Sentiment Classification using Language Models and Sentence Position Information

Sunil Khanal
sukhanal@stanford.edu

CS224N Final Project
Spring 2010

Abstract

In this project, I aim to develop a sentiment classification system which classifies a given document as having either positive or negative sentiment. I build up on the previous work done in this area by Pang, Lee and Vaithyanathan (EMNLP 2002). I use techniques such as classifying individual sentences which provide a significant improvement over what they have been able to achieve previously. I also use in my project some of the suggestions they provided as future directions in their original work. I provide a set of simple but significant features that can be added to the baseline model in order to improve performance.

1. Introduction

The subject of topic classification in NLP is not a new one. With the increase in unstructured data over the internet, there is increasing user content that may or may not include metadata which would help classify the content in a way desirable for a problem at hand. Topic classification can span a range of classes, such as categorizing a document as financial, political, sports or health, or identifying a language a given document is written in.

The other important category that we might be interested in is that of sentiment classification, i.e. what kind of sentiment a given piece of data suggests. Sentiments are inherently subjective issues, and there might be various kinds and degrees of sentiments in any given data such as like or dislike, anger, hatred, satire, etc. Producing a system that can perform a detailed sentiment analysis for these kinds of sentiments would be extremely hard, but it would be an interesting result if it could be done. However, for most purposes, classifying a document as having an overall positive or an overall negative sentiment is often useful at times. Examples include online user reviews of commodity items, services, or service providers. Movie reviews are other important forms of documents that can serve as an aid for building sentiment classifiers, and which I am using in my project. Since it is relatively easy to obtain movie reviews online and also to get a general sense of whether a review is positive or negative from the ratings provided by the user, using movie reviews in building and training a sentiment classifier becomes a reasonable undertaking. In this project I build up on the previous work done on sentiment classification, and use the insights provided in the previous works to make an improvement on the sentiment classifier performance.

2. Previous Works

My project builds up on the works done by Bo Pang and Lillian Lee and Shivakumar Vaithyanathan in proceedings of EMNLP 2002. I also refer to works done by Nathan Sakunkoo

as one of the past CS224n final projects. Pang, Lee and Vaithyanathan used different classification techniques like Naïve Bayes, SVM, and Maximum Entropy models. Before doing so, they listed a simple approach of labeling a movie review as negative or positive depending on various keywords a review may possess. They used positive keywords such as: *love, wonderful, best, great, superb, still, beautiful* and negative keywords such as: *bad, worst, stupid, waste, boring, ?, !* and made classification decisions based on whether and how many of these keywords are contained by a particular movie review. They use this human intuition kind of model as a baseline in comparing the performance improvement they get by employing machine learning techniques. They also use different language models in order to make improvements in the performance of their system. In addition, they try to take various approaches such as using POS tagging, and using the position information of a word in a sentence. One main conclusion they reach is that the most significant factor for performance improvement is the unigram model and any other features subsequently added provides little improvement in performance in comparison[1].

Pang, Lee and Vaithyanathan suggest using a sophisticated "discourse analysis" in order to tackle the problem of "thwarted expectations" in order to improve performance of the system. A thwarted expectation occurs when a series of sentences conveying a different sentiment precede a sentence providing the intended sentiment, such as *"This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can't hold up"*[1]. They also suggest finding the focus of sentence, i.e. whether or not a sentence is talking about the movie being discussed would be helpful.

Sakunkoo also takes these suggestions and builds a "bag of sentence" model[2], in which he classifies each sentence as positive and negative. In addition, he does a subjectivity analysis by training on subjective and objective sentences, and adds to the classifier results obtained by such subjectivity summarization.

3. Data used

The data for the project was obtained from http://www.cs.cornell.edu/People/pabo/movie-review-data/ . I use the 1000 positive and 1000 negative movie reviews as well as the 5331 positive and 5331 negative sentiment tagged review snippets. The movie reviews were stripped off of any explicit rating information for the corresponding movie, so that a classifier would not use that information for sentiment classification inadvertently. The snippets were taken from Rotten Tomatoes webpage, with "fresh" reviews considered positive and "rotten" reviews considered negative.

