NLP Assignment 2

SVM-based Sentiment Detection of Reviews

Anik Roy, Christ's (ar899)

December 6, 2019

Word Count: 995[1]

1 Introduction

Support vector machines (SVMs) are an ML model which can be used to classify vectors. This method can be applied to the task of classifying documents by representing each document as a vector. In this report, I use a neural model, doc2vec, introduced by Mikolov and Le [4], in order to generate feature vectors. I also qualitatively show that the vector space produced by the doc2vec model is meaningful, by examining the behaviour of document embedding generation by doc2vec.

2 Background

In a previous report, I used a bag of n-grams representation for documents (BOW), and used an SVM to classify these feature vectors. In BOW, each document is represented by a feature-count vector $(n_1(d), \ldots, n_m(d))$, with $n_i(d)$ being the number of occurrences of f_i in d. I also trained on representation only using the presence of n-grams, setting $n_i(d)$ to 1 if f_i appeared in d, and 0 otherwise.

2.1 SVM

SVMs are supervised learning models which are used to classify feature vectors in an n-dimensional space. Training consists of finding a hyperplane which separates the two classes with the largest margin. Classification takes place by measuring the distance of feature vectors to the plane.

2.2 Doc2Vec

Doc2Vec is an unsupervised model for learning document embeddings which can be used as feature representations. It tries to overcome two flaws in BOW - word order not being taken into account, and the semantics of words being ignored. Doc2Vec produces fixed-length vectors for documents of arbitrary length. These vectors can't be directly interpreted. [4]

Doc2Vec extends word2vec [8] , a method of learning word embeddings. There are two doc2vec architectures, distributed bag of words (DBOW) and distributed memory (dm).

3 Method

I implemented an SVM classifier which used doc2vec representations[2]. I used the doc2vec implementation found in gensim [10] and the SVMlight [3] SVM implementation. I used the Stanford Large Movie Review Dataset [6] in order to train the doc2vec model. This contains 50,000 labelled (positive and negative, balanced) reviews and 50,000 unlabelled. I performed tokenization on this corpus using the nltk package [4] [1]. I also used a smaller labelled and balanced dataset of 2000 tokenised reviews, a modificaion of the reviews used by Pang et al. [9], given to us in the framework of an NLP course.

I used the smaller dataset to train the SVM classifier, performing a 10-fold round-robin stratified cross-validation.

To tune parameters for the doc2vec model, I used a 10% validation set to evaluate different doc2vec hyperparameters. I employed a nave search strategy, starting from the parameters used by Lau and Baldwin [3]. After finding the hyperparameters which gave the highest accuracy on the validation set, I discarded all other models.

I compare the doc2vec based SVM model to two baseline svm models using a BOW representation.

4 Results

The best parameters found for the doc2vec model are shown in Table 1, acheiving an accuracy of 86% on the validation set. I also set the parameter

[1] Using texcount

[2] Available at both `https://github.com/anik545/Part2-NLP` and at /home/ar899/Part2-NLP on MCS machines

[3] http://svmlight.joachims.org/

[4] https://www.nltk.org/api/nltk.tokenize.html

Method	Vector Size	Window Size	Min Count	Epochs	Hierachical Softmax
DBOW	120	7	15	5	True

Table 1: Values for the optimal hyperparameters found for the doc2vec model

		(2)		
		BOW - freq.	BOW - pres.	Doc2vec
(1)	Doc2vec	1.40×10^{-3}	2.18×10^{-2}	-
	BOW - pres.	4.00×10^{-4}	-	
	BOW - freq.	-		

Table 2: Comparing SVM systems: p-values for system (1) outperforming system (2), using a monte-carlo permuation test with $\alpha = 0.05$, $R = 5000$

dbow_words in order to learn individual word vectors alongside document vectors. I found that the DM architecture was not as effective as the DBOW architecture.

The results of then evaluating the systems under cross-validation (not using the validation set), show that the doc2vec representation is significantly better than either of the bag of words representations when using an SVM.

