Using Semantics for Sentiment Analysis

Daniel Loman, Sandeep Vanga, Selman Tunc Yilmaz

March 16, 2015

Abstract

Sentiment analysis is a well known problem in text mining and has many important applications. Here, we are attempting to categorize IMDB movie reviews as positive or negative, as part of a Kaggle competition named "Bag of Words meet Bag of Popcorn". We used two different approaches named Bag of Words and Word2Vec to solve sentiment analysis problem. As Word2Vec approach is not inherently tuned for sentiment analysis, we brought these two approaches together. Unfortunately, even the hybrid algorithms are being outperformed by traditional bag of words model. Bag of Words model with tfidf weighted uni-grams and bi-grams with regularized logistic regression yields 0.966 AUC and it's the best algorithm by far. We hope to improve the hybrid approaches in future.

1 Introduction

Sentiment Analysis is an example of text classification problem which is a well known research problem in text mining and NLP community. It plays a central role in many applications, e.g, document retrieval, web search, spam filtering. Many algorithms such as logistic regression or Naive Bayes that are traditionally used for sentiment analysis are implementations of the Bag of Words approach. However these algorithms do not take the order of the words into consideration, neither the semantic similarity between the words. Though this problem is taken care of by using n-grams, these methods suffer from issues such as sparsity and the high dimensional feature set.

In recent times, certain methods are proposed to extract the meaning of words or sentences. There are two such approaches:

1) Word2Vec: Unsupervised neural networks model to understands the semantics of words. Each word is represented by a vector. Similar words are close to each other in this vector space.

2) Doc2Vec: Unsupervised neural networks to understands the semantics of sentences. Each word as well as each review is represented by a vector. Similar words and reviews are close to each other in this vector space.

We are using these methods to perform sentiment analysis of IMDB movie reviews as part of a Kaggle competition named "Bag of Words meet Bag of Popcorn (https://www.kaggle.com/c/word2vec-nlp-tutorial).

2 Related Work

The ideas for using word vectors and paragraph vectors for sentiment analysis we borrowed from [4] and [3] respectively. In [3], averaging word vectors is mentioned as a way of using word vectors to represent each review. The authors provide a comparison of algorithms where vector averaging performs worse than weighted bag-of-bigrams which is inferior to using paragraph vectors. Bot Word2Vec and Doc2Vec are implemented using the gensim package in [2]. Doc2vec implementation in gensim is very preliminary and has not been tested enough by others.

3 The Problem

Data set contains 50,000 total reviews, of which 25,000 belong to a labeled training set and 25,000 belong to a test set. In training set each review is either labeled as positive or negative review. Our aim is to estimate the sentiment of reviews in the test set. There is also an unlabeled training set containing 50,000 unlabeled reviews that can be used to train models for Word2Vec and Doc2Vec.

4 Preprocessing

In the training set, the positive and negative classes are evenly balanced and contain roughly the same amount of words in each review. Also, we did not observe any noise in the labels.

The review text contain significant noise such as punctuation characters, numbers and html tags. So, the first step is to tokenize the reviews into words while eliminating such unnecessary content. The important steps in tokenization are mentioned below.

1) Words are converted to lower case.

2) Remove commonly occurring words known as stop words from the word corpus.

3) Add a negative tag "NEG" to all the words which follow the negative words such as not, did not, never until we hit a punctuation.

4) Stem the words to map different forms of same words to single token.

In order to train the Word2Vec vectors, the word tokens after the steps 1-4 are put together and re-tokenized as sentences.

5 Algorithms

As mentioned earlier there are two broad approaches to solve sentiment analysis problem:
1) Bag of Words approach
2) Word vectors approach

In our Bag of Words approach, we modified the standard Bag of Words model so that each feature was represented by the word's TF-IDF score in the document rather than its count frequency. We considered both words and bigrams, and reduced features to include only those which appear more than twice in the corpus. This left us with a total of roughly 350,000 features. We were still left with very sparse data so we represented it with a sparse matrix. In sklearn this limited our options for machine learning algorithms, and we found that logistic regression yielded