4. Methods and Approach

I build up on the Naïve Bayes classification models presented by Pang, Lee and Vaithyanathan. I liked the "bag of sentence" idea of Sakunkoo, so I use that idea as well in the project. I however, don't use subjectivity analysis and restrict language models to bigrams and unigrams, but show that equally good or better results can be obtained by combining various simple features and observations. In addition, I use the positional information of the sentences as opposed to the position of words in order to figure out when the reviewer is talking about the movie. I combine this positional information with the focus of each sentence, as suggested by Pang, Lee and

Vaithyanathan[1], which does offer a significant improvement on performance. In addition, I use negation tagging for words appearing between a negation word as a punctuation mark (Technique of Das and Chen 2001, adapted by Pang, Lee and Vaithyanathan). Finally, I use the porter stemmer algorithm to stem verbs and discriminating words.

Hence there are two major parts in classifying a review. First I train a language model for positive and negative reviews using unigram and bigram models. The training is quite simple – just remembering the bigrams and unigrams the classifier has encountered during the training phase. Then when a review is submitted for classification, the classifier assigns class to the review using:

$$c^* = \arg\max_c P(c|d)$$

where

$$P(c|d) = P(c) * \text{Product\_for\_all\_i}(P(f_i|c)^{n\_i(d)} / P(d)$$

Where f_i stands for a feature, which can be a bigram or a unigram present in the document, and n_i(d) indicates the frequency of the feature.

The language model provides one classification decision for a given review. Then, I use a different sentence classifier that is trained on the 5331 positive and 5331 negative sentences. The sentence classifier is trained using the same bigram and unigram language models as the review classifier. In addition, the sentence classifier uses other techniques such as negation tagging and POS tagging (I eventually remove POS tagging due to no contribution on performance).

I run each sentence of the review to obtain a decision of "positive" or "negative", as well as the log probability computed by the sentence classifier for each sentence. Then the sentences are used to "vote" the review as negative or positive on the basis of their probability scores. After that the decisions of voting from the sentence classifier and the review classifier are combined together using optimum weights and an overall decision about the sentiment of the review is made.

5. Results and testing

I improved the classifier gradually by doing a series of experimentation and performance analysis of the work the classifier was doing. I also chose features for the classifier very carefully and by analyzing what kind of impacts they would have. I used 10-fold cross validation in order to measure accuracy. For each fold, I divided the reviews into 90% training set and 10% testing set. So after the 10^{th} fold every review will have been tested. A summary of results with various features is provided below:

	Features	**Accuracy**
1.	unigrams	81.3
2.	bigrams	77.3
3.	unigrams + bigrams	81.7

4.	sentences	73.8
5.	unigram + bigram + sentences	87.2

Considering the language model, we see that addition of bigrams adds little to the improvement in performance of the classifier. This was what was observed by Pang, Lee and Vaithyanathan. But when we add the information provided by the sentence classifier, we see a significant performance improvement of 5.5%. Since adding sentence information to the language model in order to improve performance of the classifier is the main focus of my project, I will explain in more detail how I developed and improved the sentence classifier and how I integrated it with the language model.

5.1 The sentence classifier

The sentence classifier was run on the 5331 negative and 5331 positive sentences. Out of those I separated 500 positive and 500 negative sentences for testing the accuracy. Hence the Sentence classifier was tested on the remaining sentences. Here is a summary for the accuracy of the sentence classifier with various features:

	Feature	Accuracy
1.	unigram + bigram	80.3
2.	unigram + bigram + POS	78.9
3.	unigram + bigram + negation tagging	81.7

As we see, using parts of speech (POS) tagging made the performance worse in general. Using negation tagging, on the other hand, improved the performance of the system. Negation tagging was done in a way suggested by Pang, Lee and Vaithyanathan, which is to add a negation tag to words between a negation word and a punctuation mark. Some of the negation words used are:

```
not, isn’t, didn’t, doesn’t, hasn’t, shouldn’t, can’t, won’t
```

For example, a sentence “In the end, there isn’t much to it.” gets tagged as:

```
in the end , there isn't much_<NEGATIVE> to_<NEGATIVE> it_<NEGATIVE> .
```

But just applying negation tagging to all the words might be tagging a bit too much, especially since the movie reviews aren’t strictly grammatical and may not use punctuation correctly. Besides, even if punctuation was used correctly, it’s likely that we’re tagging a bit too many words negatively, such as:

```
it's not that_<NEGATIVE> kung_<NEGATIVE> pow_<NEGATIVE> isn't_<NEGATIVE>
funny_<NEGATIVE> some_<NEGATIVE> of_<NEGATIVE> the_<NEGATIVE> time_<NEGATIVE>
--_<NEGATIVE> it_<NEGATIVE> just_<NEGATIVE> isn't_<NEGATIVE> any_<NEGATIVE>
funnier_<NEGATIVE> than_<NEGATIVE> bad_<NEGATIVE> martial_<NEGATIVE>
```

```
arts_<NEGATIVE> movies_<NEGATIVE> are_<NEGATIVE> all_<NEGATIVE> by_<NEGATIVE>
themselves_<NEGATIVE> , without all oedekerk's impish augmentation .
```

