Research Paper Summary/Notes

Distributed Representations of Sentences and Documents, Google.

Laxman Singh Tomar

October 2019

Research Paper Summary/Notes

Distributed Representations of Sentences and Documents, Google.

Laxman Singh Tomar

October 2019

Contents

Dedication

I would like to dedicate this work to teachers whom I've never met in real but their valuable contribution have made this work possible.

Acknowledgement

(I encountered several resources which helped me in getting through this work)

I would like to express my sincere gratitude to:

- Tomas Mikolov et. al.
- Adrian Colyer

Abstract

This paper introduces "Paragraph Vectors" inspired from word vectors. Fixed-length feature vector techniques like Bag-of-Words are unable to capture semantics of words as they lose ordering of words. Our algorithms outperform bag-of-words over text representation tasks and achieves state-of-the-art results.

Keywords: Paragraph Vectors; PV-DM; PV-DBOW

Chapter 1

Introduction

Common Machine Learning Algorithms like Clustering and Classification typically require the text input to be represented as a fixed length vector. Common models for this are bag-of-words and bag-of-n-grams. Bag of words loses any meaning that might come from ordering of words. Intuitively you can sense that word order is highly likely to contain useful information! Bag-of-n-grams only considers short contexts and suffers from high dimensionality and data sparsity.

Researchers previously tried combining distributed word vectors-for example by using a weighted average of all words in a document, or combining word vectors in an order given by the parse tree of a sentence [1]. The first of these approaches also suffers from loss of word order information, the latter cannot easily be extended beyond sentences.

Our algorithm represents each documents by a dense vector which is trained to predict words in the document. Empirical results show that Paragraph-Vectors outperform bag-of-words model as well as other techniques for text representations. Finally, we achieve new state-of-the-art results on several text classification and sentiment analysis tasks.

1.1 Motivation

Word vectors can predict the next word in the sentence even when they are initialized randomly. They can eventually capture semantics as an indirect result of the prediction task. Why not to use the same for paragraphs?

1.2 Objectives

Paragraph vectors are asked to contribute to the prediction task of the next word given many contexts sampled from the paragraph. In order to do so, few architectures are required to be developed.

Chapter 2

Methods

2.1 Learning Vector Representations of Words

This section introduces the concept of distributed representation of words aka word vectors [2]. The task is to predict a word given the other words in a context. Let' say "the cat sat on ∼" is a sentence. Words are mapped to unique vectors. Given a sequence of training words: $w_1, w_2, \ldots, w_T$, the objective of the word-vector model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k} \log p(w_t|w_{t-k}, ..., w_{t+k})$$

The prediction task is typically done via a multiclass classifier, such as softmax:

$$p(w_t|w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{wt}}}{\sum_i e^{y_i}}$$

Each y_i in the formulation above is un-normalized log-probability for each output word i, computed as:

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W)$$

where U and b are the softmax parameters; h is constructed by a concatenation or average of word vector extracted from W.

Later in practice, we found that Hierarchical Softmax works better than softmax. It's structure is binary Huffman tree which helps in accessing frequent words. Also, neural network based word models are trained using Stochastic Gradient Descent and Back-propagation and henceforth referred as Neural Language Models [3].

Once training is finished, words with similar meanings are mapped to a similar position in contrast to those who are not similar in their meanings. Say, "Powerful" and "Strong" are close while "Powerful" and "Paris" are more distant. Also word vectors are popular for drawing out useful analogies.

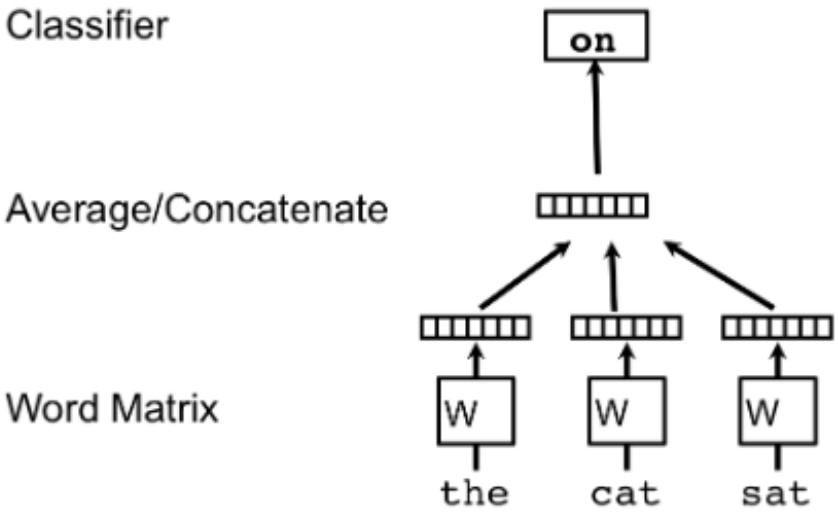

Figure 1: A framework for learning word vectors. Context of three words ("the," "cat," and "sat") is used to predict the fourth word ("on"). The input words are mapped to columns of the matrix W to predict the output word.