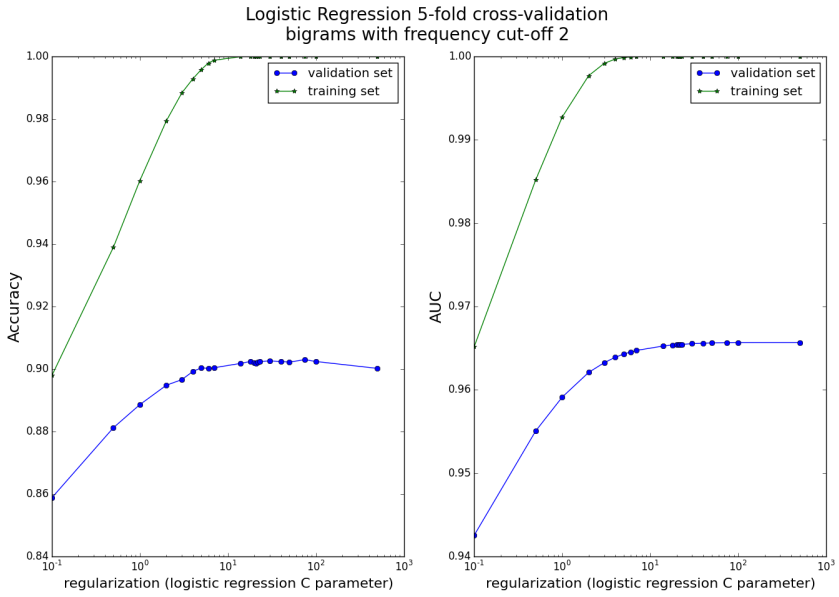

Figure 1: 5-fold cross-validation for logistic regression classifier with the tf-idf weighted bigrams.

our best results. Our final model gave us an AUC of .96, which is our best model to date. The cross-validation curves are shown in Fig. 1.

The second approach we took was using Word2Vec, in the gensim package. Word2Vec is a neural networks model that creates features for classification by extracting meaning from words. Word2Vec has 2 main parameters: number of features in each word vector, and the minimum word count threshold (a word must appear this many times in the corpus to be considered in the model). After running 5 fold cross validation we determined that the optimal parameters are 300 features in a vector and a minimum number of words threshold of 40.

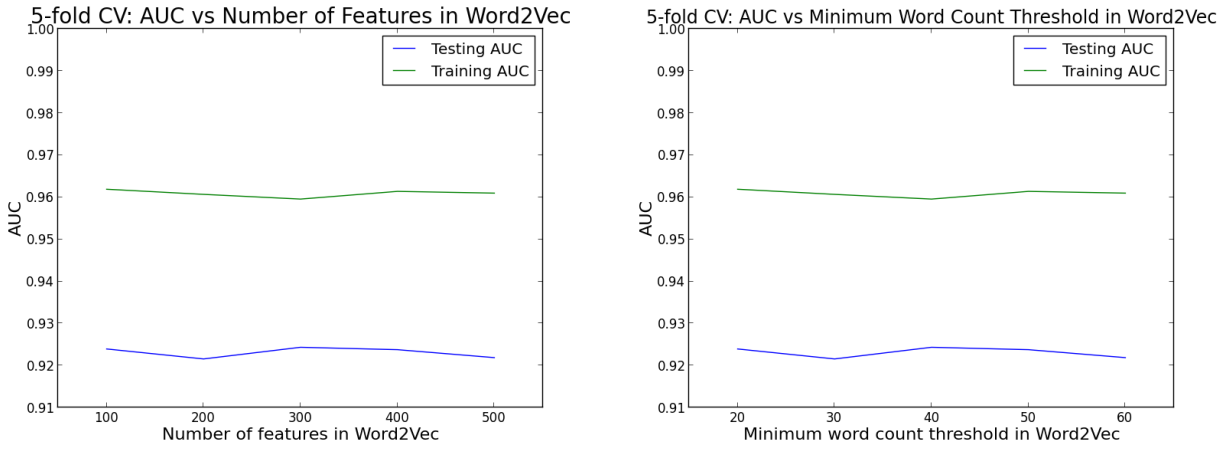

Figure 2: 5-fold cross-validation for logistic regression classifier with the tf-idf weighted bigrams.

