An Exploration into the Semantic Space of doc2vec

Sumaiyah Kola
December 6, 2019

1 Introduction

Text understanding relies on first deriving a machine-understandable representation that can accurately capture the semantics.

Vector Space Models aim to capture semantics by embedding documents as points in a shared vector space, the semantic space, mapping similar documents to nearby points. Although the vectors do not correspond to any traditional notion of meaning, I aim to show that the semantic space generated by doc2vec, a neural embedding technique, has preserved meaning in the context of sentiment classification and semantic relatedness. I use the term 'document' irrespective of granularity.

2 Background

2.1 Support Vector Machines - SVM

SVMs are a generalisation of simple maximal margin classifiers formally defined by a separating hyperplane. Given labelled training data in the form of n-dimensional vectors, the algorithm outputs an optimal $(n-1)$-dimensional hyperplane maximising the margin between vectors in each class. The predicted class of a test instance is determined by which side of the hyperplane it falls on.

The input vectors can be generated using the methods below:

2.1.1 Bag of Words - BoW

The standard BoW framework treats documents as a set, $f_1, f_2, \ldots, f_m$, of m possible features. Although sentence semantics and syntactic order are lost, the document vectors generated can produce highly competitive baselines (Pang et al., 2002).

However, BoW document vectors fail to capture similarity between documents and suffer from sparsity and high dimensionality.

2.1.2 Doc2Vec

Neural embeddings, first proposed in (Bengio et al., 2003), use distributional statistics to extract semantic information from word co-occurrences producing dense, low-dimensional, fixed-length vectors. (Mikolov et al., 2013) proposed an efficient neural approach, word2vec, that learns distributed vector representations of words from their contexts and is completely unsupervised. 'Context' refers to a specified k number of words surrounding a given word.

In doc2vec, (Le and Mikolov, 2014) generalise this idea to learn vector representations for documents. In this framework each word and paragraph in the document are mapped to unique vectors: a column, in word matrix W and matrix D respectively.

As with word2vec, there a 2 architectures: dm (Figure 1), based on word2vec:cbow, predicts the centre word by concatenating/averaging the context words and paragraph id.

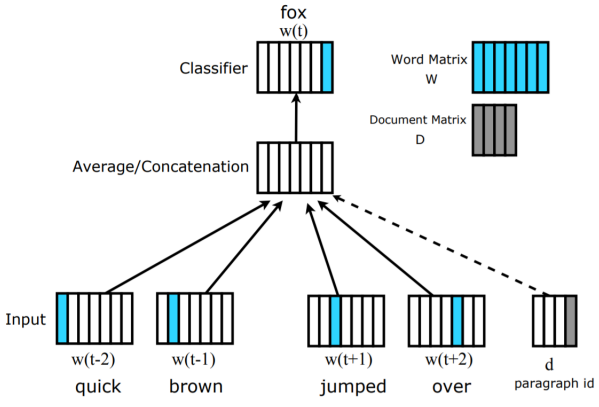

Figure 1: Distributed Bag of Words architecture

Conversely, dbow (Figure 2), based on word2vec:skip-gram, predicts the context words given only the paragraph id.

Using doc2vec, (Le and Mikolov, 2014) report accuracies over 90% for sentiment classifica-

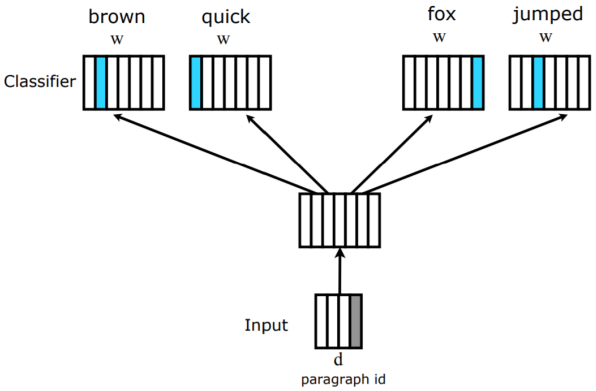

Figure 2: Distributed Bag of Words architecture

tion.

