and gave much more averaged results for each iteration of the k-fold cross validation.

The classifier type we used from the LingPipe library was an 'NGramProcessLM'. According to the javadoc, this type did process using a combination of Bayes, as well as weighting the values of words and their extensions. Using this type without k-fold cross validation, and just evaluation on a distributed number of values we found high accuracy on our data set, using half training and half evaluation data left us with approximately 96 percent accuracy. But reducing the training set down to just 213 of our 10 000 tuples still gave us approximately ninety percent accuracy. To validate these results further, the implemented k-fold cross validation was ran with fold values between 3 to 15, rendered accuracies between ninety-two to ninety-five, respectively. Figure 3 shows Percentage of success compared to number of test cases.

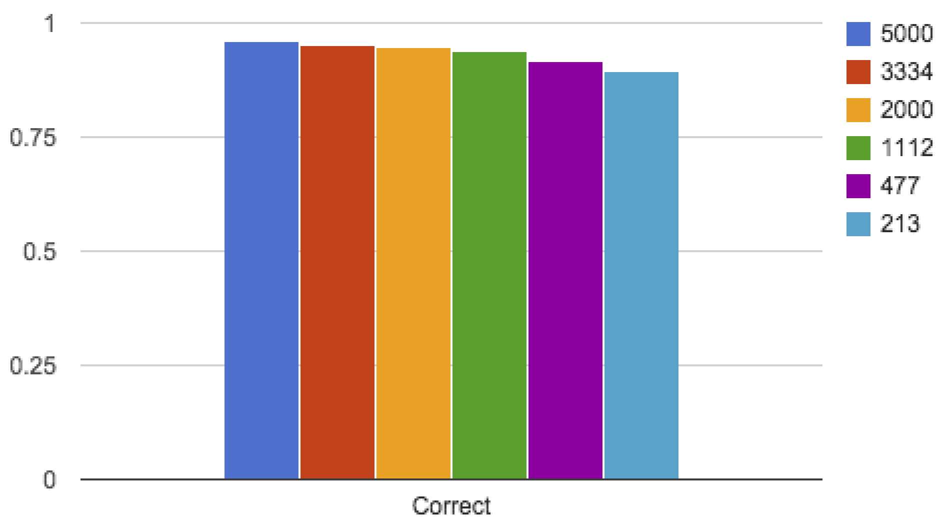


Figure 3. Percent success compared to number of test cases.

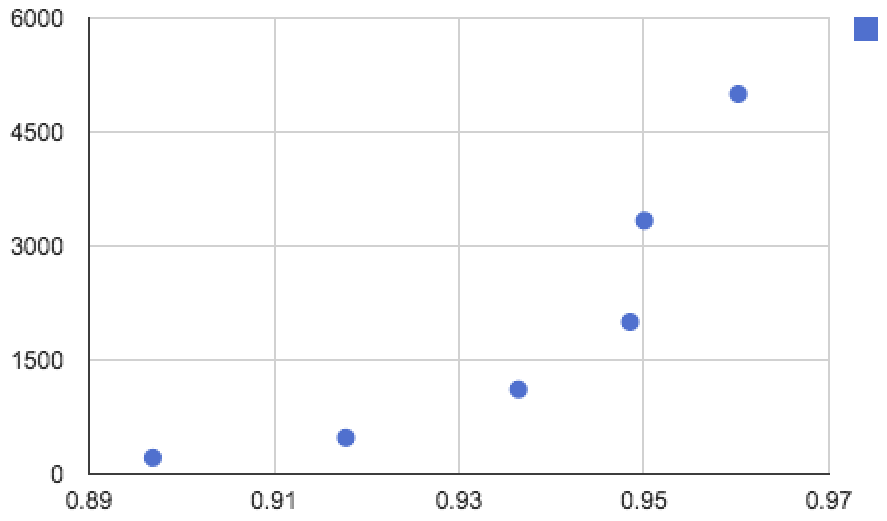


Figure 4. Test cases versus correctness of evaluation

5 Conclusion

In this project, we applied different classification techniques of sentiment analysis on reviews of yelp dataset to classify a reviews as positive and negative. We learned to use different algorithms for sentiment analysis and also played around with different settings preprocessing settings. We found that Support Vector Machine was slow but gave good accuracy and Multinomial Naive bayes gave good balance of speed and accuracy. Java Lingpipe library also gave us good accuracy and did not show a drastic decrease in accuracy on reducing number of test instances.