Tagging words such as `time, the, of, martial, arts` etc. as negative doesn't look very right. Hence, I experimented a little bit, and found out that tagging three words following a negation word gives the optimum result:

```
it's not that_<NEGATIVE> kung_<NEGATIVE> pow_<NEGATIVE> isn't_<NEGATIVE>
funny_<NEGATIVE> some_<NEGATIVE> of_<NEGATIVE> the time – it just isn't
any_<NEGATIVE> funnier_<NEGATIVE> than_<NEGATIVE> bad martial arts movies are
all by themselves, without all oedekerk's impish augmentation .
```

We see that the words such as `time, funny, funnier` which we want negated get negated while words such as martial arts, which we don't want negated remain untagged.

The change in performance due to this factor can be seen below:

	Feature	Accuracy
1.	no proximity	80.9
2.	use 3 as proximity	81.7

The improvement, although small, is still significant.

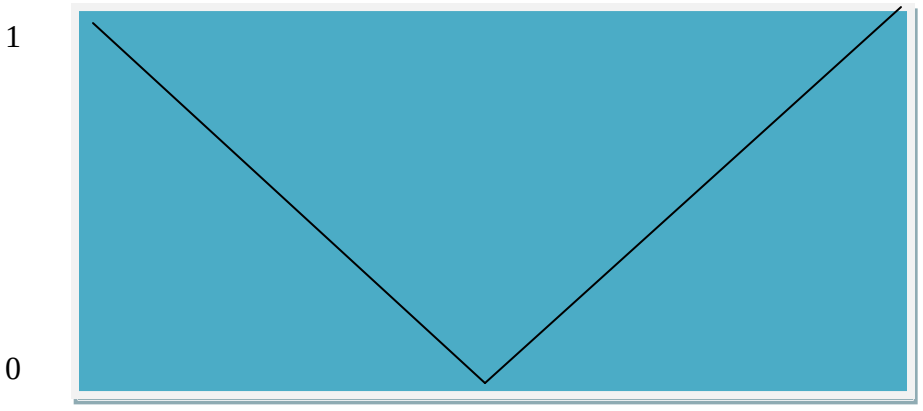
5.2 Integrating sentence classifier with Language model

While integrating the sentence classifier with the language model, I took a number of factors into consideration. Log probability of the sentiment class provided by the sentence classifier, position of the sentence in the review, and the focus of the sentence are those factors. I summarize the results of review classification below (these results are combining the sentences with language model):

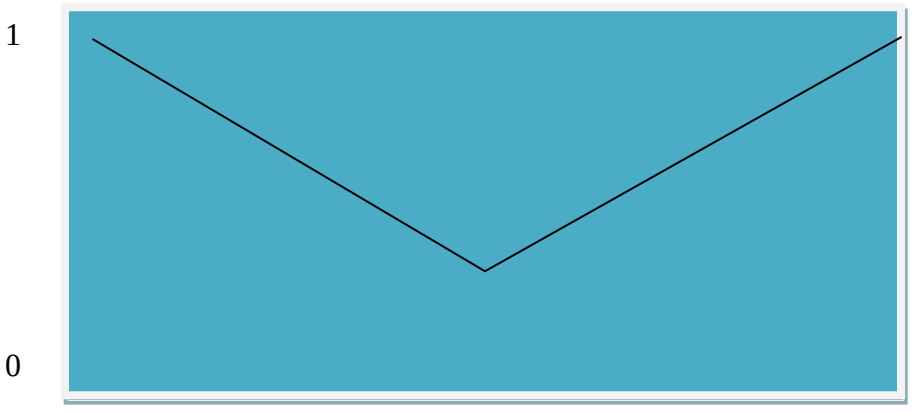
	Feature	Accuracy
1.	use 0/1 as voting	82.4
2.	use log probabilities returned by sentence classifier	84.3
3.	log probabilities + position	87.2
4.	log probabilities + position + focus	87.2

The first column displays results obtained when a 0/1 voting was used. A negatively classified sentence says the review is negative and a positively classified sentence says the review is positive. Then the majority vote is used to classify the review. Obviously, this scheme provides the worst results because some sentences might be highly negative or highly positive, and other could be moderately so. Hence I used the actual scores of positivity and negativity computed for the sentence and added that for all sentences to classify the review. This gave a considerable 1.9% improvement in performance.

