# Comparison of sentiment prediction techniques on a common movie review data set.

Matthew Smith
mrs9107@g.rit.edu
Rochester Institute of Technology dept. of Computer Engineering

***Abstract*—In language, sentiment is a major component of overall expression. Sentiment analysis and classification uses traditional machine learning techniques but learns to predict what the text is trying to express. In this paper we will survey, discuss, and compare various techniques for sentiment analysis. Such techniques include older methods such as the Bag of Words model and more modern techniques such as neural net based word and paragraph vector analysis. These models would be used on a common random forest classification tree. These methods will be implemented and tested on the IMDB movie review dataset. This dataset expresses the star review system as a binary value, where greater than seven is a '1', and less than five is a '0'. Intermediate values were thrown out. In testing, the paragraph vector based Doc2vec outperformed in sentiment accuracy prediction.**

## I. Introduction

We're currently living in a world saturated with textual reviews, discussions, and opinions. Sentiment is a crucial datapoint for being able to take these pieces of text and extract a useful understanding of the author's intention. While much of machine learning tends to lean towards predicting what the data represents, sentiment analysis gives an opportunity to understand emotion and opinion behind the data. Sentiment analysis is distinct from subject analysis in the inference phase of training. For both types of analysis, vocabularies are built for the model to have an understanding of surrounding words. These methods split during training for classification, as subject analysis would train using subject ground truth labels and sentiment analysis would train using sentiment labels accordingly. Sentiment analysis is primarily performed on text input taken from sources such as reviews, forums, and comment sections. For this paper, we will be using the IMDB dataset, "Learning Word Vectors for Sentiment Analysis", introduced by Maas et al. [1]. This dataset consists of 100k individual reviews, with 50k labeled with either a positive or negative sentiment. This will allow for a large vocabulary to be available for training producing a diverse model to be used on unknown reviews. This dataset will be beneficial due to the medium to large sized paragraphs available, which will be allow for more robust models. In this paper, we will be implementing several models beneficial in sentiment analysis, including Bag of Words, Word vectors, and paragraph vectors. These models will be compared using identical classification trees in a random forest configuration. In the end, it was shown that Bag of Words is still a very powerful method for sentiment classification but with minor optimization, Word2Vec and more so Doc2Vec, was able to continue to improve accuracy past Bag of Words abilities.

## II. Discussion on the dataset

In this section we will be discussing the IMDB dataset and its format. Discussion on the data split, labeled and unlabeled, and data cleaning will be included.

Also discussed will be the processing of the data, such as removing any html tags, removing any unwanted punctuation, and splitting the data on white space to create an array from which to learn on. Additionally, discussion on what punctuation should be kept and what should be ignored.

The Large Movie Review Dataset introduced by mass et al. [1],was built with sentiment analysis in mind. This dataset consists of 100k movie reviews from the Internet Movie DataBase (IMDB). Half of the reviews in the dataset were labeled using Amazon's Mechanical Turk. For each sample, if a review gave greater than or equal to seven stars, the review was given a positive label. Likewise, a review with less than or equal to four, it received a negative label. This binary classification labels allow for the task of classification to be easier later on, as it avoids vague moderate reviews and their effect on the learning.

In raw form, the data was separated into 5 directories, each containing many text files of individual reviews. Test and train data were separated into separate directories, and within those, positive and negative reviews were separated as well. In order to make use of these reviews, each directory in the lowest level was concatenated into a single TSV file, with the file name as the ID, the sentiment, and the full review, each separated by tab characters.

Next, the data must be further processed and cleaned. For each review, punctuation, capital letters, and HTML tags were stripped and the clean reviews were written to a new TSV file with the same format. This process was done a second time, but this time removing stop-words from the reviews. Stop words are words which connect together sentences and often do not provide much meaning. These are removed to avoid noise in the models.

Finally, it was important to consider non text expression such as extra "!" or "?", or emoticons included in the review.

Go et al. [2] explained how the effect of including emoticons moves focus away from other features, primarily the n-gram features being analyzed. N-gram analysis will be discussed later in the paper but they will be the most important features for training.

## III. Survey and comparison of methods

Here we will be an overview of methods to solve the problem of sentiment analysis as well as a comparison of their strengths and weakness. Text analysis is not a new field, in fact it has been a point of focus in machine learning for several decades. Word and text analysis can take many forms, from analysis on small parts of words via N length grams, whole words, or even entire sentences and paragraphs.

Methods using N length grams or N-grams, are particularly common. The N-gram works by taking each word sans punctuation and tokenizer each word. It then counts the frequency of each token. [3] One special case of the N-gram is the Bag of Words model, where N = 1. This model introduced by Harris [4] with the intention of constructing a model in which language can be structured with respect to independent features. Harris' writings were inherently based in the understanding of linguistics over pure machine learning, but this basis and approach has proven effective, as the bag of words model has persisted over the years.

Working off the Bag of Words model, Mikolov et al. [5] from Google introduced the Word vector (or Word2Vec) model. Here, the goal was to learn high-quality word vectors from extremely large scale data sets to extract almost as large vocabularies. Word2Vec excelled in its performance as it was able to operate on vocabularies of several million words where previous models only trained on several hundred. Word2Vec has the added benefit of being able to train each vector with either a continuous Bag of Words, or a continuous skip gram. These differ in how neighboring word vectors are weighted. Continuous Bag of Words averages the weights of the neighboring vectors while continuous skip grams give more weight to word vectors that are closer to the target word.

Next to be considered is the more mature version of Word2Vec, the paragraph vector model. Introduced by Mikolov and Lee [6], the paragraph vector model (or Doc2Vec) was introduced as an extension to Word2Vec. Similarly to Word2Vec, Doc2Vec has a choice of two methods to train on, either a distributed memory model, or a distributed Bag of Words. For the distributed memory, paragraph vectors are built along side the word vectors and included as an additional word to the vocabulary. This can be seen in Fig. 2. The distributed Bag of Words loses order of the words and creates the paragraph vector based on the Bag of Words understanding of each paragraph. This can also be seen in Fig. 3.

Both Word2Vec and Doc2Vec train via a two layer neural net, trained using either hierarchical softmax or negative sampling. These can be chosen based on a focus on infrequent words or frequent words accordingly.

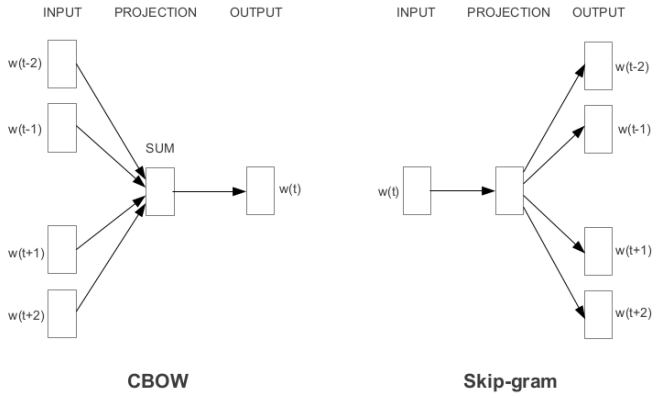


Fig. 1: Comparison of CBOW and Skipgram methods Mikolov et al. [5] introduced in the original paper. It can be seen how CBOW will take surrounding words and predict the word that fits, while Skip-gram will guess surrounding words from a given word.

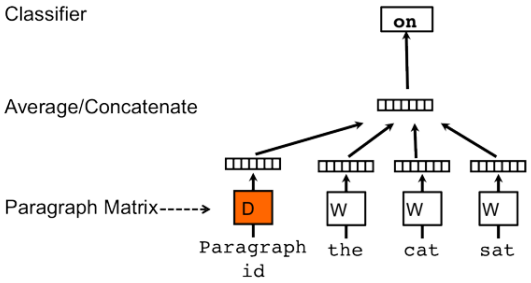


Fig. 2: Distributed memory model from Doc2Vec from Mikolov and Lee [6]

## IV. Method of Test

In this section, we will be discussing the methods that will be chosen for comparison. Also to be discussed is why each was chosen.

When testing to compare these methods, it was important to ensure data was consistent. Certain implementations had taken influence from online tutorials such as [7] but had datasets pre-processed from the source in varying ways. Each implementation needed to be modified to accept the processed format defined for this experiment.

First, the Bag of Words (BoW) model was implemented. In order to construct a BoW model, a vectorizer object was defined using sklearn's CountVectorizer library. The reviews of the corpus were concatenated into a list and fed into the CountVectorizer fit_transform function to create the vector model that will be used for testing.

Next, the word vector model better known as Word2Vec was implemented. Here, the training corpus was split into an array of sentences. This was then fed into the gensim library's implementation of Word2Vec. For initial baseline tests, the parameters were given as follows; 300 features, minimum of 40 words in a sample, 4 workers, a context

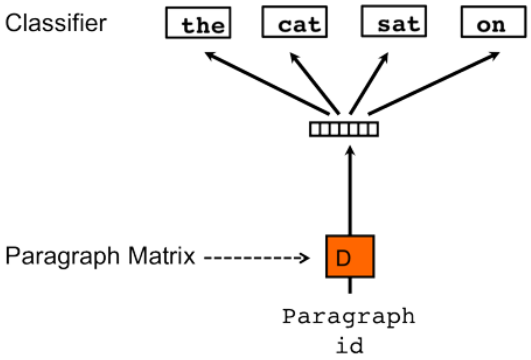


Fig. 3: Distributed bag of words model from Mikolov and Lee [6]

window of 10 words, and a frequent word down-sampling of $1x10^{-3}$. The model is unsupervised, and therefore could be trained on both labeled and unlabeled reviews. The model was generated and saved for later use.

Finally, the paragraph vector model, Doc2Vec was implemented. Here, more work needed to be done as the gensim implementation requires very specific input. A named tuple list, which behaves in a similar manner to a dictionary, was created to hold lists of words from each review, as well as a tag for the Doc2Vec model to keep track of. This used the full training corpus, labeled and unlabeled in a similar manner to the Word2Vec implementation. Once again this model was saved for later use.

In order to compare the models and not classification methods, the same random forest tree classifier was used. This implementation was taken from a tutorial for BoW and Word2Vec by Kaggle [7]. The random forest classifier will fit a classification tree to the trained model and the corresponding training labels. The test data would then be vectorized and applied to the tree, where sentiment could be predicted. For all tests except one, the random forest used 100 estimator trees. Using this method, all three models could be used to classify in such a way that allows for comparison.

## V. Results

For the first set of trials, each chosen model was put through the random forest classifier with 100 trees. For the Word2Vec and Doc2Vec models, base level settings described above were used. For this unoptimized comparison, Bag of Words slightly out performed Word2Vec but was out performed by Doc2Vec. Regardless, the far more simple Bag of Words performs above expectations when put alongside two far more modern models.

Next, in an attempt to change the same variable for all three tests, the number of trees used in the classifier was increased from 100 to 1000. The same models were fed into this modified random forest. This time, classification took far longer but produced interesting results. With the same model and only improved classification, Bag of Words was found to have the greatest accuracy by .07%. Doc2Vec was very close behind but once again, Word2Vec had the worst performance.

!htb

TABLE I: Experiment results

| Settings | Bag of Words accuracy (%) | Word2Vec accuracy (%) | Doc2Vec accuracy (%) |
|---|---|---|---|
| Unoptimized (100 trees) | 82.84 | 82.64 | 83.06 |
| Optimized (1000 trees) | 83.70 | 83.26 | 83.63 |
| Optimized (context window = 20 words) | n/a | 85.38 | 86.1 |

Finally, an attempt to optimize our vector methods was made, while returning the tree count back to 100. For both Word2Vec and Doc2Vec, the context windows that control how many words around a given word are considered, were increased. For both, the context window was increased from ten words to twenty words. As this change did not effect Bag of Words, it was left out of the results. With this change, both Word2Vec and Doc2Vec showed considerable improvements. Word2vec found a 2.7% increase in accuracy and Doc2Vec found a 3.04% increase over the unoptimized scores. With this simple change, the word vector models were able to improve a considerable amount over general Bag of Words performance and their own unoptimized models. The results can be seen in Table I

In addition, our Word2Vec model was used with Tensorflow's Tensorboard utility to create 3D vector space representations of our model. Several words were searched in the vocabulary, including "emotional", "stoic", "business", and "video". With this visualization tool, one could search for these terms and have the 100 closest words highlighted. Each word was searched and distances between these clusters were observed. Images were taken of each search. It can be seen that the area shown when “emotional” is highlighted, seen in Fig. 4, shows a very different area from when “business”, Fig. 5, is searched. When “stoic” was searched, the area highlighted was in between “emotional” and “business”, but closer to “emotional” as the model decides two states of emotion are more related. This can be seen in Fig. 6. The axis was then rotated 90 degrees around the Y axis and the term “video” was searched. By comparing the color axis, it can be seen that “video” highlights another distinct group all together. Another interesting aspect of these results found by noting the words deemed similar to the searched term. For video, words like “youtube” and “download” are closer to “Video” than “games”, indicating that the dataset for movies will mention methods of watching videos more than mentioning video games. This closeup can be seen in Fig. 7.

## VI. Conclusion

Sentiment analysis is a crucial aspect in any kind of textual analysis. The internet is saturated with opinion, and

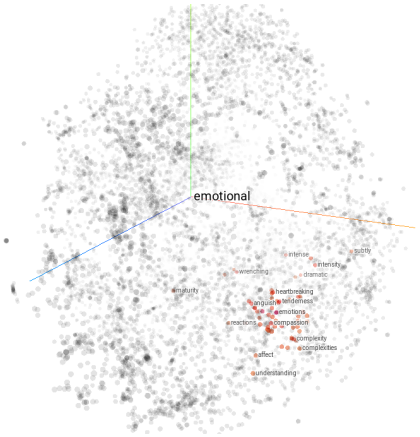


Fig. 4: Word vector representation in 3D space showing words similar to "emotional"

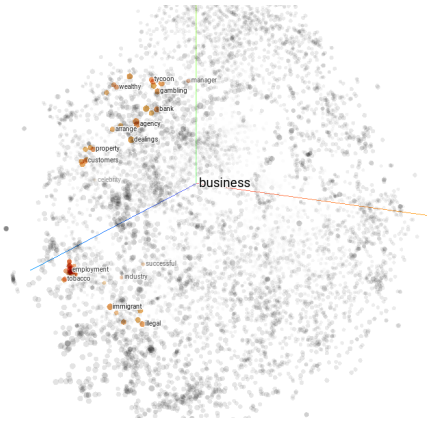


Fig. 5: Word vector representation in 3D space showing words similar to "business"


Fig. 6: Word vector representation in 3D space showing words similar to "stoic"

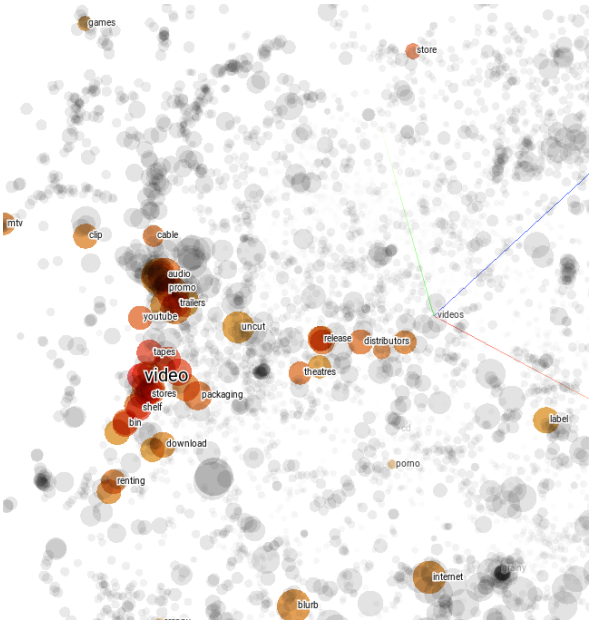


Fig. 7: Word vector representation in 3D space showing a close-up of words similar to "video"

being able to construct a model to extract that opinion is a priceless function. In this paper, we have covered what sentiment analysis is as well as the distinction between sentiment and subject prediction. We also studied methods for representing and understanding the vocabulary. By building models such as the Bag of Words or Doc2Vec and passing them through identical random forest classifiers, we were able to compare these word representation models to gain an understanding of their abilities. In the end, Doc2Vec showed the strongest results but all three methods had considerably close performance.

## References


[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[2] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," vol. 150, 01 2009.

[3] J. M. T. William B. Cavnar, "N-gram-based text categorization." P.O. Box 134001 Ann Arbor MI 48113-4001: Environmental Research Institute of Michigan.

[4] Z. S. Harris, "Distributed structure," vol. 10:2-3, pp. 146–162, 1954, word.

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[6] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[7] Bag of words meets bags of popcorn. [Online]. Available: https://www.kaggle.com/c/word2vec-nlp-tutorial