	Representation	Accuracy
A	BOW - frequency	73.3
B	BOW - presence	87.1
C	Doc2vec	88.2

Table 3: Accuracies of SVM, using ten-fold cross validation over 1800 reviews

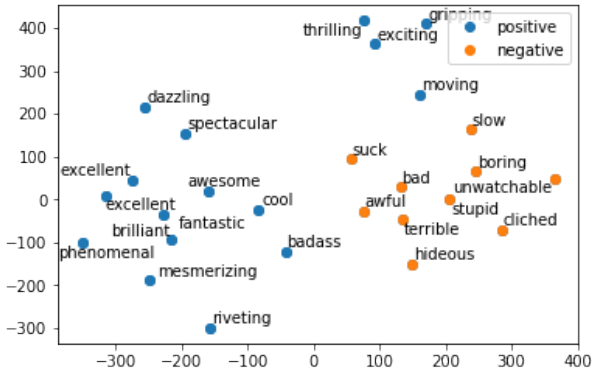

Figure 1: T-SNE projection of sentiment indicators

5 Analysis

Since doc2vec vectors are not directly interpretable, we have to analyse document embeddings to attempt to see what the model is doing.

Since word embeddings were also learned by the DBOW model, I look at word vectors generated for a variety of positive and negative adjectives. Figure 1 shows a 2-d t-SNE[5] projection [7] of embeddings for sentiment words suggested by Pang et al. [9]. The negative words are grouped, suggesting that the model learns the sentiment of key individual words.

I also employ a technique for visualising word vectors as heatmaps (Li et al. 2015) [5]. Each column shows the difference from the mean of each component in each word embedding. The influence of a word embedding is determined by its deviation from the mean vector, since these vectors will contribute more to learned document embeddings. From this, we can hypothesise that more importance is placed on key sentiment indicators - e.g. 'hate'. In addition, words connecting clauses and negation words are salient, since they can change the meanings of parts of the document. An example of this is shown in Figure 2, in the phrase *"I hate the movie, although the plot is interesting"*, where we can see that the words hate, although, plot and interesting are the most salient. Here, a negative clause is combined with a positive clause, with the overall class being negative - 'hate' overshadows 'interesting' on the plot. This can also be seen on the heatmap, with 'hate' being the more salient adjective.

We can also visualise entire document vectors as heatmaps, allowing us to see the effect of intensification and negation. Figures 3 and 4 show the document vectors for several short phrases and their modifications. Some patterns can be seen (qualitatively) here, with certain dimensions changing on negation, e.g. the ones highlighted on the figure. With negation, we also observe the difference between entire embeddings for different classifications,

[5]The t-SNE implementation at https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html was used