Within Word2Vec we took 2 separate approaches. The first approach was to average each word vector within each review, to get an single feature vector representing the meaning of the review. We determined that rather than evaluating just the mean of the word vectors, it would be best to weigh each word based on its importance within the corpus, and take a weighted average. For this we used a modified TF-IDF weight. The modified TF-IDF weight of each word is the absolute difference between its mean TF-IDF score in positive reviews and its mean TF-IDF score in its negative reviews. This approach would give heavy weight to words that are either influential in positive or negative reviews, and give light weight to more neutral words. This weighted approach yielded an AUC of .921, compared to an AUC of .912 in the non-weighted approach. We also tried using feature reduction using the same TF-IDF metric or a Chi Squared approach for extracting word importance, but found no improvement.

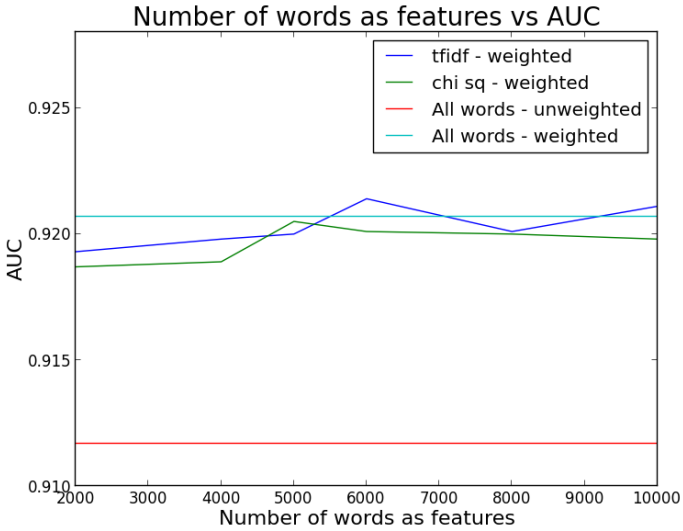

Figure 3: AUC vs Number of words as features for different feature selection approaches

The other approach we tried with Word2Vec was clustering words. Since each word has a unique word vector associated with it, we can cluster the word vectors to find similar words. Some of the clusters are given below.

```
['hebrew', 'speaker', 'fluent', 'hungarian', 'pronounc', 'dialect']
['machin', 'gun', 'shoot', 'bullet']
['support', 'perform']
```

Each review can be represented by these bag of centroids by counting the number of words in the review in each cluster. While constructing the bag of centroids, instead of giving same weight to each word we can weight the word by its tf-idf score. Once each review is represented by bag of centroids, we can use any supervised machine leaning algorithm to predict sentiments. We applied different machine learning algorithms such as naive-bayes, random forests and logistic regression and tuned the hyper parameters with cross validation. All of these techniques provide more or less similar results.

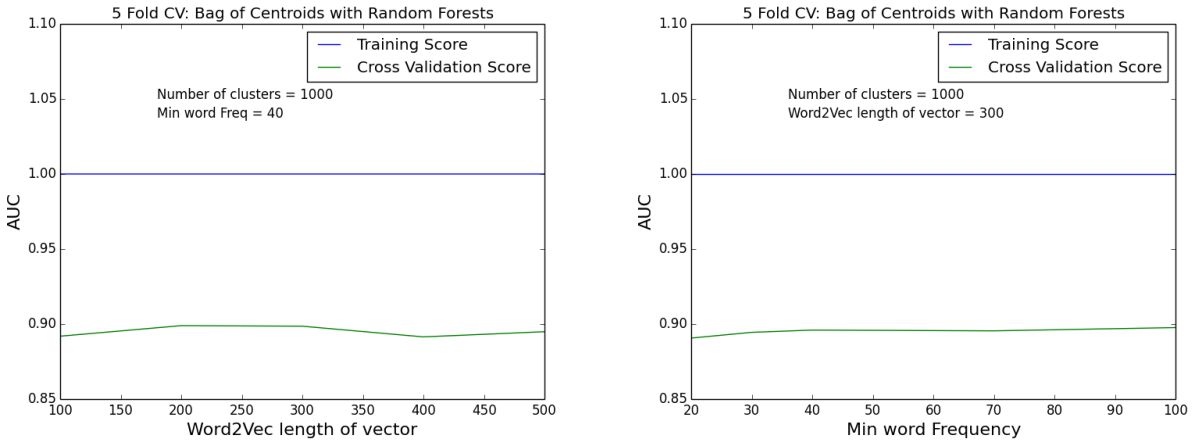

Figure 4: 5-fold cross-validation for word2vec bag of centroids based random forests classifier .

