NLP Assignment 2 Report

Doc2Vec for Sentiment Detection of Reviews

Zébulon Goriely, Queens', zg258

Thursday 21st November, 2019

Word Count: 987[1]

1 Introduction

In a 2014 paper, based on `word2vec`, Le and Mikolov [2] introduced `doc2vec` for learning embeddings for sequences of words. Here, we investigate the use of `doc2vec` document vectors in the task of sentiment detection, using a Support Vector Machine (SVM) classifier.

We compare the success of the classifier trained with `doc2vec` vectors to the same classifier trained[2] with simple word-frequency and one-hot coded vectors. Finally, we examine means of qualitatively analysing the behaviour of `doc2vec`, in particular how sentiment-coding adjectives are represented.

2 Background

To classify reviews, we use an SVM classifier trained and tested on document vectors.

Previously, we looked at document vectors where each document d was represented as a vector $\vec{d} := (n_1(d), n_2(d), \ldots, n_m(d))$ where $n_i(d)$ is the occurrences f_i of document d.

Selecting unigrams as our feature, we will compare the performance of training and testing an SVM classifier on:

- frequency-based bag-of-words document vectors
- one-hot encoded bag-of-words document vectors
- vectors inferred from a trained `doc2vec` model

2.1 Doc2Vec

Mikolov et al. introduced the ideas of skip-grams and `word2vec` to create compact vector-space representations of words [5]. Unlike our frequency-based document vectors and one-hot encoded vectors, the dimensions of the vectors are not directly interpretable.

Le and Mikolov extended this idea to `doc2vec` which learns embeddings of *sequences* of words. A key feature is that it is agnostic to granularity, generating fixed-length vectors from variable-length pieces of text. In this case, we train `doc2vec` to output document embeddings, a new type of vector that Le and Mikolov claim is effective for sentiment analysis.

The two architectures in `doc2vec` are the distributed memory and distributed bag of words model which are closely correlated to the `word2vec` and skip-gram models respectively.

2.2 Support Vector Machine

Pang et al. [6] introduced the use of SVM classifiers for sentiment analysis.

Letting $c_j \in \{1, -1\}$ (corresponding to positive and negative) be the correct class of document d_j, the procedure finds the hyperplane represented by vector $\vec{w}$ which divides document vectors into two classes.

Once trained, classification proceeds by determining which side of the hyperplane each d_i falls on. We use Joachim's (1999) SVM^{light} package[3], using default parameters.

[1] *texcount docs/assignment2/report.tex*

[2] We were given a dataset of movie reviews in the framework of a course in NLP

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

Hyperparameter	Description	Best Value
Vector Size	Dimension of word vectors	124
Window Size	Left/right context window size	6
Min Count	Minimum frequency threshold for word types	20
Negative Sample	No. of negative word samples	0
Epoch	Number of training epochs	5

Table 1: A description of `doc2vec` hyperparameters and the best values found for this task.

	Unigram-frequency	One-hot unigrams	doc2vec
Unigram-frequency	1	2.00×10^{-4}	2.00×10^{-4}
One-hot unigrams	2.00×10^{-4}	1	1.60×10^{-3}
doc2vec	2.00×10^{-4}	1.60×10^{-3}	1

Table 2: The p-values from a Monte-Carlo permutation test with a significance level of $k = 0.01$.

3 Method

We implemented our SVM classifier using Python[4]. For the `doc2vec` implementation, we use the gensim library[5].

We train the `doc2vec` model on a large external corpora of 100,000 movie reviews provided by the Stanford Large Movie Review Dataset [4]. In a 2014 evaluation of `doc2vec` , Lau and Baldwin [1] suggest relevant parameters to use for training, as described in Table 1.

To tune our parameters we set aside a validation corpus comprising of 10% of our 2000 movie review dataset. For each set of parameters chosen, we trained `doc2vec` on the 100,000 unlabelled files, used this model to generate vectors to train SVM on 90% of our 2000 movie review dataset and tested on this validation corpus. The validation corpus was never used for training and once suitable parameters were found, it was not used for testing (testing was done through cross-validation on the remaining 1800 reviews).

