Bag of Words Meets Bags of Popcorn

Amir Sadeghian
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
amirabs@stanford.edu

Ali Reza Sharafat
Department of Electrical Engineering
Stanford University
Stanford, CA 94305
sharafat@stanford.edu

Abstract

‘ This problem is selected from one of the Kaggle’s competitions [2]. In this problem, we dig a little ”deeper” into sentiment analysis. Word2Vec is a deep-learning inspired method that focuses on the meaning of words. Word2Vec [3] attempts to understand meaning and semantic relationships among words. It works in a way that is similar to deep approaches, such as recurrent neural nets or deep neural nets, but is computationally more efficient [3].

1 Introduction

Sentiment analysis has long been one of the main applications of NLP techniques. The classic way of performing sentiment analysis relied on a fixed-length presentation of a paragraph, mostly using either clustering or a bag-or-words model. In their classic paper, Bo and Lee [1] provide a comprehensive review of sentiment analysis techniques. With the introduction of deep learning techniques in NLP, sentiment analysis has benefited from the added accuracy of the deep learning models. The word2vec model, specifically has added a new dimension with its unsupervised learning nature, where it is possible to feed in a large corpus without labeling the sentiments that they convey.

This work is inspired by a Kaggle challenge. In this work, we examine the effect of applying word2vec-based models to predict sentiments of IMDB movie reviews. We begin by defining a baseline using a bag-of-words model, and then add use word2vec based models and compare their performance on the test set. The performance is measured via the AUC of the ROC curve. We then also experiment with an extension of word2vec, namely paragraph vectors, where the ordering of the words in each review is also taken into account.

The rest of this paper is organized as follows. In Section 2 we discuss the related work on the subject in the literature. In Section 3 we describe the data set we used, and in Section 4 we explain how we cleaned and organized the data set for our work. In Section 5 we explain the overall methodologies and algorithm we used, and in Section 6 we describe each of our models in detail. The evaluation of our models is presented in Section 7, which is followed by our conclusions and directions for future work in Section 8.

2 Related Work

The main idea and methodology comes from Word2Vec algorithm, which was published by Google in 2013, is a neural network implementation that learns distributed representations for words [**?**]. Recent work out of Stanford has also applied deep learning to sentiment analysis; their code is available in Java. However, their approach, which relies on sentence parsing, cannot be applied in a straightforward way to paragraphs of arbitrary length [4]. We also used random forests as a baseline. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The

generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large [6].

3 Data

The labeled data set consists of 50,000 IMDB movie reviews, specially selected for sentiment analysis. The sentiment of reviews is binary, meaning the IMDB rating < 5 results in a sentiment score of 0, and rating ≥ 7 have a sentiment score of 1. No individual movie has more than 30 reviews. The 25,000 review labeled training set does not include any of the same movies as the 25,000 review test set. In addition, there are another 50,000 IMDB reviews provided without any rating labels.

4 Data Preparation

Our dataset consists of a TSV file, where each line contains a review and possibly a label. Each review is taken directly from the IMDB website, so it includes html markup. We first use the BeautifulSoup package to strip all html markup. We then strip the stop words and use the Porter Stemming package in NLTK to stem. Each processed sentence is then split into an array of words and is passed into the classification models.

5 Methodology/Algorithm

Deep learning has been state-of-the-art machine learning techniques, inspired by the architecture of the human brain and made possible by recent advances in computing power, have been making waves via breakthrough results in image recognition, speech processing, and natural language tasks. We train a model using bag of words, Word2Vec, and Paragraph Vectors and use the resulting word vectors for sentiment analysis. We use random forest regression trees as a baseline more over deep learning models to compare. We also use softmax as our classifier (in place of random forest).
The evaluation metric will be the area under the ROC curve. In statistics, a receiver operating characteristic (ROC), or ROC curve, is a graphical plot that illustrates the performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate against the false positive rate at various threshold settings.

6 Models

6.1 Features from a Bag of Words

We convert the sentences to some kind of numeric representation for machine learning. One common approach is called a Bag of Words. The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. In the IMDB data, we have a very large number of reviews, which will give us a large vocabulary. To limit the size of the feature vectors, we should choose some maximum vocabulary size. Below, we use the 5000 most frequent words (remembering that stop words have already been removed). We will have 25,000 rows and 5,000 features (one for each vocabulary word).

6.2 Random Forest and Softmax

At this point, we have numeric training features from the Bag of Words and the original sentiment labels for each feature vector. Random Forest uses many tree-based classifiers to make predictions, hence the ”forest”. In our project, we set the number of trees to 100 as a reasonable default value. More trees may (or may not) perform better, but will certainly take longer to run. Likewise, the more features you include for each review, the longer this will take. We also used softmax as our classifier. You can find the result of this model in the results section.

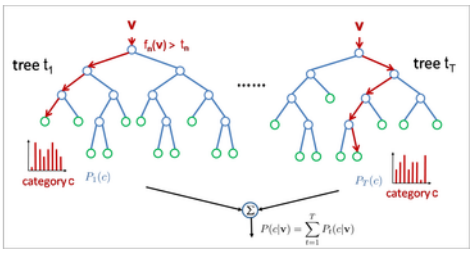


Figure 1: A random forest estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy.

6.3 Features from a Word2vec

We train the word2vec model, in the same manner that was described in class. That is, by counting the co-occurance of words in the same sentence, we create a vector representation of each word. We then use this matrix to train our prediction models. The training phase has several parameters that can affect the performance on the test set.

- Architecture: Architecture options are skip-gram (default) or continuous bag of words. We found that skip-gram was very slightly slower but produced better results.

- Training algorithm: Hierarchical softmax (default) or negative sampling. For us, the default worked well. Downsampling of frequent words: The Google documentation recommends values between .00001 and .001. For us, values closer 0.001 seemed to improve the accuracy of the final model.

- Word vector dimensionality: More features result in longer runtimes, and often, but not always, result in better models. Reasonable values can be in the tens to hundreds; we used 300.

- Context / window size: How many words of context should the training algorithm take into account? 10 seems to work well for hierarchical softmax (more is better, up to a point).

- Minimum word count: This helps limit the size of the vocabulary to meaningful words. Any word that does not occur at least this many times across all documents is ignored. Reasonable values could be between 10 and 100. In this case, since each movie occurs 30 times, we set the minimum word count to 40, to avoid attaching too much importance to individual movie titles. This resulted in an overall vocabulary size of around 15,000 words. Higher values also help limit run time.

Note that an advantage of word2vec is that we can train the model without having labeled data, which enhances our training set to 50,000 reviews. To make predictions on the test data, we experiment multiple ways of coming up with a vector representation of a review.

- A naive way of doing so would be to represent each review as the average of the word vectors representing it.
- We note that in a given paragraph, the thesis and the conclusion are most indicative of the tone of the paragraph. The thesis is usually in the beginning and the conclusion is towards the end. Thus, to take advantage of that, instead of using averages, we use weighted averages, where words at the beginning and end of a review receive more weight than those in the middle.
- Another way to so was by first clustering the words from the word2vec model using K-means and then assigning each cluster an index. Then, we model each sentence by using

a bag of words model (as we did earlier), but this time the index of each word is the index of the cluster that it represents. The training and predicting on the test data is done in the same manner as we do in Section 6.1.

6.4 Features from a Paragraph Vectors

Here, we implement the Paragraph Vectors model as described in [2]. This model is similar to the word2vec model, but with the difference that each sentence (or paragraph) also has a vector representation (in addition to the words in the sentence). When it comes to training and prediction, it treats each sentence as one vector by concatenating the paragraph vector with the word vectors. This came nicely in a gensim package called Doc2Vec. The parameters of doc2vec mirrored those from word2vec and we used random forest and softmax for classification purposes. We used random forest for classification of the results.

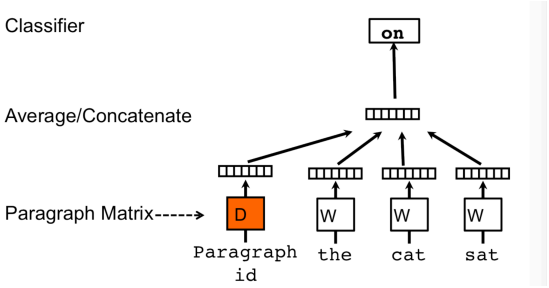


Figure 2: A framework for learning paragraph vector. This framework is similar to the word to vec framework; the only change is the additional paragraph token that is mapped to a vector via matrix D.

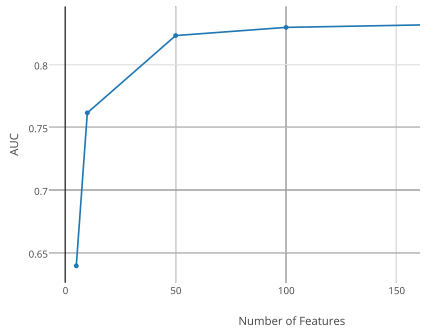
7 Results

The evaluation part of our work makes use of the Kaggle evaluator. The evaluation consists of 25,000 unlabeled reviews that we classify with each of our models. We submit our results to Kaggle and we receive an AUC value for the corresponding ROC curve. Our evaluation is based on the AUC value, and hence comparison with the reported accuracies in the literature may not be vis-a-vis.

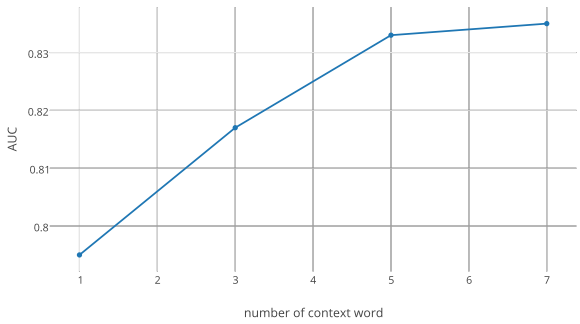
For our baseline, we used the bag of words model from Section 6.1. We limited our vocabulary to 5000 and used 100 trees in random forest, and obtained a baseline AUC of 84.44. Our evaluation consists of two types of analysis. In the first part, we explore the effect of the parameters of word2vec in the vanilla case (averaging the word vectors). The second part of the evaluation consists of evaluating different models with the same set of parameters used across them.

The results of the first part of our evaluation are shown in Figure 3. We note that in the range we explored, the AUC increases, as the parameters get larger. There is an exception in the case, and that is for the min word parameter. The min word parameter sets the minimum frequency of a word, for it to be considered in word2vec. This threshold helps increase the AUC until a certain limit, but after that the AUC drops sharply. We attribute this sharp fall to removal of a large number of words from our model, which reduces predictability of the model due to the small dictionary. Another surprising outcome was the effect of the number of trees on the AUC. We find that the training error is almost zero for all sizes of tree, meaning that the model is close to overfitting, but as we increase the tree size, the AUC increases as well.

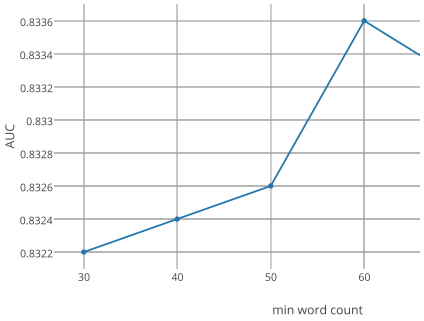
?



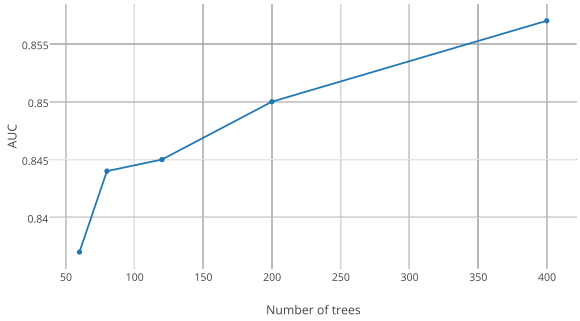
(a) Area under curve (AUC) vs number of features



(b) AUC vs number of context words



(c) Remove the words that are less frequent than min word count can effect the accuracy as the above figure shows AUC vs min word count



(d) AUC vs number of trees in random forest regression model in word2vec features

Figure 3: Comparison of AUC vs different parameters of word2vec and random forest

The results for the second part of our evaluation are shown in Table 1. The introduction of word2vec causes a significant increase in the AUC, compared to our baseline. The state of the art method in this area is the Paragraph Vector model and we see that it outperforms all other methods we evaluated. We note that classification via softmax has the best AUC among the word vector averaging models. We attribute this to the fact that random forests reach their limit after we use a large number of trees adn they begin to overfit.

With respect to clustering and averaging word vectors, we find that clustering performs a bit better. We attribute this to the information gain about the sentiments from the clusters. Our weighted averaging of the word vectors seem to validate our thesis about the placement of sentiments in the paragraph. While the word2vec model is inferior to the paragraph vector model, the weights of the word vectors make up for some of the information loss during averaging.

8 Conclusion and Future Work

We observed that the word2vec models significantly outperform the vanilla bag-of-words model. We observe that the variation of the parameters of word2vec have sometimes surprising outcomes. While the AUC increases with the number of trees, it falls sharply as we increase the minimum frequency of words. Amongst the word2vec based models, softmax provides the best form of clas-

Table 1: The performance of Paragraph Vector and Word2Vec with weighted averaging compared to other approaches on the IMDB dataset.

Model	**AUC**
BoW (bnc)	0.832
Word Vector Averaging + Softmax	0.886
Word Vector Averaging + Random Forrest	0.861
Word Vector Clustering	0.873
Paragraph Vector	0.912
Word Vector Weighted Averaging + Random Forrest (Ours)	**0.909**

sification, while weighted averaging of the words based on their position, gives a significant boost to the accuracy and comes very close to the state of the art.

We have a few recommendation for future work. We disregard non-alphabet characters in our analysis, but it may be useful to also take those into accounts. For instance, emojis and a repetition of punctuation (e.g., multiple exclamation marks) may infer the sentiment of the review significantly. Other classification methods may also be examined, such as regression and k-nearest neighbor. For the learning part, we recommend examining recurrent neural networks and RNTN model (if parse trees are available).

References

[1] Bo Pang and Lillian Lee. 2008. Opinion mining and sentimentanalysis. Found. Trends Inf. Retr., 2(1-2):1-135.

[2] Kaggle competition

[3] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[4] Stanford sentiment analysis

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. "Learning Word Vectors for Sentiment Analysis." The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[6] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32

[7] 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.