Apart from tuning hyper parameters of these ML algorithms, we also tuned two parameters used in building the Word2Vec model a) length of word vectors b) minimum word frequency. We determined that the optimal parameters are word vector length of 300 and minimum word frequency of 40 as shown in Fig. 4. Another important parameter to be tuned for clustering was the number of clusters. Instead of number of clusters we tuned number of average words per cluster. As the umber of average words per cluster increases, number of clusters decreases. We found that it's optimal to use 3-7 words per cluster as shown in Fig. 5. After tuning all these parameters, clustering approach yielded an AUC of .935.

These approaches have their own advantages and disadvantages, and were unable to match the result of our Bag of Words model. Our aim is to find solutions which can combine these approaches to exploit their strengths and maximize the accuracy.

6 Hybrid Approaches

We used two different approaches to combine traditional bag of words with Word vectors model.

1) Translation: First we build a word vector model using the tokenized word corpus obtained from labeled and unlabeled training sets. We cluster these words using the word vectors into topics. We select a random word to represent the cluster. Then we replace all the words in corpus belonging to a cluster with the word representing the cluster. This step is called "translation". Then we use traditional bag of words approach on the new word corpus obtained from "translation". As word vectors capture meaning of words we replace all the similar words with a single representing word. We have two main advantages using this approach a) Noise in BOW model can be reduced b) Number of features of BOW model can be reduced

We came across few short comings of this approach. Word vectors based clustering tend to group similar words. In some cases they also group two opposite sentiments such as "good" and "bad". This can be avoided by sub clustering each topic into positive or negative by exploiting the

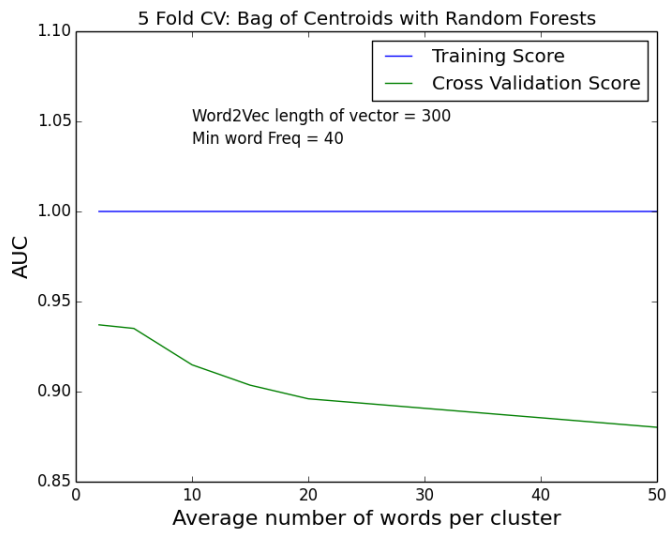

Figure 5: 5-fold cross-validation for tuning number of clusters in word2vec bag of centroids based random forests classifier.

sentiment labels in training set.

2) Document vectors: We tried a preliminary implementation to Doc2Vec in gensim, that creates vectors for each review similar to the word vectors in Word2Vec. Word2Vec algorithm creates vectors for each word by training the model to predict a word in a sentence given the other words. These word vectors are shared by all the reviews that contain those words. In dDc2Vec, we add an imaginary word to each review that represents the review itself. When we train a vector for this imaginary words, it can capture the meaning of the review. Therefore Doc2Vec has the potential of capturing the meaning and sentiment of the review without losing information by aggregating the word vectors.

7 Other Approaches

Here is a quick summary of few other approaches we tried apart from the ones we mentioned in earlier sections.