2.2 Paragraph Vectors: A Distributed Memory Model

Words are still mapped to unique vectors as before. In addition every paragraph (or document, if working at the document level) is also mapped to a unique vector. Word vectors are captured as columns in the matrix W, and paragraph vectors as columns in the matrix D.

The only change compared to word vector learning is that the paragraph vector is concatenated with the word vectors to predict the next word in a context. Contexts

are of fixed length and sampling from a sliding window over a paragraph. Paragraph vectors are shared for all windows generated from the same paragraph, but not across paragraphs. In contrast, word vectors are shared across all paragraphs i.e vector for "Powerful" is same across all the paragraphs.

The paragraph vectors can be thought of as another vector which represents the missing information from the current context and can act as a memory of it-the topic of the paragraph. For this reason, we often call this model the Distributed Memory Model of Paragraph Vectors (PV-DM).

At every step in training a fixed-length context is sampled from a random paragraph and used to compute the error gradient in order to update the parameters in the model. With N paragraphs each mapped to p dimensions and M words each mapped to q dimensions the model has a total of $N \times p + M \times q$ parameters (excluding the softmax parameters). Once they have been trained, the paragraph vectors can be used as features for the paragraph in any follow-on machine learning task. At prediction time, an inference step is performed to compute the paragraph vector for a new (never seen before) paragraph. During this step the parameters for the rest of the model (word vectors W and softmax weights U and b) are fixed.

In summary, the algorithm itself has two key stages:

1) Training to get word vectors W, softmax weights U, b and paragraph vectors D on already seen paragraphs.

2) "The Inference Stage" to get paragraph vectors D for new paragraphs (never seen before) by adding more columns in D and gradient descending on D while holding W, U, b fixed. We use D to make a prediction about some particular labels using a standard classifier, e.g. Logistic Regression.

2.2.1 Advantages of Paragraph Vectors

1. They are learned from unlabeled data thus can work well for tasks that don't have enough labeled data.

2. They address some of the key weakness of bag-of-words model:

a. They inherit an important property of word vectors-semantics. Here, "Powerful" is closer to "Strong" than "Paris".

b. They take ordering of words into consideration.

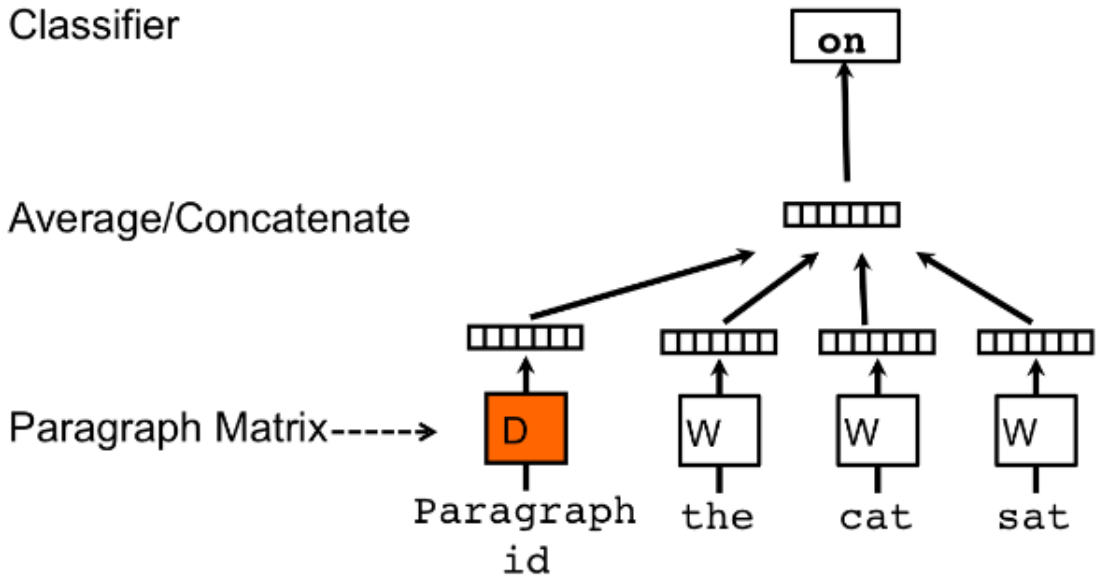

Figure 2: A framework for learning paragraph vector. This frame-work is similar to the framework presented in Figure 1; the only change is the additional paragraph token that is mapped to a vector via matrix D. In this model, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph.

