Document Embedding with Paragraph Vectors

Andrew M. Dai
Google
adai@google.com

Christopher Olah
chris@colah.ca

Quoc V. Le
Google
qvl@google.com

Greg S. Corrado
Google
gcorrado@google.com

Abstract

Paragraph Vectors has been recently proposed as an unsupervised method for learning distributed representations for pieces of texts. In their work, the authors showed that the method can learn an embedding of movie review texts which can be leveraged for sentiment analysis. That proof of concept, while encouraging, was rather narrow. Here we consider tasks other than sentiment analysis, provide a more thorough comparison of Paragraph Vectors to other document modelling algorithms such as Latent Dirichlet Allocation, and evaluate performance of the method as we vary the dimensionality of the learned representation. We benchmarked the models on three document similarity tasks, two from Wikipedia, one from arXiv. We observe that the Paragraph Vector method performs significantly better than LDA on two of the tasks, and propose a simple improvement to enhance embedding quality. Somewhat surprisingly, we also show that much like word embeddings, vector operations on Paragraph Vectors can perform useful semantic results.

1 Introduction

Central to many language understanding problems is the question of knowledge representation: How to capture the essential meaning of a document in a machine-understandable format (or "representation"). Despite much work going on in this area, the most established format is perhaps the bag of words (or bag of n-gram) representations [2]. Latent Dirichlet Allocation (LDA) [1] is another widely adopted representation.

A recent paradigm in machine intelligence is to use a distributed representation for words [4] and documents [3]. The interesting part is that even though these representations are less human-interpretable than previous representations, they seem to work well in practice. In particular, Le and Mikolov [3] show that their method, Paragraph Vectors, capture many document semantics in dense vectors and that they can be used in classifying movie reviews or retrieving web pages.

Despite their success, little is known about how well the model works compared to Bag-of-Words or LDA for other unsupervised applications and how sensitive the model is when we change the hyperparameters.

In this paper, we make an attempt to compare Paragraph Vectors with other baselines on two datasets that have significant practical implications. First, we benchmark Paragraph Vectors on the task of Wikipedia browsing: given a Wikipedia article, what are the nearest articles that the audience should browse next. We also test Paragraph Vectors on the task of finding related articles on arXiv. In both of these tasks, we find that Paragraph Vectors allow for finding documents of interest via simple and intuitive vector operations. For example, we can find the Japanese equivalence of "Lady Gaga.".

The goal of the paper is beyond benchmarking: The positive results on Wikipedia and arXiv datasets confirm that having good representations for texts can be powerful when it comes to language understanding. The success in these tasks shows that it is possible to use Paragraph Vectors for local and non-local browsing of large corpora.

We also show a simple yet effective trick to improve Paragraph Vector. In particular, we observe that by jointly training word embeddings, as in the skip gram model, the quality of the paragraph vectors is improved.

2 Model

The Paragraph Vector model is first proposed in [3]. The model inserts a memory vector to the standard language model which aims at capturing the topics of the document. The authors named this model "Distributed Memory":

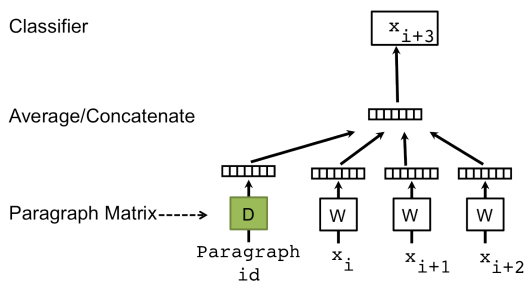

Figure 1: The distributed memory model of Paragraph Vector for an input sentence.

As suggested by the figure above, the paragraph vector is concatenated or averaged with local context word vectors to predict the next word. The prediction task changes the word vectors and the paragraph vector.

