Text Classification using Sentiment Analysis

Shailesh Ahuja
School of Computer Engineering

Prof Chang Kuiyu
School of Computer Engineering

Abstract **-** This paper illustrates a method to classify reviews of various products as positive, negative or neutral. The method involves parsing of reviews to extract adjectives and relations. The extracted data is used to score the reviews using a dictionary of term-score data. Rules are imposed on top of the dictionary score to modify the overall score of the review. The final score determines which class the review belongs to. This method is compared to established methods of classification such as Support Vector Machines and Naive Bayes. It is also shown how this approach was derived and justifies this approach.

Keywords – sentiment analysis; text classification; text parsing; english; reviews; artificial intelligence

1 INTRODUCTION

The established methods of text classification such as Support Vector Machines, Naive Bayes or Neural Networks use a lot of mathematics and do not represent how humans judge documents. These use statistical information from a corpus and therefore random cases are classified incorrectly. The objective of this research was to develop a system to classify reviews that represents human judgement as close as possible and therefore achieves high precision and recall. The following steps are followed:

1) Reviews are broken down in sentences and sentences into terms.
2) Various word classes are extracted such as adjectives, nouns, verbs and adverbs. The dependencies between terms are also extracted.
3) A lexicon that stores the score of each term is generated.
4) This lexicon is used to calculate the overall score of the document.
5) Various rules are applied and tested, for e.g. the term ‘not’ can change the polarity of a word.
6) Test sets with human labelled reviews are used to find out the significance of the features and rules in classification.
7) More relevant features are given higher weighting than less relevant features.

Third party tools like Stanford Parser [1] and nltk [2] are used for the first two steps. The first part of this research focussed on developing a lexicon, as stated in the third step. The other part of the research focuses on testing, application and implementation of as many features possible. These features will be discussed in details in the coming sections.

2 TESTING AND EVALUATION

2.1 TEST SET

A common test corpus is used for all the methods described in the subsequent sections. 1000 restaurant reviews were crawled from hungrygowhere [3] and were manually labelled as positive, negative or neutral. If a review was considered a positive review for the restaurant, it was labelled as 1, if neutral then 0, and if negative than -1. For some reviews, whose sentiment could not be decided easily, a polarity was preferred over neutral.

2.2 EVALUATION MEASURES

For each class, the precision, recall and F1 score was computed on the results of the classifier. For the overall score of the classifier accuracy of the results was computed on the entire set. A point to note is that all measures may not be available for some evaluations as these measures were decided in the course of this research.

3 PREPARING THE TERM SCORE LEXICON

The idea behind the first part of the research was to find ways to give a score to the words. There are various possible ways to give score to a term. There are more than a million terms in English and it was not feasible in this research to manually give a score to all terms. Two kinds of approaches were tried out:-

1) Automatic scoring using WordNet [4] or Thesaurus [5] and propagation through synonyms and antonyms.
2) Using a statistically scored resource and manually modifying scores of domain relevant terms.

3.1 AUTOMATIC LEXICON GENERATION

This section covers two ways of generating a lexicon automatically. They both use the score propagation method using synonyms and antonyms as described in [6]. This approach uses a graph representation of English terms. The graph is represented as an adjacency matrix. The vertices of the graph are the terms to be scored. The steps involved are discussed in [6] in detail, but a brief summary is given here to help the readers understand the issues in this approach.

Initially a list of positive, neutral and negative terms are chosen. These are given scores of 1, 0, -1 respectively. All those terms which are synonyms of a certain term are given a score similar to that the term. This way the matrix is updated. Then this matrix is multiplied by the score matrix to a certain number of times. This gives us a final score matrix.

To have a feasible solution, the 5000 most common words in English were used. This process has exponential complexity and therefore it is not feasible to use the entire dictionary of English words.

3.1.1 Using WordNet

WordNet has a graph like structure in which a Synset is defined as a group of synonyms. Antonyms are a one-one relation.

Table 1 Sample Adjacency matrix using three words. Note how synonyms have a positive influence and antonyms have a negative influence

	Good	Bad	Acceptable
Good	1.2	-0.2	0.2
Bad	-0.2	1.2	0.2
Acceptable	0.2	-0.2	1.2

The following table shows the precision accuracy and recall after the overall scores were tested with a threshold of 1.

Table 2 Test Results using Wordnet graph score

	Precision	Recall	F1
Positive	65.84%	59.16%	61.85%
Negative	35.56%	42.17%	38.18%
Neutral	12.36%	9.72%	10.28%
Accuracy		50.47%	

As it can be seen, the F1 scores using this method are very low. This can be understood by the following reasoning:

1) Synonyms don't mean that the words should have similar score, and antonyms don't mean that words should have opposite scores. Some words have different meanings in different contexts, e.g. 'better' can be used in the followings ways:

   i. The food here is better than George's (positive)

   ii. The food at George's was better (negative)

   A synonym of 'better' is 'well'. 'well' would be rarely used in this scenarios and also the polarity of the word is a bit lower.

2) Even if synonyms had the same meaning, a everything depends on the seed word lists given to the algorithm. Choosing the correct seed list is a very hard task. There is no clear boundary which words are relevant to the domain.

3) The polarity of these words can be relatively more or less, e.g. giving 'awesome' and 'good' the same initial score of '1' means they weigh equally, which is rarely true.

4) The individual word scores depend on a variety of thresholds which gives this method a range of possible values. It is difficult to adjust all of them because there is no real definition of what should be the correct value.

5) The choice of whether to include or exclude every word makes a difference to the individual word score. Also, whether the word should be included in the final review score calculation is another question. A word like "went" gets a score of '5.5'. This word doesn't really matter when deciding the sentiment of the review. This observation will guide the most important part of this research.

3.1.2 Another attempt at automatic lexicon generation using Thesauras

Thesauras is an online dictionary with REST API for getting synonyms and antonyms of terms. Instead of WordNet, this dictionary was used to determine the matrix scores and used to compute the individual word scores. After evaluating this result, it resulted in a similar F1 score as shown in the table below:

Table 3 Test Results using Thesauras graph score

	Precision	Recall	F1
Positive	62.98%	60.23%	60.98%
Negative	38.76%	39.42%	38.49%
Neutral	14.18%	10.43%	11.66%
Accuracy		50.20%	

3.1.3 Using a bootstrapping algorithm to pick relevant words

One of the problems with the automatic scoring is that it is difficult to judge all terms in the English dictionary as to whether they are domain appropriate or not. Aboot [7] is an algorithm to extract all words related to a

word. It uses a recursive strategy to keep adding words related to the words already added. It needs a corpus to get all the domain words. Briefly, these rules are followed:-

1) If two words occur together in the same document (review), then they are more likely to be related
2) If both the words don't occur in the document, then they are more likely to be related
3) If one word occur occurs in a document, and the other doesn't then they are less likely to related

Following these rules we get the words related to the domain. However, on inspection, it turns out that there are a quite few words that are not related, but still present in the final set. This could also mean that quite a few words could be relevant terms could be left out. This algorithm does quite well given just one seed word, but it is not sufficiently precise. The results don't improve, and this means that automatic word scoring is not precise enough for the task ahead.

3.2 MANUALLY ALTERING A STATISTICALLY SCORED SET

The automatic scoring of words did not give good results. This can be attributed to the fact that the system involves two automatic steps.

1) Automatically assign scores to individual terms.
2) Automatically assign sentiment to a review (classification).

Even if we assume 70% F1 score for the first step and second step, the overall score becomes:

0.7 X 0.7 X 100= 49%

If we want to improve the second step, we need a precise dictionary of terms and their scores. It is not possible to judge the improvements in the second step correctly if the first step has incorrect results.

3.2.1 Using SentiWordnet dictionary for scoring terms

SentiWordnet [8] is a resource that has statistically given scores to terms. These scores denote the sentiment carried by the terms. The sentiment carried by the term is directly proportional to the absolute weight of the term. A positive weight means positive sentiment and likewise for negative sentiment. It is another automatically scored resource, but it is a good start for manually modifying the scores.

After just using the raw dictionary, the following scores were obtained:

Table 3 Test Results using raw SentiWordnet

	Precision	Recall	F1
Positive	72.87%	79.11%	75.86%
Negative	59.49%	57.71%	58.59%
Neutral	25.87%	19.79%	22.42%
Accuracy		63.27%	

Scores are already better, and we haven't yet started to modify the scores.

In SentiWordnet, the adjectives, verbs, nouns, adverbs have been marked separately. Since we earlier observed that there are some types of words that add no value to the overall sentiment of the document. These sets of terms were separated and various permutations were tried to test which set gives the highest score. It turned out that adjectives are the most useful set of terms for sentiment analysis. This can be verified by the table below. Also it makes sense from the point of view of the English language. Adjectives are descriptive words and used to express feelings towards certain things. Other set of terms added more junk than value.

