NLP Assignment 2

Exploration of Doc2Vec for Sentiment Detection of Reviews

Andreea Bacanu, Christ's, aeb83

Friday 6^{th} December, 2019

Word Count: 999[1] – Code location: `https://github.com/andreea794/nlp-part2`

1 Introduction

This paper describes the training of a sentiment classifier using Le & Mikolov (2014)'s `doc2vec` embeddings for sequences of words as feature vectors for a Support Vector Machine (SVM). It further compares the resulting model to 2 previous baseline systems employing SVM with 2 distinct bag-of-words (BOW) approaches.

Finally, this paper aims to evaluate the performance of the previously described systems when tested on several disparate categories of reviews. The assumption is that when encountering words belonging to new domains, the `doc2vec` model will outperform the rest.

2 Background

Previous work includes my replication of Pang et al. (2002)'s experiment on sentiment classification, which compares two baseline systems: Naïve Bayes and SVM. The experiment is carried out with a set of preprocessing conditions (unigrams, bigrams, stemming, frequency cutoffs), as well as different approaches to feature counting (frequency, presence).

Feature frequency and presence are both word counting methods belonging to the bag-of-words (BOW) framework. In this framework, each document d is represented as a vector:

$$\vec{d} := (n_1(d), n_2(d), \ldots, n_m(d))$$

where $n_i(d)$ is either the number of occurrences of feature f_i in document d (for frequency-based features), or a binary value stating whether feature f_i appears in document d (for presence-based features).

2.1 Support Vector Machine

SVMs are *large-margin* classifiers that find a hyperplane represented by a vector $\vec{w}$ that separates the document vectors in one class from the ones in the other and for which the margin is as large as possible. The trained model then determines which side of the plane each document vector $\vec{d_i}$ in the test set falls on. I use SVM^{light} – Joachims (1999), with all parameters set to their default values.

2.2 Doc2Vec

The work of Mikolov et al. (2013) on the skip-gram model for learning high-quality vector representations of words insipred the creation of `doc2vec`, an "unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of text, such as sentences, paragraphs, and documents" - Le & Mikolov (2014).

In this framework, each document is represented by a vector $\vec{d}$ of n dimensions, where n is a fixed parameter. There are two possible `doc2vec` architectures: the DM (Distributed Memory) architecture, where the paragraph vector and word vectors contribute to the prediction task, and the DBOW (Distributed Bag-Of-Words) architecture, where the paracgraph vectors alone are trained to predict words in a window. From the set of `doc2vec` parameters, I have selected the following for tuning the model:

- `dm`: whether the model should use DM or DBOW architecture
- `vector_size`: dimension of feature vectors
- `window`: size of left-right context window
- `min_count`: minimum frequency threshold for words

[1] Word count performed by the TexCount web interface: `https://app.uio.no/ifi/texcount/online.php`

	unigrams + presence	doc2vec
unigrams + frequency	2.00×10^{-4}	2.00×10^{-4}
unigrams + presence	1.00	2.00×10^{-4}
doc2vec	2.00×10^{-4}	

Table 1: p-values from Monte-Carlo permutation test ($R = 5000$) with confidence level $\alpha = 0.01$

- `hs`: whether to use hierarchical softmax
- `epochs`: number of training epochs
- `negative`: number of negative word samples

3 Method

This paper evaluates SVM classifiers trained on unigrams and employing both feature frequency and presence. It then changes the implementation to include feature vectors inferred from a trained `doc2vec` model.

The classifiers are trained on a dataset of 900 positive and 900 negative Imdb movie reviews provided in the framework of the Part II NLP unit of assessment. The `doc2vec` model's training corpus consists of 100,000 movie reviews provided by the Standford Large Movie Review Dataset – Maas et al. (2011).

The `gensim`[2] library was used for implementing `doc2vec`. The originally 2000 Imdb labelled reviews were employed as follows: a balanced 10% served as validation set for parameter tuning. Selecting a set and respectively a range of potential values for each parameter and generating the Cartesian Product and respectively selecting each value randomly[3], I have through these two methods trained a `doc2vec` model for each septuple[4] using the Stanford Large Movie Review Dataset. This resulted in 64 models, each of which was subsequently used for inferring feature vectors when training an SVM classifier on the remaining 90% of the dataset. All documents were tokenised using `gensim`'s `simple_preprocess` utility function and the 64 resulting accuracies are plotted in Figure 1.