3 Method

I evaluate doc2vec in two contexts: sentiment analysis and semantic relatedness.

In all cases, pre-processing involved stemming (Porter, 1997), stop-word removal, tokenizing and case normalisation.

3.1 Sentiment Analysis

Dataset A is publicly available from 'IMDB' and consists of 100,000 movie reviews. Dataset B, an extended version of that proposed in (Pang and Lee, 2004), was provided in the framework of an NLP course. It contains 2000 labelled, $\{positive, negative\}$ movie reviews. Both datasets are balanced.

I use dataset A to train doc2vec models from which document embeddings for dataset B are inferred and used as input to a SVM classifier. Any labels in dataset A are ignored as doc2vec is unsupervised.

Dataset B is first partitioned: 90% train and 10% validation used for doc2vec hyper-parameter tuning.

Guided by the findings in (Lau and Baldwin, 2016), I navigate the hyper-parameter space linearly, optimising one at a time. The success of a model is defined by the performance of the SVM on validation data.

Once I locate optimal hyper-parameters, validation data is discarded. The performance of the optimal model is evaluated using 10-fold cross validation on the remainder of dataset B, 1800 balanced examples.

3.2 Semantic Relatedness

I aim to use doc2vec to identify duplicate question pairs. The public Quora[1] dataset provides 400,000 pairs of questions marked as duplicates or non-duplicates. Each question has an average length of 11 words.

Using the optimal parameters found earlier, a doc2vec model is trained with 100,000 question strings.

From the remaining data, I construct 50,000 question triplets in which two are duplicates and the third randomly chosen. To predict the duplicate pair in each triplet, I infer document vectors from the model and use cosine similarity,

$$cosine(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \, \|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

to find the closest pair.

A baseline system was established using the Jaccard distance,

$$jaccard(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} = \frac{|X \cap Y|}{|X| + |Y| - |X \cap Y|}$$

to measure similarity between the features in each question.

4 Results

4.1 Sentiment Analysis

Table 1 presents a small portion the results of doc2vec hyper-parameter tuning with line 8 presenting an optimal dbow model with a vector size of 100 and window size of 10.

Table 2 reports the results of the optimal doc2vec model. A BoW model constructed in my previous work (Kola, 2019) was used as a baseline. Table 3 shows significant[2] performance improvement (p-value of 0.077) of doc2vec over the BoW system. I also found that the doc2vec model preferred unstemmed data (p-value of 0.187) over stemmed data. The model has captured sentiment in the embeddings generated as it has improved upon the supervised BoW baseline.

Figure 3 presents the spread of predictions for optimal systems D (BoW) and A (doc2vec). It shows the average confidence value of doc2vec is larger than that of BoW implying doc2vec

[1] src: www.kaggle.com/c/quora-question-pairs/data

[2] Significance reported with a two-tailed permutation test where $\alpha = 0.08$

	dm/dbow	Vector Size	Epochs	Min Count	Sub Sampling	Window Size	Accuracy
(1)	dm	50	5	2	0	10	53
(2)	dm	100	5	2	0	10	55
(3)	dm	150	5	2	0	10	57
(4)	dbow	50	5	2	0	10	58
(5)	dbow	100	5	2	0	10	61
(6)	dbow	150	5	2	0	10	59
(7)	dbow	100	10	2	0	10	70
(8)	dbow	100	20	2	0	10	84
(9)	dbow	100	30	2	0	10	77
(10)	dbow	100	20	4	0	10	73
(11)	dbow	100	20	6	0	10	72
(12)	dbow	100	20	2	10^{-5}	10	79
(13)	dbow	100	20	2	0	5	74

Table 1: Doc2vec hyper-parameter tuning. Validation set accuracies, in percent

id	Classifier	Stem Data	Accuracy
A	doc2vec-SVM	Y	86.6
B	BoW-SVM	Y	85.2
C	doc2vec-SVM	N	88.3
D	BoW-SVM	N	86.3

Table 2: 10-fold cross validation accuracies, in percent

	A	B	C	D
A		0.237	0.187	0.831
B			0.077	0.435
C				0.239
D				

Table 3: Comparison of best systems, p-values to 3dp

better captures sentence sentiment semantics as it is more confident in the case it classifies correctly.