Table 4 Accuracy with various term types

Term Type	Accuracy
Adjectives	68.25%
Verbs	55.45%
Nouns	58.27%
Adverbs	56.24%
Adjectives + Verbs	65.46%

3.2.2 Modification of scores of terms

The first 100 reviews were treated as the training set. After manually going through the reviews, the scores were adjusted to best match the overall sentiment of the documents. For e.g. the term 'good' had an initial score of 5.75. It was observed that this term is used often even in negative reviews, for instance, "although the decor was good..." or "the meatballs were good but not great...". So the score was changed to 1.5. After careful analysis many term scores were changed and some new terms were added such as text emoticons.

4 ADDING FEATURES TO THE DOCUMENT SCORING SYSTEM

After a precise lexicon was in place, the focus was on to improve the precision and recall of the system. A lot of mistakes in scoring were observed, and various solutions were tried out to make the system think like humans.

The document was examined term by term. Instead of just adding the score of the term to the final score, some modifications were done to the score to make it better represent the sentiment automatically.

4.1 REVERSING THE SCORE OF A TERM

There were a few cases in which the term score should be exactly opposite to what it is, e.g. "not good" or "wasn't good". In such cases the sentiment conveyed is exactly opposite of what the score suggests. There is also an issue of how to detect such cases. Consider the following scenarios:

1) "**not** good"
2) "**not** so good"
3) "I did **not** like the grilled meatballs, they are supposed to be soft and tender"
4) "the food was**n't** that great"

As it can be seen, there are various ways in which negation can be conveyed. Initially only the previous word was checked for 'not' or 'n't'. It was extended to previous two words. This covers case (1), (2) and (4). Case (3) is difficult to detect. We will see how Stanford Parser (4) helped us solve this problem.

We might get some false positives with this method. E.g. :

"... was **not** great. Awesome decor though."

Here 'not' comes two words before 'awesome', and it will trigger the negation rule. To solve this problem, sentence parsing was also necessary. This was done using nltk package.

The test scores with and without the rule are shown in the table below.

Table 5 Test Results without 'not' reverse rule

	Precision	Recall	F1
Positive	74.17%	71.05%	72.58%
Negative	44.48%	59.09%	50.75%
Neutral	22.91%	17.83%	20.06%

Table 6 Test Results with 'not' reverse rule

	Precision	Recall	F1
Positive	72.93%	72.80%	72.87%
Negative	48.36%	59.59%	53.39%
Neutral	25.00%	1891.%	21.53%

4.2 DOUBLING THE SCORE OF A TERM

There are some terms for which score can be doubled, e.g. 'very'. The full list can be found in the table below.

Table 3 The terms in various modifier categories

Modifier	Terms
Negative	not, n't, no, nothing
Positive	very, so, really, super
Neutral	neither, nor
Positive impact from term "too"	good, awesome, brilliant, kind, great, tempting, big, overpowering, full, filler, stuffed, perfect, rare
Negative impact from term "too"	All other terms

Some terms such as 'too' can have both positive and negative impact. Consider the following sentences:

1) "the mushrooms were too good"
2) "the staff were too friendly"

The first sentence indicates a positive impact, whereas the second sentence indicates a negative impact. It was observed that 'too' has the same impact on the same word most of the time. So it is just a matter of classifying which terms it has a positive impact on, and which terms it a negative impact online applying the rules.

It is not necessary that we exactly double the score, or exactly reverse the score. We can create a lexicon that stores the multiplier associated with a term.

4.3 IGNORING ALL SCORES BEFORE A TERM

It is widely said that "Anything said before the word but doesn't count". It was observed that by ignoring the scores before the term 'but' in a sentence led to better results. The term 'if' also replicates the same behavior.

4.4 EXTRACTING DEPENDENCIES

When considering negative impact, case (3) was not detected. So to solve that issue and also avoid complicated checks in the context of the term, Stanford Parser was used for dependency parsing.

There are various dependencies [9] but not all are useful. Those dependencies were extracted which have at least one adjective in them. 'amod', 'acomp', 'ccomp', 'pobj' were observed to be the most useful.

These dependencies replaced the complicated checks for surrounding text and made the whole process much more smooth and readable.

It was also observed that conjunctions between adjectives transfer the modifier weight of the first term to the other term. For e.g. "…was really good and tasty". Here, the modifier is attached to the term "good", but also modifies the term "tasty". By extracting this dependency as well, the modifier information could be transferred.

4.4 OTHER FEATURES

4.4.1 Adding scores of short sentences

The parser behaves incorrectly if very short sentences of three terms or less are given. It was observed that short sentences convey a lot of emotion and the adjectives need to be considered while scoring the document. To tackle this issue, all the terms within a short sentence were checked for existence in the term score lexicon and added their score was added if available.

4.4.2 Checking important terms universally

The parser was not perfect. There were many adjectives which couldn't be detected. Also, there were terms which were not adjectives but also conveyed a lot of sentiment. So, a list of terms was created, they were checked for existence and their scores were added. Some examples: 'disappointing', 'fresh', 'underwhelming', 'sucks' etc.