Table 2 shows the optimal[5] set of parameters for which the SVM model achieves an accuracy of 86.50% on the validation set.

Parameter	Value
`dm`	0
`vector_size`	141
`window`	13
`min_count`	13
`hs`	0
`epochs`	13
`negative`	9

Table 2: Optimal parameters

Once the highest-accuracy model has been selected, I discarded the validation set and trained 3 SVM classifiers (unigrams + frequency, unigrams + presence, `doc2vec` embeddings) on the remaining 1,800 reviews using 10-fold cross-validation. Average accuracies are presented in Table 3.

4 Results

Embedding	Accuracy
unigrams + frequency	76.33%
unigrams + presence	87.00%
doc2vec	88.22%

Table 3: 10-fold cross-validation mean accuracies

The results confirm that `doc2vec` embeddings improve the performance of both frequency-based and presence-based unigrams-trained systems. The improvements are significant under a Monte Carlo permutation test ($R = 5000, \alpha = 0.01$), with p-values of 1.99×10^{-4} and 1.99×10^{-4} respectively, as depicted in Table 1. The `doc2vec`-trained SVM achieves the highest accuracy, of 88.22%.

[2] `https://radimrehurek.com/gensim/models/doc2vec.html`
[3] using Python's `random.randint`
[4] `[dm, vector_size, window, min_count, hs, epochs, negative]`
[5] to the extent of the search space

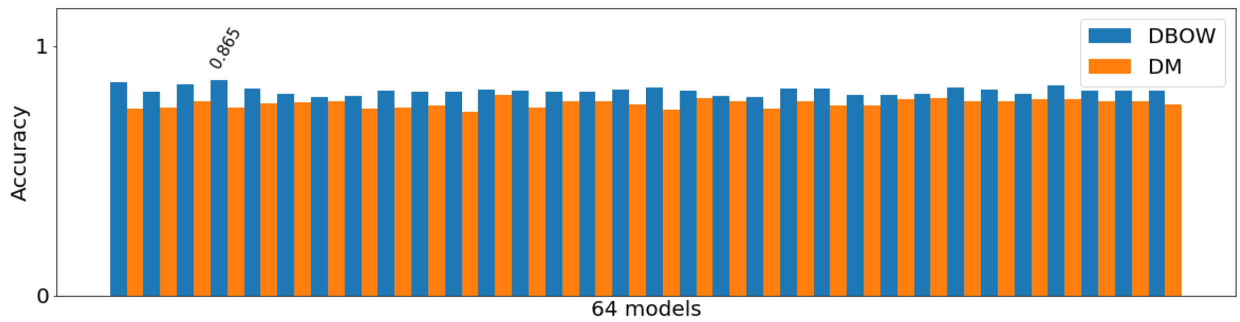

Figure 1: SVM accuracies for 64 `doc2vec` models, each trained with different parameters

