L90 Assignment

Kaho Sato
King's College
ks789@cam.ac.uk

1 Introduction

The Internet is flooded with a huge amount of personal opinion. This ranges from a blog post on the recent U.S. election to a short review for a purchased item on an e-commerce website. Understanding what is in the mind of the population may bring a significant monetary or political value, and this gave rise to research in *sentiment analysis*, an application of natural language processing which aims to mine subjective information.

In particular, we look at the task of polarity classification of movie reviews. The polarity classification is a problem which concerns with identifying whether a given review has a positive or negative overall sentiment. The bag of words model is a simple representation which can used in polarity classification, where a text is represented by a set of words it contains. As it ignores the syntactic structure of the text completely, much of semantic information is lost. For instance, in the bag of words model, *Alice loves Bob.* and *Bob loves Alice.* have exactly the same representation, as the words they contain are identical, even though the semantics they convey are very different.

In this report, we attempt to recover the effect of negation which is otherwise lost in the bag of words model. For example, the sentiment of a sentence *The movie was not exciting.* is negative, even though it contains a positive word *exciting*, because of *not*. In this report, we refer to words which mark a negation as *negation term*. Intuitively, such information can be used to improve the performance of the classifier which uses the bag of words model. However, it would be an approximation to say a negation term affects all the words in the sentence. For instance *not* in *It was not scary, so I loved it.* only contributes to the semantics of the main clause. In other words, the effect of negation terms are limited to some scope.

In this report, I explore various heuristics that identify the scope of the negation. I use these heuristics to improve two classifiers which uses bag of words model. One is a symbolic classifier, originally presented in (Wilson et al., 2005), which uses a lexicon where each word is associated with a sentiment it represents. The other is a Naive Bayes classifier described in Pang et al. (2002). The lexicon and the dataset was provided on the framework of the module L90 in MPhil Advanced Computer Science at University of Cambridge.

The report proceeds as following: Section 2 describes the approaches presented in (Wilson et al., 2005) and (Pang et al., 2002) in detail; Section 3 describes the scoping heuristics examined; Section 4 describes implementation choices made while setting up the experiment, including how negated terms are treated by each classifier; Finally, Section 5 discusses the results obtained from this experiment.

2 Background

2.1 Symbolic Approach

Wilson et al. (2005) presents two simple approaches to sentiment analysis using a lexicon which maps a word to *sentiment score*. The sign of the sentiment score represents whether a word is associated with positive or negative document, and the magnitude of the score represents how strong the sentiment is. For instance, a word *disastrous* could have a negative score of high magnitude, and *good* could have a low and positive score. The following presents a formal definition of symbolic approaches.

Definition 1 (Symbolic approach) *Let $SLex$ be a lexicon where $SLex[w_i]$ returns the sentiment score associated with w_i. Then functions S_{binary} or $S_{weighted}$ which take a sequence of words are*

defined as follows.

$$S_{binary}(w_1...w_n) = \sum_{i=1}^{n} sgn(SLex[w_i])$$

$$S_{weighted}(w_1...w_n) = \sum_{i=1}^{n} SLex[w_i]$$

In the symbolic approach, a sequence of words $w_1...w_n$ can be classified as positive or negative according to the sign of $S_{binary}(w_1...w_n)$ and $S_{weighted}(w_1...w_n)$.

The advantage of symbolic approach over machine learning approaches is that there is no training required, which can be a serious bottleneck of the counterpart.

2.2 Naive Bayes Classifier

Pang et al. (2002) presents how one may use Naive Bayes classifier for sentiment analysis. They frame this task as a classification problem where given a feature vector f that represents a document, $\hat{c}$ among the set of classes C such that

$$\hat{c} = \arg\max_{c \in C} P(c|f)$$

is chosen. Applying Bayes' rule, $P(c|f)$ is

$$P(c|f) = \frac{P(c)P(f|c)}{P(f)}$$

As it is not feasible to compute $P(c|f)$, they make *independence assumption*, which state that each component of f is conditionally independent on class c of the document. Therefore $P(c|f)$ simplifies to

$$P_{NB}(c|f) = \frac{P(c)\prod_{i=1}^{n} P(f_i|c)}{P(f)}$$

When comparing $P_{NB}(c|f)$ for a given f, $P(f)$ is a constant which can be ignored. Therefore the definition of Naive Bayes Classifier used in (Pang et al., 2002) is as follows.

Definition 2 (Naive Bayes Classifier) *Let C be a set of possible classes. The Naive Bayes classifer takes a feature vector f and returns $\hat{c}$ which satisfies the following.*

$$\hat{c} = \arg\max_{c \in C} P(c|f) = \arg\max_{c \in C} P(c) \prod_{i=1}^{n} P(f_i|c)$$

Pang et al. (2002) computes $P(c)$ and $P(f_i|c)$ using relative-frequency estimation with add-one smoothing from the training set. That is,

$$P(c) = \frac{N_d(c)}{\sum_{c \in C} N_d(c)}$$

$$P(f_i|c) = \frac{N_c(f_i) + \kappa}{N_f(c) + \sum_{w \in V} \kappa}$$

where

- $N_d(c)$ is the number of documents in class c
- $N_c(f_i)$ is the frequency of a feature f_i in all the documents in class c
- $N_f(c)$ is the number of features in all the documents in class c
- κ is the smoothing constant
- V is the set of vocabulary

Despite of strong independence assumption, Naive Bayes classifier is known to perform fairly well on text classification tasks (McCallum et al., 1998), which together with the ease of implementation makes it a popular baseline system.

3 Scoping

In this section, we describe scoping heuristics I experimented with. In this report, I refer to terms and tokens which are within the scope of negation as *negated term* and *negated tokens*, respectively.

One straight-forward approach is to use punctuations as an indicator of the end of a scope. This is based on the idea that punctuations, such as commas or quotation marks, are often used to mark the end of some "logical unit". Clearly, this is an approximation and one may easily think of cases where such heuristics would fail (e.g. *I am not, you know, very happy*).

Intuitively, a syntactic structure of the sentence should be more informative when determining the scope of the negation. In this experiment, I utilise *dependency structure* of the sentence, as it was previously shown effective by Jia et al. (2009). A dependency structure is a graph consists of *dependency relations* drawn between a head and its dependants. For instance, *I swim.* could be represented by a subject relation from *swim* to *I*.

The following describes four heuristics I evaluated. Each heuristic is presented with an example, where negation terms are marked with *<negation>*, and computed scopes of negation are marked with *[negated terms]_*.

punc

With this heuristic, proposed by Pang et al. (2002), all the words after the negation term is regarded as negated, up to a punctuation.

e.g.) *I was <not> [happy with what the director did]_, but I liked the plot.*)

after_x

With this heuristic, proposed by Hu and Liu (2004), x words after the negation term is regarded as negated up to a punctuation which normally marks the end of a sentence, such as a period or exclamation mark.

e.g.) *I was <not> [happy, with, the, movie]_!.* with $x \geq 4$.

dir_dep

With this heuristic, only the head of the negation term in the dependency structure that represents the sentence is marked as negated.

e.g.) *I <barely>, though I took a coffee in the middle, [finished]_ it.*)

head_obj

With this heuristic, the head of the negation term and its subtrees, rooted at its object or its complement are negated.

e.g.) *I did <not> [think it was amazing]_, but people clearly loved it.*

In this sentence, the head of *not* is *think* and *think it was amazing* is a clausal complement of *think*)

4 Method

This section describes various implementation choices I made in the experiment.

4.1 Tokenisation

A document is first split into sentences, which get tokenised separately. The first letter of the sentence is converted to lower case, and the sentence is split at the white spaces which results in the initial list of tokens. Then, 11 rules are applied sequentially to each token in the list. Each rule is associated with a regular expression, and all the rules whose pattern matches the token are applied. Most rules split the token at a punctuation which becomes its own token (e.g. "/" in "love/hate" → ["love", "/", "hate"]), with the following exceptions.

- If the token contains non-alphabetical character, it is not split at hyphens. (e.g. "long-term" → ["long", "-", "term"], "3-G21" → ["3-G21"])
- If the token is suffixed by "'ll", its prefix and "will" are added. (e.g. "you'll" → ["you", "will"])
- If the token is suffixed by "n't", its prefix and "not" are added. (e.g. "don't" → ["do", "not"])
- If the token is suffixed by "'ve", its prefix and "have" are added. (e.g. "you've" → ["you", "have"])
- If the token is suffixed by "'s", its prefix and "'s" are added. (e.g. "aunt's" → ["aunt", "'s"])

4.2 Cross Validation

I use k-fold cross validation to evaluate and compare the performance of models, where models are tested on each fold after being trained on the remaining $k-1$ folds. The list of all the documents are *randomly* partitioned into k equal-sized fold.

4.3 Symbolic Approach

4.3.1 Sentiment Score

I use the magnitude of 1 for a word marked "strong" and 0.5 for "weak". For words marked "neutral" or "both" I assign the sentiment score of 0.

4.4 Naive Bayes Classifier

4.4.1 Features

I use unigrams obtained from the tokens from the tokenizer described in Section 4.1, without any further processing or filtering.

4.4.2 Smoothing

I evaluate Naive Bayes classifier both with and without smoothing. I chose 0.2 for the smoothing constant.

4.5 Negation

4.5.1 Negation Terms

The list of negation terms I used in this experiment is *not*, *never*, *no*, *n't*, *less*, *without*, *barely*, *hardly* and *rarely*. These are the most common negation words, also utilised in (Jia et al., 2009).

4.5.2 Parser

I use *spaCy* (Online, 2017a) in order to compute the dependency structure of each sentence. Though spaCy provides its own tokeniser, the tokens which are computed from the tokenizer described in Section 4.1 are used. I first experimented with Stanford Parser (Online, 2017b). However, it runs extremely slow and was not

Table 1: Classification Accuracy

Binary	Weighted	UNB	NB
61.9%	64.2%	50.4%	80.1%

Table 2: Significance Results

	Binary	Weighted	UNB	NB
Binary				
Weighted	==			
UNB	<<	<<		
NB	>>	>>	>>	

appropriate for this task where the dependency graphs needed to be produced for each sentence in all the documents.

4.5.3 Negated Tokens in Symbolic Approach

The score for a negated token is obtained by multiplying -1 to the score for its normal form. For instance, if a "weak positive" token is marked as negated, then it is assigned with a score of -1 in the binary symbolic approach, and -0.5 in weighted symbolic approach.

4.5.4 Negated Tokens in Naive Bayes Classifier

For Naive Bayes classifier, I experimented with two methods of treating negated tokens.

`simple`
When encountering a token marked as negated, a new token is created by prefixing the term it represents with "not_", and increment the count for the new token. For example, if a token "good" in a positive document is marked as negated, then a count for "not_good" is incremented for the positive class.

`augmented`
As stated before, it was plausible that the number of negated tokens would be too small to improve the classification. In order to mitigate this problem, we experimented with the approach proposed by Narayanan et al. (2013), where all the tokens in a document of a class contributes to the frequency of their negated form in the opposite class. For instance, if a token "good" appears in a positive document, then the count for "not_good" is incremented for the negative class.

5 Result

5.1 Baselines

The table 1 shows the classification rate for each classifier. Binary stands for a binary symbolic approach, Weighted for a weighted symbolic approach, UNB for Naive Bayes classifier without smoothing and NB for Naive Bayes classifier with smoothing. Note that the classification rate for the Naive Bayes classifier is the mean of 10 results from each fold in 10-fold cross validation.