The paragraph vector can be further simplified when we use no local context in the prediction task. We can arrive at the following "Distributed Bag of Words" (PV-DBOW) model:

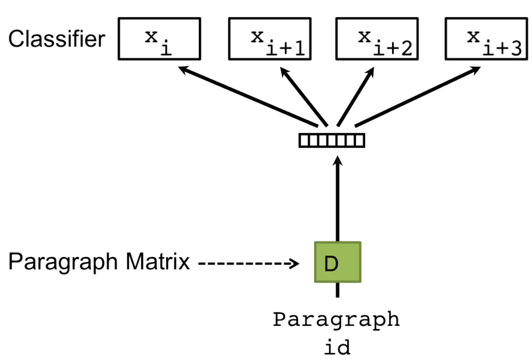

Figure 2: The distributed bag of words model of Paragraph Vector.

The original model PV-DBOW trains only the paragraph vectors but our model also trains word vectors to predict the adjacent words as in skip-gram word embeddings. This process helps to regularize the softmax classifier and hence usually produces better paragraph vectors. It also allows paragraph vectors and word vectors to be used together in vector operations. For document understanding purposes, the classifier parameters and word vectors are by-products of the training process and can be discarded.

The entire model is trained using stochastic gradient descent. At inference time, backpropagation is used to tune the paragraph vectors while holding the rest of the parameters constant.

As the distributed bag of words model is more efficient, the experiments in this paper focuses on this version of the Paragraph Vector. In the following sections, we will explore the use of Paragraph Vectors in different applications in document understanding.

3 Experiments

We conducted experiments with two different publicly available datasets: a dataset from a repository of electronic preprints (arXiv), and a dataset from the online encyclopaedia (Wikipedia).

In each case, all words were lower-cased before the datasets were used. We also jointly trained word embeddings with the paragraph vectors since preliminary experiments showed that this can improve the quality of the paragraph vectors. Preliminary results also showed that training with both unigrams and bigrams does not improve the quality of the final vectors so our experiments only use unigrams. We present a range of qualitative and quantitative results. We give some examples of nearest neighbours to some Wikipedia articles and arXiv papers as well as a visualisation of the space of Wikipedia articles. We also show some examples of nearest neighbours after performing vector operations.

For the quantitative evaluation, we attempt to measure how well paragraph vectors represent semantic similarity of related articles. We do this by constructing (both automatically and by hand) triplets, where each triplet consists of a pair of items that are close to each other and one item that is unrelated.

For the publicly available datasets we trained paragraph vectors over at least 10 epochs of the data and use a hierarchical softmax constructed as a Huffman tree as the classifier. We use cosine similarity as the metric. We also applied LDA with Gibbs sampling and 500 iterations with varying numbers of topics. For LDA, we set α to 0.1 and used values of β between 0.01 and 0.000001. We used the posterior topic proportions for each paper with Hellinger distance to compute the similarity between pairs of documents. For completeness, we also include the results of averaging the word embeddings for each word in a paper and using that as the paragraph vector. Finally, we consider the classical bag of words model where each word is represented as a one-hot vector weighted by TF-IDF and the document is represented by that vector, with comparisons done using cosine similarity.

3.1 Performance of Paragraph Vectors on Wikipedia entries

We extracted the main body text of 4,490,000 Wikipedia articles from the English site. There were an average of 360 words per article. We removed all links and applied a frequency cutoff to obtain a vocabulary of 915,715 words. We trained paragraph vectors on these Wikipedia articles and visualized them in Figure 3 using t-SNE [5]. The visualization confirms that articles having the same category are grouped together. There is a wide range of sport descriptions on wikipedia, which explains why the sports are less concentrated.

We also qualitatively look at the nearest neighbours of Wikipedia articles and compare Paragraph Vectors and LDA. For example, the nearest neighbours for the Wikipedia article "Machine learning" are shown in Table 1. We find that overall Paragraph Vectors have better nearest neighbours than LDA.

We can perform vector operations on paragraph vectors for local and non-local browsing of Wikipedia. In Table 2a and Table 2b, we show results of two experiments. The first experiment is to find related articles to "Lady Gaga." The second experiment is to find the Japanese equivalence of "Lady Gaga." This can be achieved by vector operations: pv("Lady Gaga") - wv("American") + wv("Japanese") where pv is paragraph vectors and wv is word vectors. Both sets of results show that Paragraph Vectors can achieve the same kind of analogies like Word Vectors [4] and that vector operations between the two types can be mixed. We also tried doing the same operations using just paragraph vectors for the corresponding Wikipedia pages but that didn't perform as well.