2.3 Paragraph Vector without Word Ordering:Distributed Bag of Words

This is variation to the above scheme. Here we are ignoring context words in the input (i.e., do away with the sliding window), and instead force the model to predict words randomly sampled from the paragraph in the output.

At each iteration of Stochastic Gradient Descent we sample a text window, then sample a random word from the text window and form a classification task given

the Paragraph Vector. In this version, we trained the paragraph vector to predict the words in a small window. We name this version the Distributed Bag of Words version of Paragraph Vector (PV-DBOW), as opposed to Distributed Memory version of Paragraph Vector (PV-DM) in the previous section.

In our experiments, each paragraph vector is a combination of two vectors: one learned by the standard PV-DM and one learned by the paragraph vector with PV-DBOW. PV-DM alone usually works well for most tasks(with state-of-art performances), but its combination with PV-DBOW is usually more consistent across many tasks that we try and therefore strongly recommended.

Chapter 3

Results

3.1 Experiments

We performed experiments to better understand the behavior of the paragraph vectors. To achieve this, we benchmark Paragraph Vector on two text understanding problems that require fixed-length vector representations of paragraphs:sentiment analysis and information retrieval.

Sentiment Analysis was done over two datasets: Stanford sentiment treebank database in which each example was having a single sentence and IMDB database in which each example had several sentences. We also tested our method on an information retrieval task, where the goal is to decide if a document should be retrieved given a query.

3.2 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

The Stanford Sentiment Treebank dataset contains 11855 sentences labeled from very negative to very positive on a scale from 0.0 to 1.0. Using a window size of 8, and a vector which is a concatenation of one from PV-DBOW and one from PV-DM (both of 400 dimensions), we achieved the following results:

Table 1: The performance of our method compared to other approaches on the Stanford Sentiment Treebank dataset. The error rates of other methods are reported in (Socher et al., 2013b).

Model	**Error rate (Positive/Negative)**	**Error rate (Fine-grained)**
Naive Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naive Bayes (Socher et al., 2013b)	16.9%	58.1%
Word Vector Averaging (Socher et al., 2013b)	19.9%	67.3%
Recursive Neural Network (Socher et al., 2013b)	17.6%	56.8%
Matrix Vector-RNN (Socher et al., 2013b)	17.1%	55.6%
Recursive Neural Tensor Network (Socher et al., 2013b)	14.6%	54.3%
Paragraph Vector	**12.2%**	**51.3%**

Distributed Representations of Sentences and Documents Tasks and Baselines: In (Socher et al., 2013b), the authors propose two ways of benchmarking. First, one could consider a 5-way fine-grained classification task where the labels areVery Negative, Negative, Neutral, Positive, Very Positive or a 2-way coarse-grained classification task where the labels areNegative, Positive. The other axis of variation is in terms of whether we should label the entire sentence or all phrases in the sentence. In this work we only consider labeling the full sentences.

Despite the fact that it does not require parsing, paragraph vectors perform better than all the baselines with an absolute improvement of 2.4% (relative improvement 16%) compared to the next best results.

3.3 Beyond One Sentence: Sentiment Analysis with IMDB dataset

Of course paragraph vectors are not restricted to just sentences, thus we also applied the technique to a dataset of 100,000 movie reviews taken from IMDB. One key

aspect of this dataset is that each movie review has several sentences.

The 100,000 movie reviews are divided into three datasets:25,000 labeled training instances, 25,000 labeled test instances and 50,000 unlabeled training instances. There are two types of labels: Positive and Negative. These labels are balanced in both the training and the test set.

We learn the word vectors and paragraph vectors using 75,000 training documents (25,000 labeled and 50,000 unlabeled instances). The paragraph vectors for the 25,000 labeled instances are then fed through a neural network with one hidden layer with 50 units and a logistic classifier to learn to predict the sentiment. At test time, given a test sentence, we again freeze the rest of the network and learn the paragraph vectors for the test reviews by gradient descent. Once the vectors are learned, we feed them through the neural network to predict the sentiment of the reviews.

Once again, paragraph vectors outperforms the prior state of the art, with a 15% relative improvement:

Table 2: The performance of Paragraph Vector compared to othe rapproaches on the IMDB dataset. The error rates of other methods are reported in (Wang Manning, 2012).

Model	**Error rate**
BoW (bnc) (Maas et al., 2011)	12.20%
BoW (bt'c) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM + BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	**7.42%**

3.4 Information Retrieval with Paragraph Vectors

We turn our attention to an information retrieval task which requires fixed-length representations of paragraphs. We looked at the top 10 results of each of the 1 million most popular queries on a search engine, and extracted paragraphs from them. They create sets of three paragraphs: two drawn from the results of the same query, and one from another query.

Our goal is to identify which of the three paragraphs are results of the same query. To achieve this, we will use paragraph vectors and compute the distances between the paragraphs. A better representation is one that achieves a small distance for pairs of paragraphs of the same query, and a large distance for pairs of paragraphs of different queries. Thus this experiment is a way of assessing whether paragraph vectors in some way capture the meaning of paragraphs as word vectors do.

The results show that Paragraph Vector works well and gives a 32% relative improvement in terms of error rate. The fact that the paragraph vector method significantly outperforms bag of words and bigrams suggests that our proposed method is useful for capturing the semantics of the input text.

Table 3: The performance of Paragraph Vector and bag-of-words models on the information retrieval task. "Weighted Bag-of-bigrams" is the method where we learn a linear matrix W on TF-IDF bigram features that maximizes the distance between the first and the third paragraph and minimizes the distance between the first and the second paragraph.

Model	**Error rate**
Vector Averaging	10.25%
Bag-of-words	8.10 %
Bag-of-bigrams	7.28 %
Weighted Bag-of-bigrams	5.67%
Paragraph Vector	3.82%

3.5 Some Further Observations

1. PV-DM is consistently better than PV-DBOW. PV-DM alone can achieve results close to many results in this paper (see Table 2). For example, in IMDB, PV-DM

only achieves 7.63%. The combination of PV-DM and PV-DBOW often works consistently better (7.42% in IMDB) and therefore recommended.

2. Using concatenation in PV-DM is often better than sum. In IMDB, PV-DM with sum can only achieve 8.06%. Perhaps, this is because the model loses the ordering information.

3. It's better to cross validate the window size. A good guess of window size in many applications is between 5 and 12. In IMDB, varying the window sizes between 5 and 12 causes the error rate to fluctuate 0.7%.

4. Paragraph Vector can be expensive, but it can be done in parallel at test time. On average, our implementation takes 30 minutes to compute the paragraph vectors of the IMDB test set, using a 16 core machine(25,000 documents, each document on average has 230 words).

Chapter 4

Discussion

4.1 Discussion

We saw how Paragraph Vector; an unsupervised learning algorithm learns vectors representations for variable length pieces of text such as sentences and documents. The vector representations are learned to predict the surrounding words in contexts sampled from the paragraph.

We saw how Paragraph Vector outperformed all other techniques over several text classification tasks implying their potential to capture semantics of the paragraphs.

It also overcome many weaknesses of bag-of-words models.

4.2 Conclusions

Paragraph Vector can be used to learn vectors of sentences and paragraphs. They are quite useful for non-text domains where parsing isn't available and would prove to be a strong alternative to bag-of-words models.

List of Figures

List of Tables

Bibliography

[1] Socher, L. C. C. N. A., Richard & Manning, C. Parsing natural scenes and natural language with recursive neural networks. *ICML-11, pp. 129-136* (2011).

[2] Tomas Mikolov, G. C., Kai Chen & Dean, J. Efficient estimation of word representations in vector space. *ICLR* (2013).

[3] Bengio, S. H. S. J. S. M. F., Yoshua & Gauvain, J.-L. Neural probabilistic language models. *Springer pp. 137-186* (2006).

Appendix A

Datasets

A.1 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

The dataset can be downloaded at:

`http://nlp.Stanford.edu/sentiment/`

A.2 Beyond One Sentence: Sentiment Analysis with IMDB dataset

The dataset can be downloaded at:

`http://ai.Stanford.edu/amaas/data/sentiment/index.html`

Appendix B

Doc2Vec Implementation in Pytorch

This is a pytorch implementation of Doc2Vec:

`https://github.com/inejc/paragraph-vectors`