The table 2 shows the result of two-tailed sign test. Each cell contains a symbol for the result of test performed between classifiers read from the row and the column. $==$ marks $p >= 0.01$, $>$ or $<$ marks $0.01 > p >= 0.005$ and $>>$ or $<<$ marks $0.005 > p$. The inequality should be read as if the classifier in the column is on the left and the classifier in the row is on the right. For instance, $>>$ in the cell with the column NB and the row Binary should be read as: "Naive Bayes classifier with smoothing is better than binary symbolic approach with a high statistical significance."

From this, I conclude that the Naive Bayes classifier with smoothing is significantly better than the symbolic classifiers, whose performance is statistically indistinguishable. Naive Bayes classifier without smoothing is significantly worse than other classifiers. This is because without smoothing all the document is assigned with 0 probability for a class if it contains a word which does not appear in the documents of the class in the test set. As it often happens that null probability is assigned to a document for both positive and negative class, one of the class is constantly chosen, resulting to the classification accuracy which is close to the proportion of the documents of the chosen class(i.e. 50% with this dataset). I exclude Naive Bayes classifier without smoothing from the next sections.

5.2 Negation

In this section, I evaluate the effect of addressing negations present in the document. Specifically, I look at the classification accuracy of the classifiers which takes negation into account computed from 10-fold cross validation, and compare them with the corresponding baseline using two-tailed sign test.

Table 3: Classification Accuracy of Symbolic Approaches with Negation

	punc	after_1	after_3	after_5	dir_dep	head_obj
Binary	66.7%	63.9%	65.5%	65.8%	63.0%	64.8%
Weighted	67.9%	66.1%	66.2%	67.0%	66.0%	65.4%

Table 4: Significance Results of Symbolic Approaches with Negation

	punc	after_1	after_3	after_5	dir_dep	head_obj
Binary	==	==	==	==	==	==
Weighted	↓↓	↓↓	↓↓	↓↓	↓↓	↓↓

5.2.1 Symbolic Approaches

Table 3 shows the classification accuracy of the symbolic approaches which takes negation into consideration. The column indicates the type of symbolic approach, and the row indicates the heuristic used to compute the negation scope. Similarly, Table 4 shows the result of the statistical test carried out to compare each classifier to the corresponding baseline. In this table, $\uparrow\uparrow$ and $\downarrow\downarrow$ indicates with significance $0.005 > p$ an improvement and deterioration respectively, $\uparrow$ and $\downarrow$ indicates with significance $0.01 > p \geq 0.005$ an improvement and deterioration respectively and $==$ indicates no change with significance $p \geq 0.01$. Though there is a small improvement in the classification accuracy, I conclude that it is not statistically significant.

5.2.2 Naive Bayes

Table 6 shows the classification accuracy of the Naive Bayes classifier with negations considered. The column indicates the strategy used to treat negated tokens, and the row indicates the heuristic used to compute the negation scope. Similarly, Table 4 shows the result of the statistical test carried out to compare each classifier to the baseline Naive Bayes classifier with smoothing. With `simple` strategy, all the figures are lower than that of the baseline, aside from one for `punc`. However, none of changes is statistically significant. With `augmented` approach, all the figures are lower than that of the baseline, and the sign test shows that taking negation into account in this manner harms the performance of the classifier.

5.3 Analysis

For either baseline, it is hard to conclude whether a more sophisticated scoping heuristic will lead to a significant improvement to the baseline. In fact, the only heuristic which was close to give an improvement to a baseline was `punc`, whose improvement on the binary symbolic approach had a significance of $p = 0.035$, and it is arguably the most crude heuristic of all. This suggests that, even if the scoping was computed accurately for each negated term, we might not be able to improve the baseline systems, perhaps because negation occurs too rarely in the documents. As a result, the negated terms are overpowered by the normal terms and can not contribute to the classification. This would affect the Naive Bayes classifier more than the symbolic approaches, as not all words in the document in the symbolic approach contributes to the classification.