To quantitatively compare these methods, we constructed two datasets for triplet evaluation. The first consists of 172 triplets of articles we knew were related because of our domain knowledge. Some examples are: "Deep learning" is closer to "Machine learning" than "Computer network" or "Google" is closer to "Facebook" than "Walmart" etc. Some examples are hard and probably require

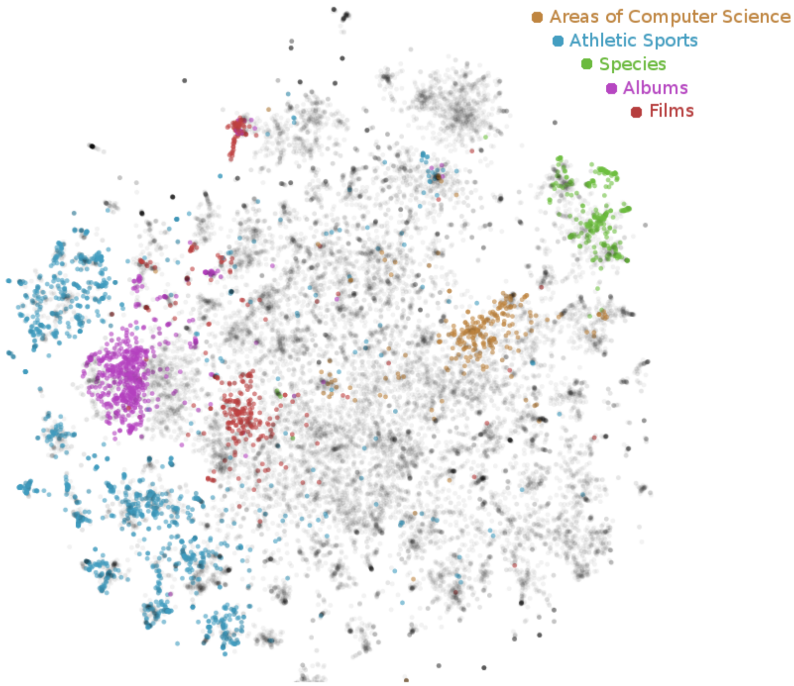

Figure 3: Visualization of Wikipedia paragraph vectors using t-SNE.

Table 1: Nearest neighbours to "Machine learning." Bold face texts are articles we found unrelated to "Machine learning." We use Hellinger distance for LDA and cosine distance for Paragraph Vectors as they work the best for each model.

LDA	**Paragraph Vectors**
Artificial neural network	Artificial neural network
Predictive analytics	Types of artificial neural networks
Structured prediction	Unsupervised learning
Mathematical geophysics	Feature learning
Supervised learning	Predictive analytics
Constrained conditional model	Pattern recognition
Sensitivity analysis	Statistical classification
SXML	Structured prediction
Feature scaling	Training set
Boosting (machine learning)	Meta learning (computer science)
Prior probability	Kernel method
Curse of dimensionality	Supervised learning
Scientific evidence	Generalization error
Online machine learning	Overfitting
N-gram	Multi-task learning
Cluster analysis	Generative model
Dimensionality reduction	Computational learning theory
Functional decomposition	Inductive bias
Bayesian network	Semi-supervised learning

some deep understanding of the content such as "San Diego" is closer to "Los Angeles" than "San Jose." This dataset will be released in the future.

The second dataset consists of 19,876 triplets in which two articles are closer because they are listed in the same category by Wikipedia, and the last article is not in the same category, but may be in a

Table 2: Wikipedia nearest neighbours

(a) Wikipedia nearest neighbours to "Lady Gaga" using Paragraph Vectors. All articles are relevant.