Parameters were initially chosen using Lau and Baldwin's suggestions. We then tested a range of values for each parameter, in order to increase the accuracy when testing on the validation corpus.

During this process we also found that the distributed bag of words (`dbow`) model was more effective than the distributed memory (`dm`) model and that using hierarchical softmax also improved performance. The best parameters found from this process are shown in Table 1, giving an accuracy of 89% on the validation corpus.

[4]https://github.com/ZGoriely/cambridge-nlp

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

4 Results

	Vector Type	Accuracy
(1)	Unigram-frequency	75.9
(2)	One-hot unigrams	86.7
(3)	doc2vec	89.3

Table 3: Average ten-fold cross-validation accuracies, in percent. Dataset: 1800 movie reviews.

We ran ten-fold cross-validation for our three experiments as detailed in Table 3, using our 1800 movie reviews.

The result of 89.3% achieved using `doc2vec` is significantly better than the 75.9 and 86.7 achieved by the frequency vectors and one-hot vectors respectively. The significance was calculated using a permutation test, with p-values recorded in Table 2.

5 Discussion

Our results show that the more complicated `doc2vec` model, taking the *context* of words into account, performs better at sentiment analysis than the simple naive bag-of-word models for

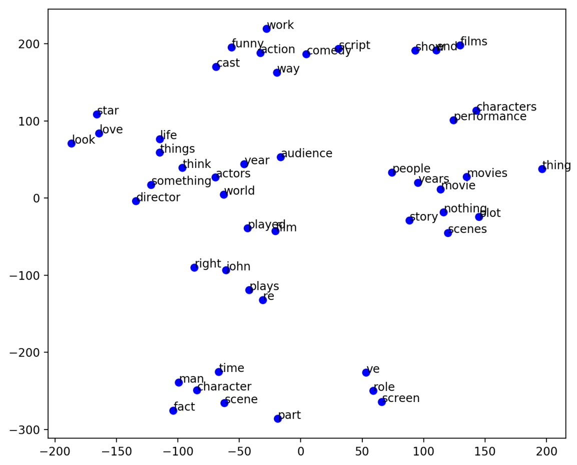
Figure 1: Two-dimensional t-SNE projection of doc2vec embeddings of the 50 most frequent nouns

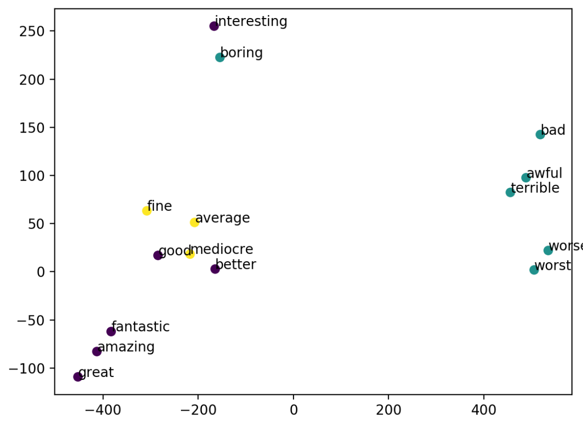
Figure 2: Two-dimensional t-SNE projection of doc2vec adjective embeddings

generating document vectors. Using T-sne and heatmap graphs, we can qualitatively examine what the model is doing.

By plotting the 50 most frequent nouns in our 2000 files using t-SNE in Figure 1, we see how `doc2vec` learns context. Nouns used in the same context such as *plot, story* and *man, character* are grouped together and are thus represented by similar vectors in the `doc2vec` embedded vector space.

We hypothesise that `doc2vec` learns sentiments of common adjectives from their usage in the positive and negative files that `doc2vec` is trained on. We plot these adjectives using t-SNE in Figure 2.