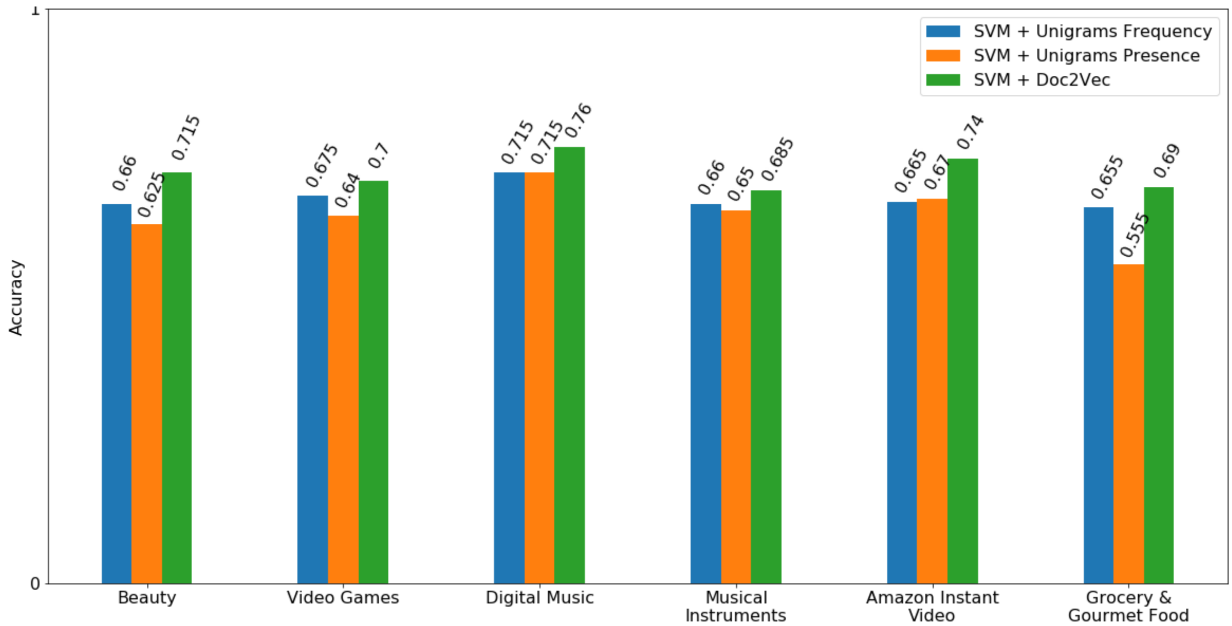

Figure 2: Performance of the 3 systems on 6 new domains, on balanced test sets of 200 reviews

5 Exploration

In order to evaluate the three systems on new categories of data, I used a subset of an Amazon corpus[6] of reviews for different types of products. The dataset categories are distinguished through the reviews being split in different JSON files and the polarity of each review can be inferred from its ranking. For this experiment, I attribute a POSITIVE label to any review with a ranking of 5 and a NEGATIVE label to any with a ranking of 1.

I parse the JSON files by extracting truples of (sentiment, review name, list of features). The list of features is once again obtained through `gensim`'s `simple_preprocess` function. The 3 SVM models from before are once again trained on the 1,800 Imdb labelled reviews[7]. Testing is carried out on randomly selected[8] balanced samples of 200 reviews[9] of each category. From the 24 available categories, I have selected 6 and plotted the accuracies for each system in Figure 2.

As expected, `doc2vec` outperforms the other 2 systems in each case, with a highest accuracy of 76%, on the Digital Music category. The differences are all significant under a Monte Carlo permutation test ($R = 5000$, $\alpha = 0.01$). The second highest accuracy is achieved on Amazon Instant Video reviews – 74%. That is unsurprising given the relatedness of video reviews and movie reviews in terms of employed vocabulary and specialised language. Similarly, less related cate-

[6] `http://jmcauley.ucsd.edu/data/amazon/`
[7] with the validation set discarded
[8] using `numpy.random.choice`
[9] to match the previous test sets used in cross-validation on the 2,000 reviews

gories such as Musical Instruments and Gourmet Food lead to lower accuracies (of 68.5% and 69% respectively).

6 Conclusion

This experiment confirms that the use of `doc2vec` embeddings for SVM classification outperforms both frequency and presence-based bag-of-words approaches. Furthermore, it supports the hypothesis that the `doc2vec` system will prove significantly better when tested on new categories.

References

Joachims, T. (1999), Making large-scale SVM learning practical, *in* B. Schölkopf, C. Burges & A. Smola, eds, 'Advances in Kernel Methods - Support Vector Learning', MIT Press, Cambridge, MA, chapter 11, pp. 169–184.

Le, Q. & Mikolov, T. (2014), Distributed representations of sentences and documents, *in* 'International conference on machine learning', pp. 1188–1196.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. & Potts, C. (2011), Learning word vectors for sentiment analysis, *in* 'Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies', Association for Computational Linguistics, Portland, Oregon, USA, pp. 142–150.
URL: *http://www.aclweb.org/anthology/P11-1015*

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, *in* 'Advances in neural information processing systems', pp. 3111–3119.

Pang, B., Lee, L. & Vaithyanathan, S. (2002), Thumbs up?: sentiment classification using machine learning techniques, *in* 'Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10', Association for Computational Linguistics, pp. 79–86.