Article	Cosine Similarity
Christina Aguilera	0.674
Beyonce	0.645
Madonna (entertainer)	0.643
Artpop	0.640
Britney Spears	0.640
Cyndi Lauper	0.632
Rihanna	0.631
Pink (singer)	0.628
Born This Way	0.627
The Monster Ball Tour	0.620

(b) Wikipedia nearest neighbours to pv("Lady Gaga") - wv("American") + wv("Japanese") using Paragraph Vectors. Note that Ayumi Hamasaki is one of the most famous singers, and one of the best selling artists in Japan. She also has an album called "Poker Face" in 1998.

Article	Cosine Similarity
Ayumi Hamasaki	0.539
Shoko Nakagawa	0.531
Izumi Sakai	0.512
Urbangarde	0.505
Ringo Sheena	0.503
Toshiaki Kasuga	0.492
Chihiro Onitsuka	0.487
Namie Amuro	0.485
Yakuza (video game)	0.485
Nozomi Sasaki (model)	0.485

sibling category. For example, the articles for "Barack Obama" are closer to "Britney Spears" than "China." These triplets are generated randomly.

We will benchmark document embedding methods, such as LDA, bag of words, Paragraph Vector, to see how well these models capture the semantic of the documents. The results are reported in Table 3 and Table 4. For each of the methods, we also vary the number of embedding dimensions.

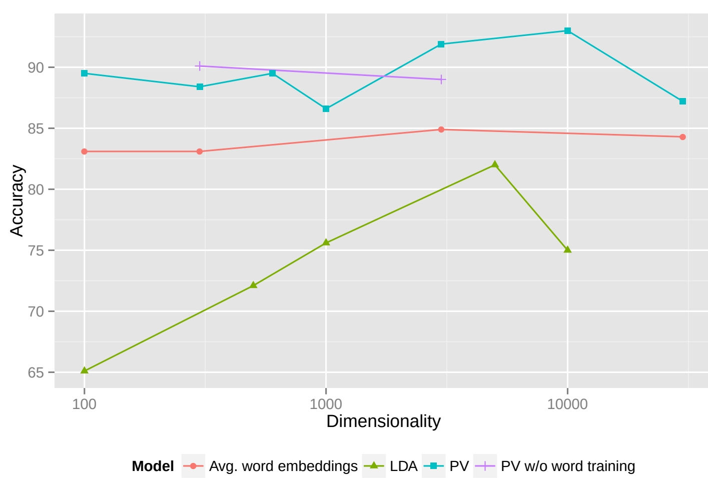

Figure 4: Results of experiments on the hand-built Wikipedia triplet dataset.

From the results in Table 3 and 4, it can be seen that paragraph vectors perform better than LDA. We also see a peak in paragraph vector performance at 10,000 dimensions. Paragraph Vectors are also less sensitive to differences in embedding dimensionality than LDA is to the number of topics. Both paragraph vectors and averaging word embeddings perform better than the LDA model. For LDA, we found that TF-IDF weighting of words and their inferred topic allocations did not affect the performance. From these results, we can also see that joint training of word vectors improves the final quality of the paragraph vectors.

Table 3: Performances of different methods on hand-built triplets of Wikipedia articles on the best performing dimensionality.

Model	Embedding dimensions/topics	Accuracy
Paragraph vectors	10000	93.0%
LDA	5000	82%
Averaged word embeddings	3000	84.9%
Bag of words		86.0%

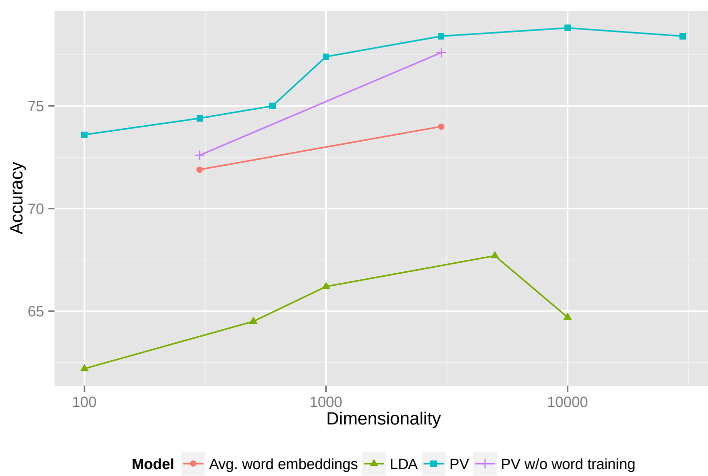

Figure 5: Results of experiments on the generated Wikipedia triplet dataset.