`augmented` was expected to offset this problem. However, as we saw it only degraded the performance. This is perhaps because many words which have no disambiguation capability created noise when they are negated, and those with disambiguation capability did not occur frequently enough to influence the overall classification. This could be potentially be solved by using a list of *stopwords*, common words which do not normally have disambiguation capability, to filter the tokens. I confirmed this hypothesis by comparing `augmented` Naive Bayes classifier which ignores 32 words in a list presented in Online (2015). Table 7 shows the classification accuracy of the `augmented` Naive Bayes classifier with stopwords, and Table 8 shows the significance results when it was compared with `augmented` Naive Bayes classifier without stopwords. As can be seen, a list of stopwords gives a significant improvement to `augmented` Naive Bayes classifier. Though this does not give an improvement to the baseline, it gives some insights to the underlying problem. Note that utilising stopwords did give a statistically significant improvement to other classifiers.

One may avoid this problem which comes from

Table 5: Classification Accuracy of Naive Bayes Classifier with Negation

	punc	after_1	after_3	after_5	dir_dep	head_obj
simple	80.9%	79.8%	79.9%	79.4%	79.6%	80.0%
augmented	59.5%	61.7%	60.9%	59.6%	61.6%	60.2%

Table 6: Significance Results of Naive Bayes Classifier with Negation

	punc	after_1	after_3	after_5	dir_dep	head_obj
simple	==	==	==	==	==	==
augmented	↓	↓	↓	↓	↓	↓

the relatively low frequency of negation terms in a document by training a Naive Bayes classifier to classify the polarity of a sentence, rather than a whole document. The polarity of the document can be determined by aggregating the polarity of the sentence by, for instance, counting how many sentences were classified into each class. I did not take this approach as I can not assume that the sentences which are in the document of a positive class are all positive; this requires a dataset of sentences which are labeled with the polarity.

Negation is just one example of *valence shifters* terms that can change the semantic orientation or intensity of another termKennedy and Inkpen (2006). A difficulty with the treatment of valence shifters in symbolic approach is in determining their exact effect on other termsKennedy and Inkpen (2006). Recall that in this experiment, the sign of the semantic score of a term was flipped. This does not perfectly capture, for example, the effect of *not* in *not excellent*; *excellent* represents a strong positive sentiment, but *not excellent* does not convey a strong negative sentiment.

All in all, treatment of negation in the polarity classification is not straight-forward and perhaps is impossible to solve by a good scoping heuristics alone.

References

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*, pages 168–177. ACM, 2004.

Lifeng Jia, Clement Yu, and Weiyi Meng. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1827–1830. ACM, 2009.

Alistair Kennedy and Diana Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational intelligence*, 22 (2):110–125, 2006.

Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

Vivek Narayanan, Ishan Arora, and Arjun Bhatia. Fast and accurate sentiment classification using an enhanced naive bayes model. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 194–201. Springer, 2013.

Online. Ranks nl stopwords. `http://www.ranks.nl/stopwords`, 2015. Accessed:2017-01-03.

Online. spacy. `https://spacy.io`, 2017a. Accessed:2017-01-04.

Online. Stanford parser. `http://nlp.stanford.edu/software/tokenizer.shtml`, 2017b. Accessed:2017-01-03.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural lan-*

Table 7: Classification Accuracy of `augmented` Naive Bayes Classifier with Stopwords

punc	after_1	after_3	after_5	dir_dep	head_obj
66.7%	69.1%	69.3%	70.0%	70.3%	67.6%

Table 8: Significance Results of `augmented` Naive Bayes Classifier with Stopwords

punc	after_1	after_3	after_5	dir_dep	head_obj
↑↑	↑↑	↑↑	↑↑	↑↑	↑↑

guage processing, pages 347–354. Association
for Computational Linguistics, 2005.