Then I took a further step of weighting the sentence by their position in the review. As Pang, Lee and Vaithyanathan suggest, the reviewer might write their first impressions about a movie, and then write about the plot, and then provide and/or justify their impressions[1]. This translates roughly into providing more weights to sentences that are towards the beginning and end of a review. To take this observation into effect I initially used the following linear weighting scheme:



Here the left vertical axis represents the weight to the sentence and the bottom horizontal axis the position of the sentence in the review. But this kind of relationship provided much lesser weight to the sentences in the middle than I intended to. Hence I reduced the slope of the lines, providing a picture roughly like:



Note that the lowest score a sentence can have in the top graph is 0 (the sentence at the middle) while all sentences have non-zero scores for the bottom graph. As I expected the impacts of the change is pretty drastic:

	Feature	Accuracy
1.	Steep slope, lines touching bottom (top graph)	81.2
2.	Gentler slope, lines meeting upwards (bottom graph)	87.2

I also tried to incorporate "focus" of a sentence into classification, as suggested by Pang, Lee and Vaithyanathan. Focus here means whether a given sentence is talking about a movie or not. In order to find the focus I tried two approaches: checking if a sentence contains words such as `this movie, the plot, the actors`, etc, and checking if a sentence consists of a movie name. I used a list of movies from http://www.widemovies.com/wsmovies/title_list.html. But using such focus analysis did not add anything to the classifier. This probably happened because focus finding can be difficult and require more sophisticated techniques. One simple reason why checking if a sentence contains a movie name failed was that there are a lot of movies whose names are very common, such as Airport, Election, The End, Evolution, Fallen, etc. Sophisticated analysis using POS tagging and NER could provide more help in this sector.

5.3 Stemmer

Last but not the least; I used the Porter Stemmer algorithm to stem words. I restrict the stemming to verb like words such as words ending on `es, ne, en`, and descriptive words like `love, great, beautiful`, etc. The stemmer provides an improvement of 0.5% on accuracy.

6. Discussion and future directions

As is demonstrated by the results I obtained, Naïve Bayes proves a simple yet very strong classification model for sentiment recognition, and more broadly document classification. However, the 'bag of words' approach it takes can be supplemented by other features, adding significantly to the performance. In this case, using classification of sentences in addition to the Naïve Bayes model increased accuracy of the system by 5.5%, which is a significant figure. But we need to note that the sentence classifier was also built on Naïve Bayes. There might be further improvement on the performance if we use other machine language tools like Maxent classification, or SVM. But however, the project suggests that coming up with features to use in classification can be more determinative of system performance compared to the machine learning tool used.

To further improve sentiment classification of movies, or other data having a measurable sentiment value, we can add more features or improve on the existing features. Extending the concept of using sentences, we can divide the review into paragraphs and provide different weights to different paragraphs. Or, as was indicated earlier, using better techniques to extract the focus of a sentence and providing sentences weights based on whether they are on topic can be another important improvement. For doing this, we might try having a broad classification scheme for each sentence by associating it with what the entities it is about (such as is a sentence about a person, place, time or a date), is it interrogative, is it present/past/future, singular/plural, does it have negative words and how they are placed, and what is the relationship between objectives and adjectives (such as "This appalling movie sucks" has (movie → sucks, movie →

appalling)). Once we learn such associations, it might be possible to learn about the interrelation (for example while training the classifier could learn that (movie → sucks) means a negative sentiment).
Using some form of semantic understanding will further improve performance of the system. If we can group words into hierarchies and classify words at the top of the hierarchy as providing a positive or negative sentiment, it might assist on figuring out the sentiment conveyed by a previously unseen word.

References:

1. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan, 2002. **Thumbs up? sentiment classification using machine learning techniques.** (Proceedings of EMNLP).

2. Nathan Sakunkoo, 2007 **Improving Sentimental Classifications Using Contextual Sentences.** (CS 224n final project)

3. M.F. Porter, 1980, **An algorithm for suffix stripping**, Program, PP 130-137,