Table 4: Performances of different methods on dataset with generated Wikipedia triplets on the best performing dimensionality.

Model	Embedding dimensions/topics	Accuracy
Paragraph vectors	10000	78.8%
LDA	5000	67.7%
Averaged word embeddings	3000	74%
Bag of words		78.3%

3.2 Performance of Paragraph Vectors on arXiv papers

We extracted text from the PDF versions of 886,000 full arXiv papers. There were an average of 6,700 words per paper. In each case we only used the latest revision available. We applied a minimum frequency cutoff to the vocabulary so that our final vocabulary was 969,894 words.

We performed experiments to find related articles using Paragraph Vectors. In Table 5 and Table 6, we show the nearest neighbours of the original Paragraph Vector paper "Distributed Representations of Sentences and Documents" and the current paper. The table suggests that paragraph vectors can find better nearest neighbours in expert areas within a larger field such as machine learning within computer science whereas LDA in this case only finds only computer science papers. This is likely because LDA topics have been learnt to separate out physics fields, which is the predominant type of paper in arXiv.

In Table 7, we want to find the Bayesian equivalence of the Paragraph Vector paper. This can be achieved by vector operations: pv("Distributed Representations of Sentences and Documents") - wv("neural") + wv("Bayesian") where pv are paragraph vectors and wv are word vectors learned during the training of paragraph vectors. The results suggest that Paragraph Vector works well in these two tasks.

Table 5: Nearest neighbours to the machine learning paper "Distributed Representations of Sentences and Documents" in arXiv.

Title	**Similarity**
Paragraph vectors	
Efficient Estimation of Word Representations in Vector Space	0.8848
Polyglot: Distributed Word Representations for Multilingual NLP	0.8790
Evaluating Neural Word Representations in Tensor-Based Compositional Settings	0.8756
Lexicon Infused Phrase Embeddings for Named Entity Resolution	0.8648
Exploiting Similarities among Languages for Machine Translation	0.8635
Distributed Representations of Words and Phrases and their Compositionality	0.8635
Multilingual Distributed Representations without Word Alignment	0.8634
Learning Bilingual Word Representations by Marginalizing Alignments	0.8583
One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling	0.8555
Multilingual Models for Compositional Distributed Semantics	0.8539
LDA	
Living on the Edge: The Role of Proactive Caching in 5G Wireless Networks	0.9993
Performance Study of ETX based Wireless Routing Metrics	0.9993
Performance Comparison of Packet Scheduling Algorithms for Video Traffic in LTE Cellular Network	0.9993
Speech based Password Protected Cyber Applications	0.9993
Classification and Performance of AQM-Based Schemes for Congestion Avoidance	0.9992
The Effect of Communication and Synchronization on Amdahl Law in Multicore Systems	0.9992
Parallel Simulations for Analysing Portfolios of Catastrophic Event Risk	0.9992
PhishDef: URL Names Say It All	0.9992
An Agent-Based Modeling for Pandemic Influenza in Egypt	0.9991
Dynamic management of transactions in distributed real-time processing system	0.9991

Table 6: Nearest neighbours to the current paper in arXiv using Paragraph Vectors.

Title	**Cosine Similarity**
Distributed Representations of Sentences and Documents	0.681
Efficient Estimation of Word Representations in Vector Space	0.680
Thumbs up? Sentiment Classification using Machine Learning Techniques	0.642
Distributed Representations of Words and Phrases and their Compositionality	0.624
KNET: A General Framework for Learning Word Embedding using Morphological Knowledge	0.622
Japanese-Spanish Thesaurus Construction Using English as a Pivot	0.614
Multilingual Distributed Representations without Word Alignment	0.614
Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization	0.613
Exploiting Similarities among Languages for Machine Translation	0.612
A Survey on Information Retrieval, Text Categorization, and Web Crawling	0.610

Table 7: Nearest neighbours to "Distributed Representations of Sentences and Documents" - "neural" + "Bayesian". I.e., the Bayesian equivalence of the Paragraph Vector paper in arXiv.