1) Our best algorithm was the logistic regression using tf-idf scores of words and bigrams. We tried to improve the accuracy with a bagging technique. Since random forest does not work with sparse matrices in sklearn, we wrote our own bagging classifier with logistic regression, that we named as LogisticForest. (The code is in our GitHub repo, [1]). The bagging technique did not improve the accuracy.

2) As another way of using the word vectors, and combining them with tf-idf weighing scheme, we implemented our tf-idf bagging algorithm for each dimension of the word vectors. The output of the logistic regression algorithm for each dimension is a binary output or a probability score for each review, and for each word vector dimension. Therefore the outputs can be organized into a

table format, and a second logistic regression can be applied for the final result. The accuracy was lower than just using the tf-idf scores without using the word vectors.

3) In our clustering model we used TF-IDF scores as weights for each review. We attempted to implement a pseudo-boosting algorithm which would identify the types of reviews that were misclassified, and improve those corresponding training weights accordingly. However, this method did not show any improvement on our standard clustering model.

8 Summary of Results

Our results are summarized in Table 1. The base model is implemented using the decison stump on unigrams with a cut-off frequency of 200 counts.

	Model	number of features	test accuracy	train accuracy	test AUC	train AUC
Tf-IDF based	unigrams_cut-off 2	45657	0.892	1	0.958	1
	bigrams_cut-off2	353082	0.903	1	**0.966**	1
	bigrams with translation	407881	0.902	1	0.962	1
	Base model	4228	-	-	0.53	0.62
Word2Vec based	average	300	-	-	0.912	1
	tfidf weighted average	300	-	-	0.921	1
	bag of centroids	3594	0.823	1	0.914	1
	tfidf weighted bag of centroids	3594	0.866	1	0.935	1

Table 1: Model Comparison

We were able to attain our highest AUC of .966 with our Bag of Words TF-IDF model with a cut off of at least 2 words and bigrams.

It was disappointing to see that one of our simpler models outperformed our more interesting ones, but we were able to see throughout the project that Word2Vec is not optimal for sentiment analysis.

9 Future Work

We approached the sentiment analysis problem in two very different ways. Representing words and bi-grams as TF-IDF scores for each review is a very local representation of the review and does not capture the semantics. On the other hand Word2Vec is a distributed representation, where the word vectors are shared by all the reviews that contain the word. We would like to improve the

implementations of the following techniques that benefit both from local and distributed representation:

1) In literature Doc2Vec outperforms the other existing algorithms for sentiment analysis ([3]). We do not trust the implementation of doc2vec in gensim, while Word2Vec in gensim seems to work reasonably. Since these two algorithms are quite similar to each other, we would like to modify the Word2Vec source code to implement our own Doc2Vec.

2)Using the word vector clusters we created the reviews can be simplified. Each word in a review can be represented by the cluster to which it belongs. The simplification can drastically reduce the noise in our tf-idf representation of words and bi-grams. Our initial implementation of this idea did not improve our accuracy (see table 1: bigrams with translation). We would like to implement the same idea using the word vectors trained on a much larger corpus by Google. Using a more accurate Word2Vec model will improve the quality of the simplification.

3) Minimum word frequency is used to build the vocabulary in current gensim Word2Vec implementation. Instead we can also tfidf based thresholding.

10 Conclusion

Our best model was a Bag of Words model where we represented features as TF-IDF scores rather than word count frequency. We used a logistic regression on this sparse matrix to achieve an AUC of .96. We learned a lot about Word2Vec in this project and the implementation was very interesting. However, we came to realize that it poses some significant drawbacks in sentiment analysis and we were unable to attain the results of our Bag of Words model using Word2Vec.

We are hoping that by combining these two approaches and using Doc2Vec we can improve our already impressive Kaggle ranking.

References

[1] https://github.com/stuncyilmaz/Word2Vec_MovieReviews

[2] https://radimrehurek.com/gensim/

[3] Le, Quoc V., and Tomas Mikolov *Distributed representations of sentences and documents.* arXiv preprint arXiv:1405.4053 (2014)

[4] Maas, Andrew L., et al*Learning word vectors for sentiment analysis.* Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. Association for Computational Linguistics, 2011