4.4.3 Changing base multiplier

The number of adjective terms in a document varies, so setting a fixed threshold for classifying reviews can lead to incorrect results. We can dynamically vary the threshold depending on the number of adjectives present in the document.

The final score of the document can be calculated by the following formula:

Final score = Document score x base multiplier

Here the document score refer to the sum of scores of all adjectives in the document after the modifiers and features are applied. The base multiplier is inversely proportional to the number of adjectives in the document.

5 TESTING THE COMPLETE SYSTEM WITH MULTPLE CORPUS

5.1 RESTAURANT REVIEWS

The test corpus containing 1000 retaurant reviews from hungrygowhere was tested after implementing all the features. The following table shows the results:

Table 7 Test Results for 1000 reviews from hungrygowhere

	Precision	Recall	F1
Positive	88.67%	82.31%	84.89%
Negative	72.35%	73.13%	72.49%
Neutral	35.87%	27.36%	30.48%
Accuracy		76.50%	

Another test corpus was taken from yelp [10], which also contained 1000 restaurant reviews. A point to note is that the previous corpus had many more edge cases so the accuracy of the system is much lower on that corpus.

Table 8 Test Results for 1000 reviews from yelp

	Precision	Recall	F1
Positive	89.03%	98.57%	93.56%
Negative	93.94%	79.49%	86.11%
Neutral	50.00%	28.57%	36.36%
Accuracy		87.50%	

5.2 MOVIE REVIEWS

The manual editing of scores in the SentiWordnet lexicon was done keeping restaurant reviews in mind. So the system was not domain independent. This can also be observed in the following table. 2000 movie reviews were taken from nltk corpus and tested.

Table 9 Test Results for 2000 movie reviews

	Precision	Recall	F1
Positive	73.05%	62.84%	67.56%
Negative	76.09%	50.65%	60.82%
Neutral	N.A	N.A	N.A
Accuracy		56.75%	

To improve the accuracy for this data set, the lexicon needs to be edited to suit the needs of the domain. Work can be done to automatically do this while also not compromising the precision of the lexicon.

6 CONCLUSION

This research focussed of developing a system that represents human judgement of a review as close as possible. In most cases where results obtained were incorrect, they were marginally on the other side of the threshold. This was mainly because the reviewer was describing another subject, for e.g. a restaurant's competitor, in polarity opposite to that of the main subject. Detecting what the user is talking about is beyond the scope of this research. A setback in this research was that the entire lexicon had to be edited manually to obtain high precision. However, it is possible to design an algorithm that uses a training set to alter the existing values according to the domain. This can also make the system domain independent.

ACKNOWLEDGMENT

I wish to acknowledge the funding support for this project from Nanyang Technological University under the Undergraduate Research Experience on Campus (URECA) programme.

I would like to thank Dr Kuiyu Chang, who gave me the opportunity to pursue my field of interest and guided me whenever necessary.

Finally, I would like to thank the team members under Dr. Chang who gave me advice and pointed out some resources on the web to use.

REFERENCES

[1] "The Stanford Parser: A Statistical Parser." *Stanford Parser*. Stanford University, n.d. Web. <http://nlp.stanford.edu:8080/parser/>.

[2] Bird, Steven. "NLTK: the natural language toolkit." *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, 2006.

[3] "Singapore Food Guide." *Hungry Go Where*. N.p., n.d. Web. <http://www.hungrygowhere.com/>.

[4] Miller, George A. "WordNet: a lexical database for English." *Communications of the ACM* 38.11 (1995): 39-41.

[5] Watson, John, LLC. "Big Huge Thesaurus." *Thesauras API*. N.p., n.d. Web. <http://words.bighugelabs.com/api.php>.

[6] Blair-Goldensohn, Sasha, et al. "Building a sentiment summarizer for local service reviews." *WWW Workshop on NLP in the Information Explosion Era*. 2008.

[7] Hai, Zhen, Kuiyu Chang, and Gao Cong. "One seed to find them all: mining opinion features via association." *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012.

[8] Esuli, Andrea, and Fabrizio Sebastiani. "Sentiwordnet: A publicly available lexical resource for opinion mining." *Proceedings of LREC*. Vol. 6. 2006.

[9] De Marneffe, Marie-Catherine, and Christopher D. Manning. "Stanford typed dependencies manual." *URL http://nlp. stanford. edu/software/dependencies_manual. pdf* (2008).

[10] "Restaurant Reviews." *Yelp San Francisco*. N.p., n.d. Web. <http://www.yelp.com>.