Title	Cosine Similarity
Content Modeling Using Latent Permutations	0.629
SimLex-999: Evaluating Semantic Models With (Genuine) Similarity Estimation	0.611
Probabilistic Topic and Syntax Modeling with Part-of-Speech LDA	0.579
Evaluating Neural Word Representations in Tensor-Based Compositional Settings	0.572
Syntactic Topic Models	0.548
Training Restricted Boltzmann Machines on Word Observations	0.548
Discrete Component Analysis	0.547
Resolving Lexical Ambiguity in Tensor Regression Models of Meaning	0.546
Measuring political sentiment on Twitter: factor-optimal design for multinomial inverse regression	0.544
Scalable Probabilistic Entity-Topic Modeling	0.541

To measure the performance of different models on this task, we picked pairs of papers that had at least one shared subject, the unrelated paper was chosen at random from papers with no shared subjects with the first paper. We produced a dataset of 20,000 triplets by this method.

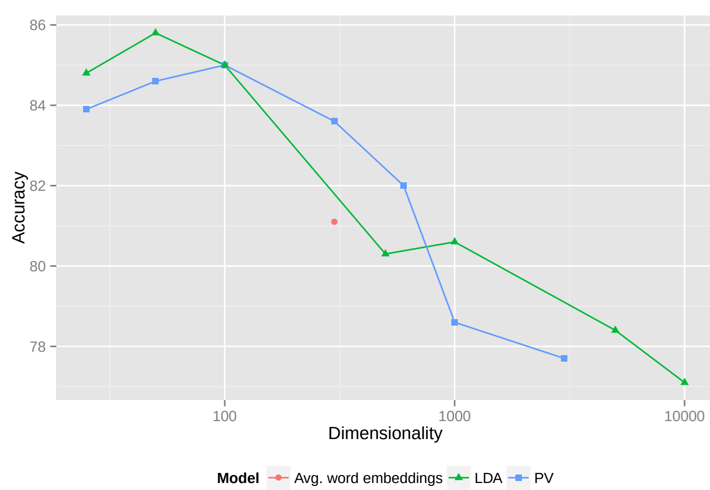

Figure 6: Results of experiments on the arXiv triplet dataset.

Table 8: Performances of different methods at the best dimensionality on the arXiv paper triplets.

Model	Embedding dimensions/topics	Accuracy
Paragraph vectors	100	85.0%
LDA	50	85.8%
Averaged word embeddings	300	81.1%
Bag of words		80.4%

From the results in Table 8, it can be seen that paragraph vectors perform less well at lower dimensions compared to LDA. We also see a peak in paragraph vector performance at 100 dimensions. Both models perform better than the vector space model. For LDA, we found that TF-IDF weighting of words and their inferred topic allocations did not affect performance.

4 Discussion

We described a new set of results on Paragraph Vectors showing they can effectively be used for measuring semantic similarity between long pieces of texts. Our experiments show that Paragraph Vectors are superior to LDA for measuring semantic similarity on Wikipedia articles across all sizes of Paragraph Vectors. Paragraph Vectors seem to perform less well in cases when a low dimensional embedding is better for representing a corpus, however they still yield good nearest neighbours. Also surprisingly, vector operations can be performed on them similarly to word vectors and additionally paragraph vectors and word vectors can be mixed in the same operation. This can provide interesting new techniques for a wide range of applications: local and nonlocal corpus navigation, dataset exploration, book recommendation and reviewer allocation.

References

[1] D. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 2003.

[2] Z. Harris. Distributional structure. *Word*, 1954.

[3] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, 2014.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[5] L. J. P. van der Maaten and G. E. Hinton. Visualizing high-dimensional data using t-SNE. *Journal of Machine Learning Research*, 2008.