We believe that high accuracy of classification can be attributed to homogeneity of our dataset as we considered reviews of only restaurants and also to class balance as we took same number of reviews for both “yes” and “no”class values. From our results we can conclude that various algorithms and tools available for text classification are good given a good clean dataset which is free from class bias.

However, this project has some limitations as we considered only a very small subset of reviews. Reviews of only restaurants were considered and all reviews except one star and five star were ignored. Our accuracy might decrease if we take all these things into account.

6 Future Work

With this project, there is further work that can be done to extend the versatility of the review classification. There is one main change that could be made to accomplish this. It would be ideal to consider more of the overall data.

More of the data would result in a more realistic accuracy than the perfectly chosen data. For example, three values, instead of two, could be used for the class value so that the two, three and four star reviews are included. A good, bad or neutral value could be given to four or five, one, and two or three stars respectively. This classifier would cover more of the overall data and give a more accurate reading of how well the classifier is working. Another way to include more of the data would be to use the entire dataset instead of just 10,000 reviews. This would provide the training and test cases with more tuples to work with and businesses other than just restaurants will be able to have a classification prediction. With these suggestions the results may have a decreased accuracy, but the results will be more valuable overall because the classifier will be modelled after a more realistic dataset.

This dataset can be used to other questions [13] like “How much of a business’ success based on its location?”, “Seasonal effects on number of reviews of specific type of business”, “Prediction of type of business on basis of text of review”, “Correlation between reviews and tips”

Acknowledgement

This report is presented as part of project for Spring 2015 Data mining course offered at University of Victoria. We thank Dr. Alex Thomo for his guidance throughout the course.

References

[1] (7 November 2014). Yelp's Academic Dataset Examples. Available: https://github.com/Yelp/dataset-examples.
[2] J. Jong, "Predicting rating with sentiment analysis," 2011.
[3] C. Leung and S. Chan, "Sentiment analysis of product reviews," SAR, Hong Kong, .
[4] J. Linshi, "Personalizing yelp star ratings: A semantic topic modeling approach," New Haven, CT, .
[5] A. Maas, R. Daly, P. Pham, D. Huang, A. Ng and C. Potts, "Learning word vectors for sentiment analysis," Stanford University, Stanford CA, .
[6] V. Narayanan, I. Arora and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model," .
[7] F. Peleja and J. Magalhaes, "Opinions in user reviews: An evaluation of sentiment analysis techniques," Caparica, Portugal, .
[8] (19 March 2015). Data Preprocessing. Available: http://en.wikipedia.org/w/index.php?title=Data_pre-processing&oldid=652054261;.
[9] (1 March 2015). Data mining. Available: http://en.wikipedia.org/w/index.php?title=Data_mining&oldid=649387557.
[10] (3 April 2015). Sentiment analysis. Available: http://en.wikipedia.org/w/index.php?title=Sentiment_analysis&oldid=654779025.
[11] A. Yates, N. Goharian and W. G. Yee, "Semi-supervised probabilistic sentiment analysis: Merging labeled sentences with unlabeled reviews to identify sentiment," Asist, Quebec, Montreal, 2013.
[12] Get the Data. Available: 12.
[13] Yelp's Academic Dataset. Available: https://www.yelp.ca/academic_dataset.
[14] K. Yessenov and S. Misailovic, "Sentiment analysis of movie review comments," May 17. 2009.
[15] "StringToWordVector" [Online]. Available : http://weka.sourceforge.net/doc.dev/weka/filters/unsupervised/attribute/StringToWordVector.html
[16] "WEKA Text Classification for First Time & Beginner Users" [Online] Available : https://www.youtube.com/watch?v=IY29uC4uem8

Glossary

arff	The data type used as input for Weka. These files consist of headers containing attribute details and comma separated data for each attribute.
CSV	Comma separated value file format is a file type that has lines of plain text separated with commas. This is a very general format used for different applications.
GitHub	A file repository system that is located online at https://github.com/ that offers revision control for a set of checked in files.
JSON	JavaScript Object Notation is a programming language used for representing groups of objects.
MySQL	MySQL is a version of a database system that uses a Structured Query Language to implement queries to manipulate data.
Weka	Weka is an application that implements and includes many different learning algorithms for data mining.