From the 70 instances misclassified by both systems, I analysed those with the top confidence scores over both systems. I found a trend of reviews in which the majority of the text was a regurgitation of the film plot. For example reviews of horror movies although positive, mention gruesome and unpleasant plot details. Humans can easily separate plot-content when inferring the tone of the overall document.

Out of the 198 instances classified correctly by doc2vec and incorrectly by BoW, I collated those that doc2vec classified with the most confidence. Interestingly 4 out of the top 8 positive reviews had many elements of sarcasm that doc2vec had managed to decipher as still being positive.

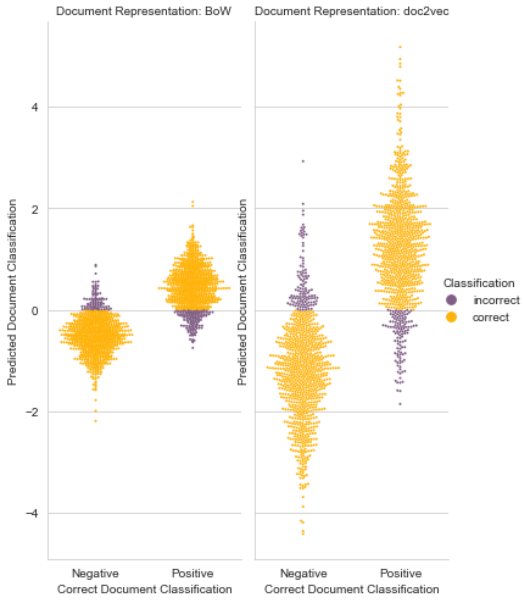

Figure 3: SVM document prediction results

4.2 Semantic Relatedness

Table 4 shows the performance of doc2vec in the task of semantic relatedness.
The task of choosing a duplicate pair of questions from a triplet has a random baseline of 33%. The Jaccard distance between feature sets for each question provided a baseline of 81%. The doc2vec model proved significantly better than this (p-value <0.001).

The T-SNE visualisation confirms that duplicate questions are grouped in the semantic space generated by doc2vec.

Document Representation	Distance Metric	Accuracy
doc2vec	Cosine	97
BoW	Jaccard	81

Table 4: Accuracy, in percent

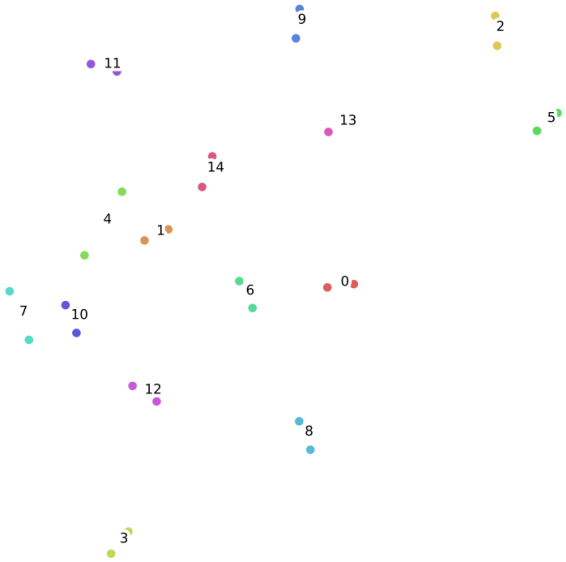

Figure 4: T-SNE of duplicate questions

5 Conclusion

In this paper[3] I evaluate the success of doc2vec embeddings as a document representation technique. Although unsupervised, the document embeddings retain notions of meaning present in the original documents. Predefined notions of meaning, such as semantic similarity and sentiment expressed are be preserved in the representation.
In both cases, the contextual document embeddings generated by doc2vec proved to outperform their baselines.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.

Sumaiyah Kola. 2019. Sentiment classification with machine learning techniques.

Jey Han Lau and Timothy Baldwin. 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. *CoRR*, abs/1607.05368.

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

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

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

M. F. Porter. 1997. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[3]Word Count: 996 calculated using https://app.uio.no/ifi/texcount/