Word	Positive	Negative
amazing	2004	516
good	15025	14728
bad	3747	14726
average	595	831

Table 4: Occurrences of words in positive and negative training files

We observe that the model seems to learn the sentiment of these adjectives. *Fantastic, amazing, great* are grouped together and *bad, awful, terrible, worse, worst* are also grouped. Interestingly, the neutral adjectives *fine, average, mediocre* seem to be grouped with the positive adjectives *good, better*. This may be due to the comparative occurrence of these adjectives in positive and negative files. As seen in Table 4, the words *amazing* and *bad* dominate in positive and negative files respectively whereas the words *good* and *average* are distributed more evenly between positive and negative files.

Li et al. [3] proposed a method to visualise indicator words by plotting the variance of word vectors in a sentence against each other. In Figure 4, we plot a variance visualisation of the sentence *"this has to be one of the greatest movies of all time"*; each grid corresponds to $||e_{i,j} - \frac{1}{N_s}\sum_{i' \in N_s} e_{i',j}||^2$ where $e_{i,j}$ denotes the value for jth dimension of word i and N denotes the number of token within the sentences. This gives us the word *greatest* as the strongest indicator for this sentence.

A heat-map visualisation of three short reviews, shown in Figure 4, reveals another analysis that `doc2vec` learns sentiment; the neutral review seems to lie between the heat-maps of the positive and negative reviews.

Conclusion

We have shown how using `doc2vec` vectors in an SVM classifier outperforms naive frequency-based vectors and one-hot vectors. We further

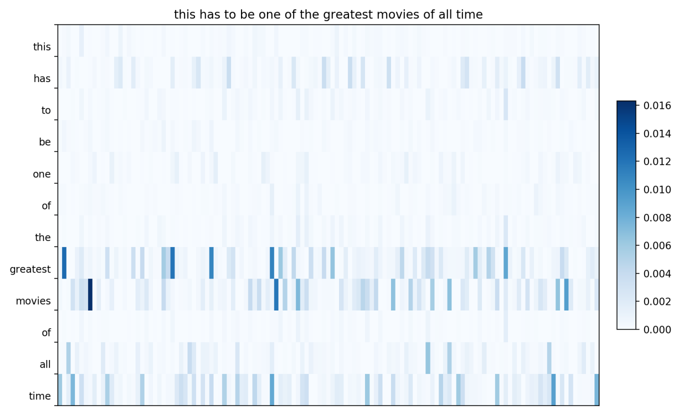

Figure 3: Variance visualisation of a single-sentence review

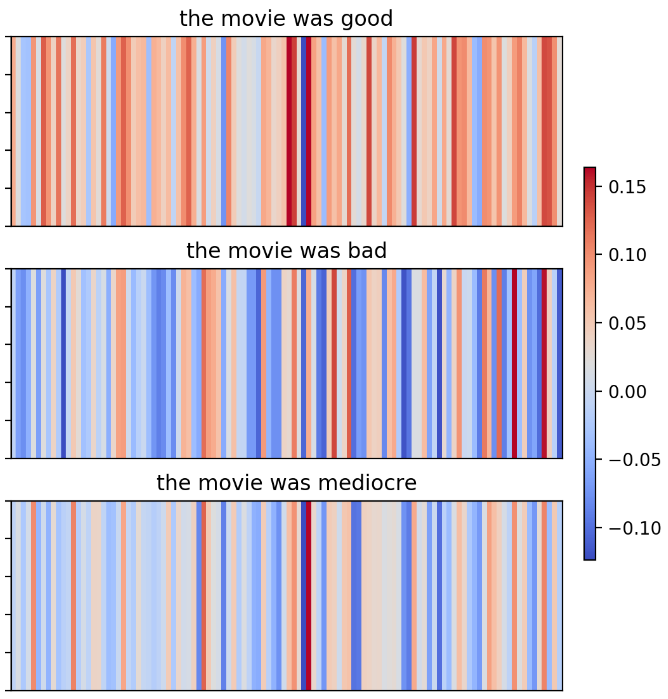

Figure 4: Heat-map visualisation of a good, bad and neutral sentiments

qualitatively analysed the behaviour of `doc2vec`, finding that the model learns sentiment through occurence of adjectives in positive and negative documents.

References

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

[2] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196, 2014.

[3] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066*, 2015.

[4] 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.

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[6] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.