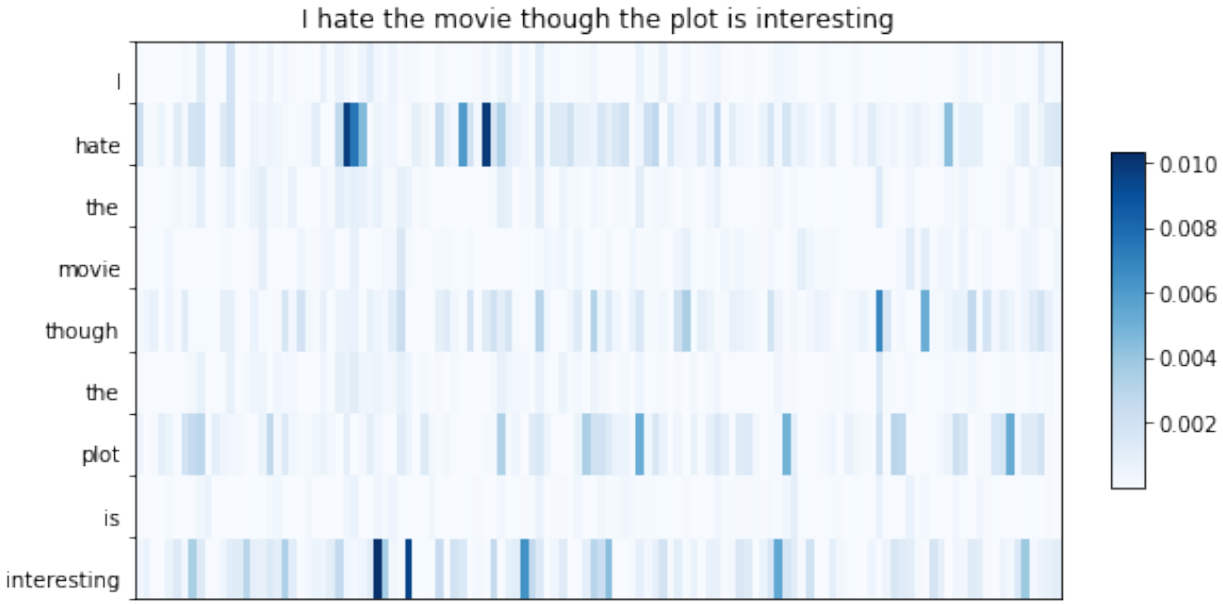

Figure 2: Plot of deviation from the mean of word vector components

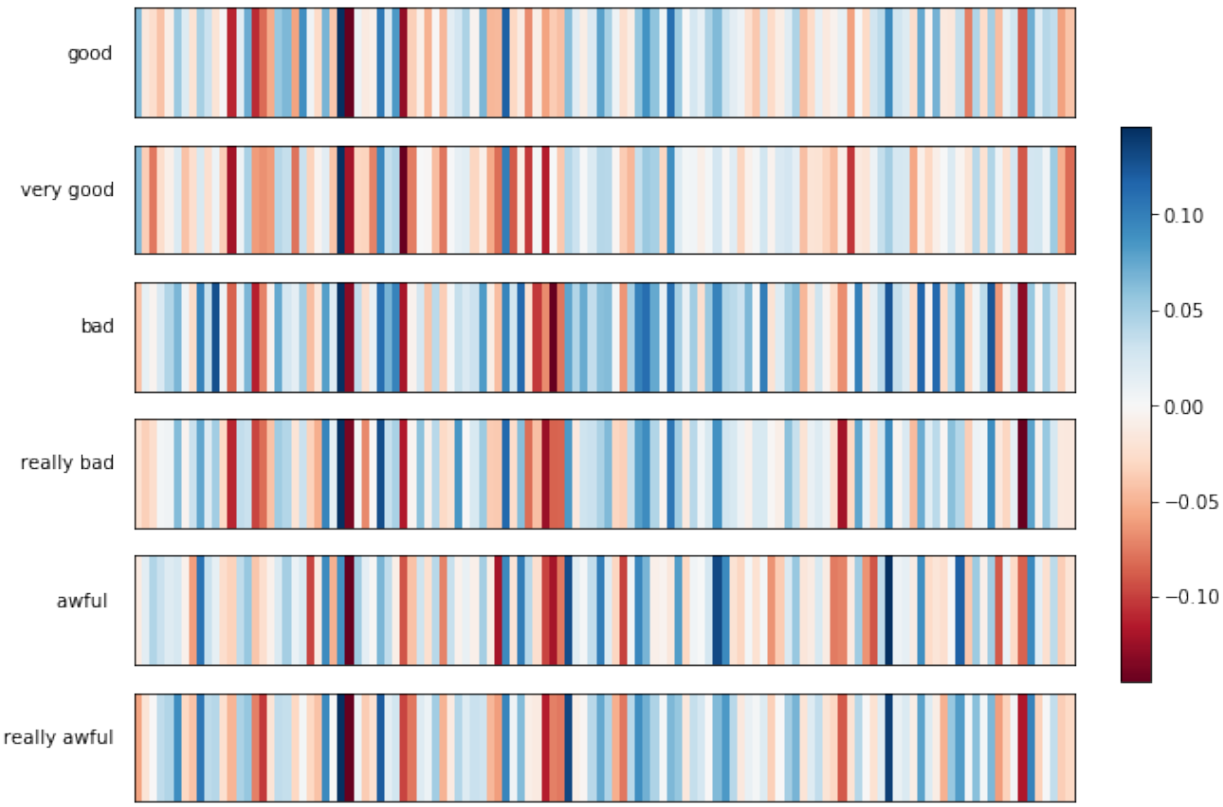

Figure 3: Plot of document embeddings for phrases and their intensifications

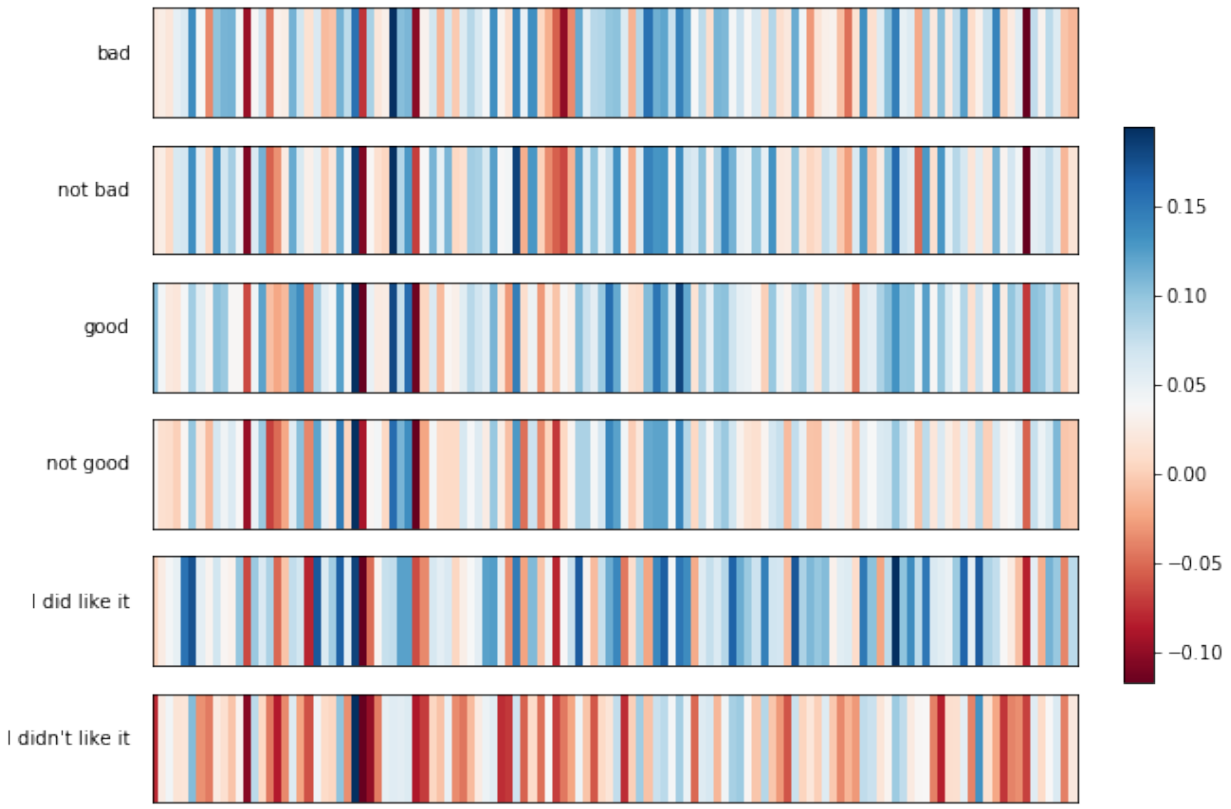

Figure 4: Plot of document embeddings for phrases and their negations

since a negated positive document is classified negative. For example, the difference between "I did like it" and "I didn't like it' is clear. The pattern for intensification is more difficult to see, since we expect these modifiers to keep the classification the same, but with higher confidence.

Figure 5: t-SNE projection of document embeddings of phrases and their negations

The T-SNE projection in figure 5 also shows how negation affects documents embeddings, with negations representing a similar shift in the vectors. I have also plotted some random common words to show the clustering of these similar words. This is analogous to the word2vec case, where analogous pairs will have similar vector differences [2].

6 Conclusion

I have shown that doc2vec produces a representation, that when paired with an SVM model, can outperform a bag-of-words representation. Despite not being directly interpretable, doc2vec still creates a meaningful semantic space for documents (analogously to word2vec), and that we can visualise vectors in this space to show this.

References

[1] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[2] Dawn Chen, Joshua C. Peterson, and Thomas L. Griffiths. Evaluating vector-space models of analogy. *CoRR*, abs/1705.04416, 2017.

[3] Jey Han Lau and Timothy Baldwin. An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*, 2016.

[4] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[5] Jiwei Li, Xinlei Chen, Eduard H. Hovy, and Dan Jurafsky. Visualizing and understanding neural models in NLP. *CoRR*, abs/1506.01066, 2015.

[6] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[7] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[9] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques.

[10] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA.