An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation

Jey Han Lau\textsuperscript{1,2} and Timothy Baldwin\textsuperscript{2}
\textsuperscript{1} IBM Research
\textsuperscript{2} Dept of Computing and Information Systems, The University of Melbourne
jeyhan.lau@gmail.com, tb@ldwin.net

Abstract

Recently, Le and Mikolov (2014) proposed doc2vec as an extension to word2vec (Mikolov et al., 2013a) to learn document-level embeddings. Despite promising results in the original paper, others have struggled to reproduce those results. This paper presents a rigorous empirical evaluation of doc2vec over two tasks. We compare doc2vec to two baselines and two state-of-the-art document embedding methodologies. We found that doc2vec performs robustly when using models trained on large external corpora, and can be further improved by using pre-trained word embeddings. We also provide recommendations on hyper-parameter settings for general-purpose applications, and release source code to induce document embeddings using our trained doc2vec models.

1 Introduction

Neural embeddings were first proposed by Bengio et al. (2003), in the form of a feed-forward neural network language model. Modern methods use a simpler and more efficient neural architecture to learn word vectors (word2vec: Mikolov et al. (2013b); GloVe: Pennington et al. (2014)), based on objective functions that are designed specifically to produce high-quality vectors.

Neural embeddings learnt by these methods have been applied in a myriad of NLP applications, including initialising neural network models for objective visual recognition (Frome et al., 2013) or machine translation (Zhang et al., 2014; Li et al., 2014), as well as directly modelling word-to-word relationships (Mikolov et al., 2013a; Zhao et al., 2015; Salehi et al., 2015; Vylomova et al., to appear).

Paragraph vectors, or doc2vec, were proposed by Le and Mikolov (2014) as a simple extension to word2vec to extend the learning of embeddings from words to word sequences.\textsuperscript{1} doc2vec is agnostic to the granularity of the word sequence — it can equally be a word n-gram, sentence, paragraph or document. In this paper, we use the term “document embedding” to refer to the embedding of a word sequence, irrespective of its granularity.

doc2vec was proposed in two forms: dbow and dmpv. dbow is a simpler model and ignores word order, while dmpv is a more complex model with more parameters (see Section 2 for details). Although Le and Mikolov (2014) found that as a standalone method dmpv is a better model, others have reported contradictory results.\textsuperscript{2} doc2vec has also been reported to produce sub-par performance compared to vector averaging methods based on informal experiments.\textsuperscript{3} Additionally, while Le and Mikolov (2014) report state-of-the-art results over a sentiment analysis task using doc2vec, others (including the second author of the original paper in follow-up work) have struggled to replicate this result.\textsuperscript{4}

Given this background of uncertainty regarding the true effectiveness of doc2vec and confusion about performance differences between dbow and dmpv, we aim to shed light on a number of em-

\textsuperscript{1} The term doc2vec was popularised by Gensim (\v{R}eh\u0161ek and Sojka, 2010), a widely-used implementation of paragraph vectors: https://radimrehurek.com/gensim/

\textsuperscript{2} The authors of Gensim found dbow outperforms dmpv: https://github.com/piskvorky/gensim/blob/develop/docs/notebooks/doc2vec-IMDB.ipynb

\textsuperscript{3} https://groups.google.com/forum/#!topic/gensim/bE6skaT45FQ

\textsuperscript{4} For a detailed discussion on replicating the results of Le and Mikolov (2014), see: https://groups.google.com/forum/#!topic/word2vec-toolkit/Q49F1rNOQRo
pirical questions: (1) how effective is doc2vec in different task settings?; (2) which is better out of dmpv and dbow?; (3) is it possible to improve doc2vec through careful hyper-parameter optimisation or with pre-trained word embeddings?; and (4) can doc2vec be used as an off-the-shelf model like word2vec? To this end, we present a formal and rigorous evaluation of doc2vec over two extrinsic tasks. Our findings reveal that dbow, despite being the simpler model, is superior to dmpv. When trained over large external corpora, with pre-trained word embeddings and hyper-parameter tuning, we find that doc2vec performs very strongly compared to both a simple word embedding averaging and n-gram baseline, as well as two state-of-the-art document embedding approaches, and that doc2vec performs particularly strongly over longer documents. We additionally release source code for replicating our experiments, and for inducing document embeddings using our trained models.

2 Related Work

word2vec was proposed as an efficient neural approach to learning high-quality embeddings for words (Mikolov et al., 2013a). Negative sampling was subsequently introduced as an alternative to the more complex hierarchical softmax step at the output layer, with the authors finding that not only is it more efficient, but actually produces better word vectors on average (Mikolov et al., 2013b).

The objective function of word2vec is to maximise the log probability of context word \( (w_0) \) given its input word \( (w_I) \), i.e. \( \log P(w_0|w_I) \). With negative sampling, the objective is to maximise the dot product of the \( w_I \) and \( w_0 \) while minimising the dot product of \( w_I \) and randomly sampled “negative” words. Formally, \( \log P(w_0|w_I) \) is given as follows:

\[
\log \sigma(v'_w \top v_{w_I}) + \sum_{i=1}^{k} w_i \sim P_n(w) \left[ \log \sigma(-v'_w \top v_{w_I}) \right] \tag{1}
\]

where \( \sigma \) is the sigmoid function, \( k \) is the number of negative samples, \( P_n(w) \) is the noise distribution, \( v_w \) is the vector of word \( w \), and \( v'_w \) is the negative sample vector of word \( w \).

There are two approaches within word2vec: skip-gram (“sg”) and cbow. In skip-gram, the input is a word (i.e. \( v_{w_I} \) is a vector of one word) and the output is a context word. For each input word, the number of left or right context words to predict is defined by the window size hyper-parameter. cbow is different to skip-gram in one aspect: the input consists of multiple words that are combined via vector addition to predict the context word (i.e. \( v_{w_I} \) is a summed vector of several words).

doc2vec is an extension to word2vec for learning document embeddings (Le and Mikolov, 2014). There are two approaches within doc2vec: dbow and dmpv.

dbow works in the same way as skip-gram, except that the input is replaced by a special token representing the document (i.e. \( v_{w_I} \) is a vector representing the document). In this architecture, the order of words in the document is ignored; hence the name distributed bag of words.

dmpv works in a similar way to cbow. For the input, dmpv introduces an additional document token in addition to multiple target words. Unlike cbow, however, these vectors are not summed but concatenated (i.e. \( v_{w_I} \) is a concatenated vector containing the document token and several target words). The objective is again to predict a context word given the concatenated document and word vectors.

More recently, Kiros et al. (2015) proposed skip-thought as a means of learning document embeddings. skip-thought vectors are inspired by abstracting the distributional hypothesis from the word level to the sentence level. Using an encoder-decoder neural network architecture, the encoder learns a dense vector presentation of a sentence, and the decoder takes this encoding and decodes it by predicting words of its next (or previous) sentence. Both the encoder and decoder use a gated recurrent neural network language model. Evaluating over a range of tasks, the authors found that skip-thought vectors perform very well against state-of-the-art task-optimised methods.

Wieting et al. (2016) proposed a more direct way of learning document embeddings, based on a large-scale training set of paraphrase pairs from the Paraphrase Database (PPDB: Ganitkevitch et al. (2013)). Given a paraphrase pair, word embeddings and a method to compose the word embeddings for a sentence embedding, the objective function of the neural network model is to optimise the word embeddings such that the cosine similarity of the sentence embeddings for the pair
is maximised. The authors explore several methods of combining word embeddings, and found that simple averaging produces the best performance.

3 Evaluation Tasks

We evaluate doc2vec in two task settings, specifically chosen to highlight the impact of document length on model performance.

For all tasks, we split the dataset into 2 partitions: development and test. The development set is used to optimise the hyper-parameters of doc2vec, and results are reported on the test set. We use all documents in the development and test set (and potentially more background documents, where explicitly mentioned) to train doc2vec. Our rationale for this is that the doc2vec training is completely unsupervised, i.e. the model takes only raw text and uses no supervised or annotated information, and thus there is no need to hold out the test data, as it is unlabelled. We ultimately relax this assumption in the next section (Section 4), when we train doc2vec using large external corpora.

After training doc2vec, document embeddings are generated by the model. For the word2vec baseline, we compute a document embedding by taking the component-wise mean of its component word embeddings. We experiment with both variants of doc2vec (dbow and dmpv) and word2vec (skip-gram and cbow) for all tasks.

In addition to word2vec, we experiment with another baseline model that converts a document into a distribution over words via maximum likelihood estimation, and compute pairwise document similarity using the Jensen Shannon divergence. For word types we explore n-grams of order \( n = \{1, 2, 3, 4\} \) and find that a combination of unigrams, bigrams and trigrams achieves the best results. Henceforth, this second baseline will be referred to as ngram.

3.1 Forum Question Duplication

We first evaluate doc2vec over the task of duplicate question detection in a web forum setting, using the dataset of Hoogeveen et al. (2015). The dataset has 12 subforums extracted from StackExchange, and provides training and test splits in two experimental settings: retrieval and classification. We use the classification setting, where the goal is to classify whether a given question pair is a duplicate.

The dataset is separated into the 12 subforums, with a pre-compiled training–test split per subforum; the total number of instances (question pairs) ranges from 50M to 1B pairs for the training partitions, and 30M to 300M pairs for the test partitions, depending on the subforum. The proportion of true duplicate pairs is very small in each subforum, but the setup is intended to respect the distribution of true duplicate pairs in a real-world setting.

We sub-sample the test partition to create a smaller test partition that has 10M document pairs. On average across all twelve subforums, there are 22 true positive pairs per 10M question pairs. We also create a smaller development partition from the training partition by randomly selecting 300 positive and 3000 negative pairs. We optimise the hyper-parameters of doc2vec and word2vec using the development partition on the tex subforum, and apply the same hyper-parameter settings for all subforums when evaluating over the test pairs. We use both the question title and body as document content: on average the test document length is approximately 130 words. We use the default tokenised and lowercased words given by the dataset. All test, development and un-sampled documents are pooled together during model training, and each subforum is trained separately.

We compute cosine similarity between documents using the vectors produced by doc2vec and word2vec to score a document pair. We then sort the document pairs in descending order of similarity score, and evaluate using the area under the curve (AUC) of the receiver operating characteristic (ROC) curve. The ROC curve tracks the true positive rate against the false positive rate at each point of the ranking, and as such works well for heavily-skewed datasets. An AUC score of 1.0 implies that all true positive pairs are ranked before true negative pairs, while an AUC score of .5 indicates a random ranking. We present the full results for each subforum in Table 1.

---

\(^5\) We multiply the divergence value by –1.0 to invert the value, so that a higher value indicates greater similarity.

\(^6\) That is, the probability distribution is computed over the union of unigrams, bigrams and trigrams in the paired documents.

\(^7\) Uniform random sampling is used so as to respect the original distribution.
Comparing doc2vec and word2vec to ngram, both embedding methods perform substantially better in most domains, with two exceptions (english and gis), where ngram has comparable performance.

doc2vec outperforms word2vec embeddings in all subforums except for gis. Despite the skewed distribution, simple cosine similarity based on doc2vec embeddings is able to detect these duplicate document pairs with a high degree of accuracy. dbow performs better than or as well as dmpv in 9 out of the 12 subforums, showing that the simpler dbow is superior to dmpv.

One interesting exception is the english subforum, where dmpv is substantially better, and ngram — which uses only surface word forms — also performs very well. We hypothesise that the order and the surface form of words possibly has a stronger role in this subforum, as questions are often about grammar problems and as such the position and semantics of words is less predictable (e.g. Where does “for the same” come from?)

Hyper-Parameter Description

| Hyper-Parameter | Description |
|-----------------|-------------|
| Vector Size     | Dimension of word vectors |
| Window Size     | Left/right context window size |
| Min Count       | Minimum frequency threshold for word types |
| Sub-sampling    | Threshold to downsample high frequency words |
| Negative Sample | No. of negative word samples |
| Epoch           | Number of training epochs |

Table 3: A description of doc2vec hyper-parameters.

3.2 Semantic Textual Similarity

The Semantic Textual Similarity (STS) task is a shared task held as part of *SEM and SemEval over a number of iterations (Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015). In STS, the goal is to automatically predict the similarity of a pair of sentences in the range \([0, 5]\), where 0 indicates no similarity whatsoever and 5 indicates semantic equivalence.

The top systems utilise word alignment, and further optimise their scores using supervised learning (Agirre et al., 2015). Word embeddings are employed, although sentence embeddings are often taken as the average of word embeddings (e.g. Sultan et al. (2015)).

We evaluate doc2vec and word2vec embeddings over the English STS sub-task of SemEval-2015 (Agirre et al., 2015). The dataset has 5 domains, and each domain has 375–750 annotated pairs. Sentences are much shorter than our previous task, at an average of only 13 words in each test sentence.

As the dataset is also much smaller, we combine sentences from all 5 domains and also sentences from previous years (2012–2014) to form the training data. We use the headlines domain from 2014 as development, and test on all 2015 domains. For pre-processing, we tokenise and lowercase the words using Stanford CoreNLP (Manning et al., 2014).

As a benchmark, we include results from the overall top-performing system in the competition, referred to as “DLS” (Sultan et al., 2015). Note, however, that this system is supervised and highly customised to the task, whereas our methods are completely unsupervised. Results are presented in Table 2.

Unsurprisingly, we do not exceed the overall performance of the supervised benchmark system DLS, although doc2vec outperforms DLS over
Table 4: Optimal doc2vec hyper-parameter values used for each tasks. “Training size” is the total word count in the training data. For Q-Dup training size is an average word count across all subforums.

| Method | Task  | Training Size | Vector Size | Window Size | Min Count | Sub-Sampling | Negative Sample | Epoch |
|--------|-------|---------------|-------------|-------------|-----------|--------------|-----------------|-------|
| dbow   | Q-Dup | 4.3M          | 300         | 15          | 5         | $10^{-5}$    | 5               | 20    |
| STS    | .5M   | 300           | 15          | 1           | $10^{-5}$ | 5             | 400             |
| dmpv   | Q-Dup | 4.3M          | 300         | 3           | 5         | $10^{-6}$    | 5               | 600   |
| STS    | .5M   | 300           | 5           | 1           | $10^{-6}$ | 5             | 1000            |

the domain of belief. ngram performs substantially worse than all methods (with an exception in ans-students where it outperforms dmpv and dbow).

Comparing doc2vec and word2vec, doc2vec performs better. However, the performance gap is lower compared to the previous two tasks, suggesting that the benefit of using doc2vec is diminished for shorter documents. Comparing dbow and dmpv, the difference is marginal, although dbow as a whole is slightly stronger, consistent with the observation of previous task.

3.3 Optimal Hyper-parameter Settings

Across the two tasks, we found that the optimal hyper-parameter settings (as described in Table 3) are fairly consistent for dbow and dmpv, as detailed in Table 4 (task abbreviations: Q-Dup = Forum Question Duplication (Section 3.1); and STS = Semantic Textual Similarity (Section 3.2)). Note that we did not tune the initial and minimum learning rates ($\alpha$ and $\alpha_{min}$, respectively), and use the following values for all experiments: $\alpha = .025$ and $\alpha_{min} = .0001$. The learning rate decreases linearly per epoch from the initial rate to the minimum rate.

In general, dbow favours longer windows for context words than dmpv. Possibly the most important hyper-parameter is the sub-sampling threshold for high frequency words: in our experiments we find that task performance dips considerably when a sub-optimal value is used. dmpv also requires more training epochs than dbow. As a rule of thumb, for dmpv to reach convergence, the number of epochs is one order of magnitude larger than dbow. Given that dmpv has more parameters in the model, this is perhaps not a surprising finding.

4 Training with Large External Corpora

In Section 3, all tasks were trained using small in-domain document collections. doc2vec is designed to scale to large data, and we explore the effectiveness of doc2vec by training it on large external corpora in this section.

We experiment with two external corpora: (1) WIKI, the full collection of English Wikipedia, and (2) AP-NEWS, a collection of Associated Press English news articles from 2009 to 2015. We tokenise and lowercase the documents using Stanford CoreNLP (Manning et al., 2014), and treat each natural paragraph of an article as a document for doc2vec. After pre-processing, we have approximately 35M documents and 2B tokens for WIKI, and 25M and .9B tokens for AP-NEWS. Seeing that dbow trains faster and is a better model than dmpv from Section 3, we experiment with only dbow here.

To test if doc2vec can be used as an off-the-shelf model, we take a pre-trained model and infer an embedding for a new document without updating the hidden layer word weights. We have three hyper-parameters for test inference: initial learning rate ($\alpha$), minimum learning rate ($\alpha_{min}$), and number of inference epochs. We optimise these parameters using the development partitions in each task; in general a small initial $\alpha$ (=.01) with low $\alpha_{min}$ (=.0001) and large epoch number (= 500–1000) works well.

For word2vec, we train skip-gram on the

---

8Using the dump dated 2015-12-01, cleaned using WikiExtractor: https://github.com/attardi/wikiextractor

9We use these hyper-parameter values for WIKI (AP-NEWS): vector size = 300 (300), window size = 15 (15), min count = 20 (10), sub-sampling threshold = $10^{-5}$ ($10^{-7}$), negative sample = 5, epoch = 20 (30). After removing low frequency words, the vocabulary size is approximately 670K for WIKI and 300K for AP-NEWS.

10That is, test data is held out and not including as part of doc2vec training.
same corpora. We also include the word vectors trained on the larger Google News by Mikolov et al. (2013b), which has 100B words. The Google News skip-gram vectors will henceforth be referred to as GL-NEWS.

dbow, skip-gram and ngram results for all two tasks are presented in Table 5. Between the baselines ngram and skip-gram, ngram appears to do better over Q-Dup, while skip-gram works better over STS.

As before, doc2vec outperforms word2vec and ngram across almost all tasks. For tasks with longer documents (Q-Dup), the performance gap between doc2vec and word2vec is more pronounced, while for STS, which has shorter documents, the gap is smaller. In some STS domains (e.g. ans-students) word2vec performs just as well as doc2vec. Interestingly, we see that GL-NEWS word2vec embeddings perform worse than our WIKI and AP-NEWS word2vec embeddings, even though the Google News corpus is orders of magnitude larger.

Comparing doc2vec results with in-domain results (1 and 2), the performance is in general lower. As a whole, the performance difference between the dbow models trained using WIKI and AP-NEWS is not very large, indicating the robustness of these large external corpora for general-purpose applications. To facilitate applications using off-the-shelf doc2vec models, we have publicly released code and trained models to induce document embeddings using the WIKI and AP-NEWS dbow models.

### 4.1 Comparison with Other Document Embedding Methodologies

We next calibrate the results for doc2vec against skip-thought (Kiros et al., 2015) and paragram-phrase (pp: Wieting et al. (2016)), two recently-proposed competitor document embedding methods. For skip-thought, we use the pre-trained model made available by the authors, based on the BOOK-CORPUS dataset (Zhu et al., 2015); for pp, once again we use the pre-trained model from the authors, based on PPDB (Ganitkevitch et al., 2013). We compare these two models against dbow trained on each of WIKI and AP-NEWS. The results are presented in Table 5, along with results for the baseline method of skip-gram and ngram.

**skip-thought** performs poorly: its performance is worse than the simpler method of word2vec vector averaging and ngram. dbow outperforms pp over most Q-Dup subforums, although the situation is reversed for STS. Given that pp is based on word vector averaging, these observations support the conclusion that vector averaging methods work best for shorter documents, while dbow handles longer documents better.

It is worth noting that doc2vec has the upper-

| Task | Metric | Domain | pp | skip-thought | dbow | skip-gram | ngram |
|------|--------|--------|----|--------------|------|----------|------|
| Q-Dup | AUC | android | .92 | .57 | .96 | .94 | .77 | .76 | .72 | .80 |
| | | english | .82 | .56 | .80 | .81 | .62 | .63 | .61 | .84 |
| | | gaming | .96 | .70 | .95 | .93 | .88 | .85 | .83 | .94 |
| | | gis | .89 | .58 | .85 | .86 | .79 | .83 | .79 | .92 |
| | | mathematics | .80 | .57 | .84 | .80 | .65 | .58 | .59 | .70 |
| | | physics | .97 | .61 | .92 | .94 | .81 | .77 | .74 | .88 |
| | | programmers | .88 | .69 | .93 | .88 | .75 | .72 | .64 | .68 |
| | | stats | .87 | .60 | .92 | .98 | .70 | .72 | .66 | .77 |
| | | tex | .88 | .65 | .89 | .82 | .75 | .64 | .73 | .78 |
| | | unix | .86 | .74 | .85 | .94 | .78 | .72 | .66 | .75 |
| | | webmasters | .89 | .53 | .89 | .91 | .77 | .73 | .71 | .79 |
| | | wordpress | .83 | .66 | .99 | .98 | .61 | .58 | .58 | .87 |
| | | headlines | .77 | .44 | .73 | .75 | .73 | .74 | .66 | .81 |
| STS | r | ans-forums | .67 | .35 | .59 | .60 | .46 | .44 | .42 | .50 |
| | | belief | .78 | .33 | .65 | .69 | .67 | .69 | .65 | .65 |
| | | images | .83 | .18 | .80 | .78 | .72 | .73 | .69 | .62 |

Table 5: Results over all two tasks using models trained with external corpora.

---

11Hyper-parameter values for WIKI (AP-NEWS): vector size = 300 (300), window size = 5 (5), min count = 20 (10), sub-sampling threshold = $10^{-5}$ (10^{-5}), negative sample = 5, epoch = 100 (150).
12https://code.google.com/archive/p/word2vec/
13https://github.com/jhlau/doc2vec
hand compared to pp in that it can be trained on in-domain documents. If we compare in-domain doc2vec results (1 and 2) to pp (Table 5), the performance gain on Q-Dup is even more pronounced.

5 Improving doc2vec with Pre-trained Word Embeddings

Although not explicitly mentioned in the original paper (Le and Mikolov, 2014), dbow does not learn embeddings for words in the default configuration. In its implementation (e.g. Gensim), dbow has an option to turn on word embedding learning, by running a step of skip-gram to update word embeddings before running dbow. With the option turned off, word embeddings are randomly initialised and kept at these randomised values.

Even though dbow can in theory work with randomised word embeddings, we found that performance degrades severely under this setting. An intuitive explanation can be traced back to its objective function, which is to maximise the dot product between the document embedding and its constituent word embeddings: if word embeddings are randomly distributed, it becomes more difficult to optimise the document embedding to be close to its more critical content words.

To illustrate this, consider the two-dimensional t-SNE plot (Van der Maaten and Hinton, 2008) of doc2vec document and word embeddings in Figure 1(a). In this case, the word learning option is turned on, and related words form clusters, allowing the document embedding to selectively position itself closer to a particular word cluster (e.g. content words) and distance itself from other clusters (e.g. function words). If word embeddings are randomly distributed on the plane, it would be harder to optimise the document embedding.

Seeing that word vectors are essentially learnt via skip-gram in dbow, we explore the possibility of using externally trained skip-gram word embeddings to initialise the word embeddings in dbow. We repeat the experiments described in Section 3, training the dbow model using the smaller in-domain document collections in each task, but this time initialise the word vectors using pre-trained word2vec embeddings from WIKI and AP-NEWS. The motivation is that with better initialisation, the model could converge faster and improve the quality of embeddings.

Results using pre-trained WIKI and AP-NEWS

| Task     | Domain | dbow     | dbow + WIKI | dbow + AP-NEWS |
|----------|--------|----------|-------------|----------------|
| android  |        | .97      | .99         | .98            |
| english  |        | .84      | .90         | .89            |
| gaming   |        | 1.00     | 1.00        | 1.00           |
| gis      |        | .93      | .92         | .94            |
| mathematica |    | .96      | .96         | .96            |
| physics  |        | .96      | .98         | .97            |
| programmers |    | .93      | .92         | .91            |
| stats    |        | 1.00     | 1.00        | .99            |
| tex      |        | .94      | .95         | .92            |
| unix     |        | .98      | .98         | .97            |
| webmasters |     | .92      | .93         | .93            |
| wordpress|        | .97      | .96         | .98            |
| headlines|        | .77      | .78         | .78            |
| ans-forums |   | .66      | .68         | .68            |
| ans-students |   | .65      | .63         | .65            |
| belief   |        | .76      | .77         | .78            |
| images   |        | .78      | .80         | .79            |

Table 6: Comparison of dbow performance using pre-trained WIKI and AP-NEWS skip-gram embeddings.

skip-gram embeddings are presented in Table 6. Encouragingly, we see that using pre-trained word embeddings helps the training of dbow on the smaller in-domain document collection. Across all tasks, we see an increase in performance. More importantly, using pre-trained word embeddings never harms the performance. Although not detailed in the table, we also find that the number of epochs to achieve optimal performance (based on development data) is fewer than before.

We also experimented with using pre-trained cbow word embeddings for dbow, and found similar observations. This suggests that the initialisation of word embeddings of dbow is not sensitive to a particular word embedding implementation.

6 Discussion

To date, we have focused on quantitative evaluation of doc2vec and word2vec. The qualitative difference between doc2vec and word2vec document embeddings, however, remains unclear. To shed light on what is being learned, we select a random document from STS — tech capital bangalore costliest indian city to live in: survey — and plot the document and word embeddings induced by dbow and skip-gram using t-SNE in Figure 1.14

14We plotted a larger set of sentences as part of this analysis, and found that the general trend was the same across all sentences.
Figure 1: Two-dimensional t-SNE projection of doc2vec and word2vec embeddings.

For word2vec, the document embedding is a centroid of the word embeddings, given the simple word averaging method. With doc2vec, on the other hand, the document embedding is clearly biased towards the content words such as tech, costliest and bangalore, and away from the function words. doc2vec learns this from its objective function with negative sampling: high frequency function words are likely to be selected as negative samples, and so the document embedding will tend to align itself with lower frequency content words.

7 Conclusion

We used two tasks to empirically evaluate the quality of document embeddings learnt by doc2vec, as compared to two baseline methods — word2vec word vector averaging and an n-gram model — and two competitor document embedding methodologies. Overall, we found that doc2vec performs well, and that dbow is a better model than dmpv. We empirically arrived at recommendations on optimal doc2vec hyper-parameter settings for general-purpose applications, and found that doc2vec performs robustly even when trained using large external corpora, and benefits from pre-trained word embeddings. To facilitate the use of doc2vec and enable replication of these results, we release our code and pre-trained models.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In Proceedings of the Second Joint Conference on Lexical and Computational Semantics (*SEM 2013), pages 32–43, Atlanta, USA.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014), pages 81–91, Dublin, Ireland.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 252–263, Denver, USA.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. The Journal of Machine Learning Research, 3:1137–1155.

Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Marc Aurelio Ranzato, and Tomas Mikolov. 2013. DeViSE: A deep visual-semantic embedding model. In Advances in Neural Information Processing Systems 26 (NIPS-13), pages 2121–2129.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013), pages 758–764, Atlanta, USA.
Doris Hoogeveen, Karin Verspoor, and Timothy Baldwin. 2015. CQADupStack: A benchmark data set for community question-answering research. In Proceedings of the Twentieth Australasian Document Computing Symposium (ADCS 2015), pages 3:1–3:8, Sydney, Australia.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antionio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In Advances in Neural Information Processing Systems 28 (NIPS-15), pages 3294–3302, Montreal, Canada.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In Proceedings of the 31st International Conference on Machine Learning (ICML 2014), pages 1188–1196, Beijing, China.

Peng Li, Yang Liu, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. 2014. A neural reordering model for phrase-based translation. In Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014), pages 1897–1907, Dublin, Ireland.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In Association for Computational Linguistics (ACL) System Demonstrations, pages 55–60, Baltimore, USA.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In Proceedings of Workshop at the International Conference on Learning Representations, 2013, Scottsdale, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems, pages 3111–3119.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), pages 1532–1543, Doha, Qatar.

Radim Rehurek and Petr Sojka. 2010. 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.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2015), pages 977–983, Denver, USA.

Md Arafat Sultan, Steven Bethard, and Tamara Sumner. 2015. DLS@CU: Sentence similarity from word alignment and semantic vector composition. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 148–153, Denver, Colorado.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. Journal of Machine Learning Research, 9(2579–2605):85.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. To appear. Take and took, gagle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), Berlin, Germany.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In Proceedings of the International Conference on Learning Representations 2016, San Juan, Puerto Rico.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), pages 111–121, Baltimore, USA.

Jiang Zhao, Man Lan, Zheng-Yu Niu, and Yue Lu. 2015. Integrating word embeddings and traditional nlp features to measure textual entailment and semantic relatedness of sentence pairs. In Proceedings of the International Joint Conference on Neural Networks (IJCNN2015), pages 1–7, Killarney, Ireland.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. Arxiv, abs/1506.06724.