Distributed Representations of Sentences and Documents

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean

KISTI-UST
JUYEON YU

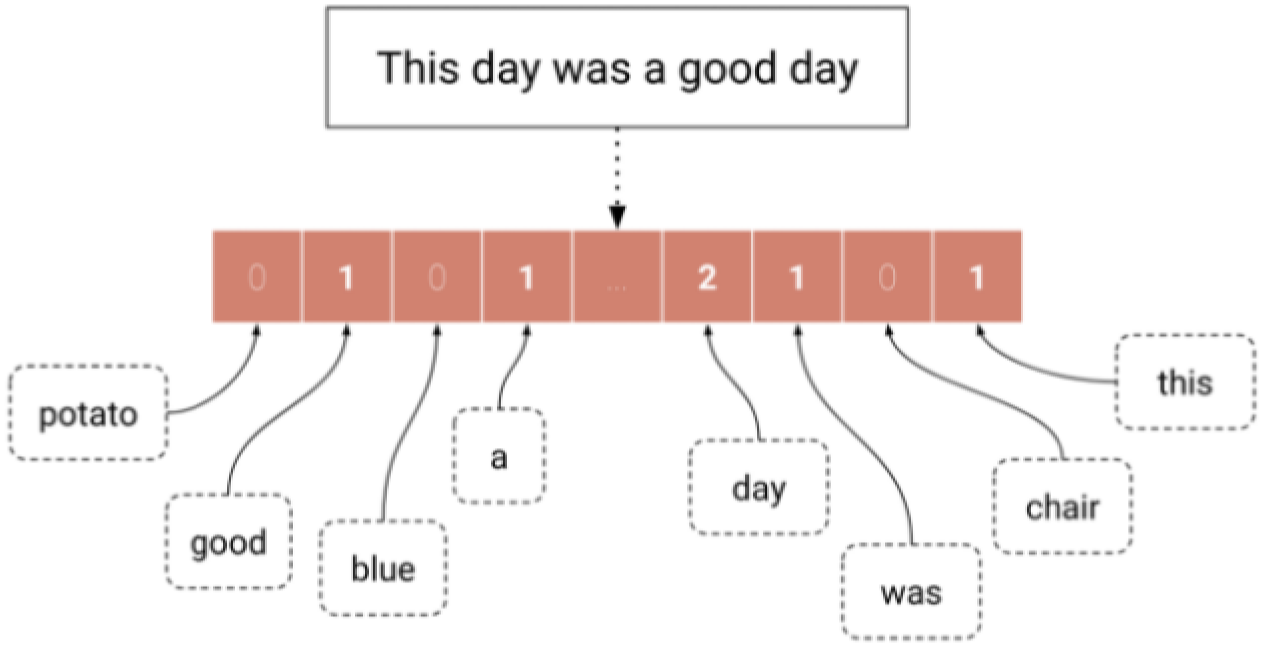
2021.03.26.FRI

Contents

1. Introduction Abstract

Bag of words weakeness

- lose the ordering of the words
- ignore semantics of the words
- e.g) “powerful,” “strong”and “Paris” are equally distant.



Example of a **Bag-Of-Words** representation

Paragraph Vector

- unsupervised algorithm
- learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents.
- this algorithm represents each document by a dense vector which is trained to predict words in the document.

  Its construction gives our algorithm the potential to overcome the weaknesses of bag-ofwords models.
- Empirical results show that Paragraph Vectors outperform bag-of-words models as well as other techniques for text representations.

1. Introduction

- In this paper, we propose Paragraph Vector, an unsupervised algorithm that learns fixed-length feature representations from variable-length pieces of texts, such as sentences, paragraphs, and documents.

- Our algorithm represents each document by a dense vector which is trained to predict words in the document.

- Empirical results show that Paragraph Vectors outperform bag-of-words models 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.

2. Algorithms

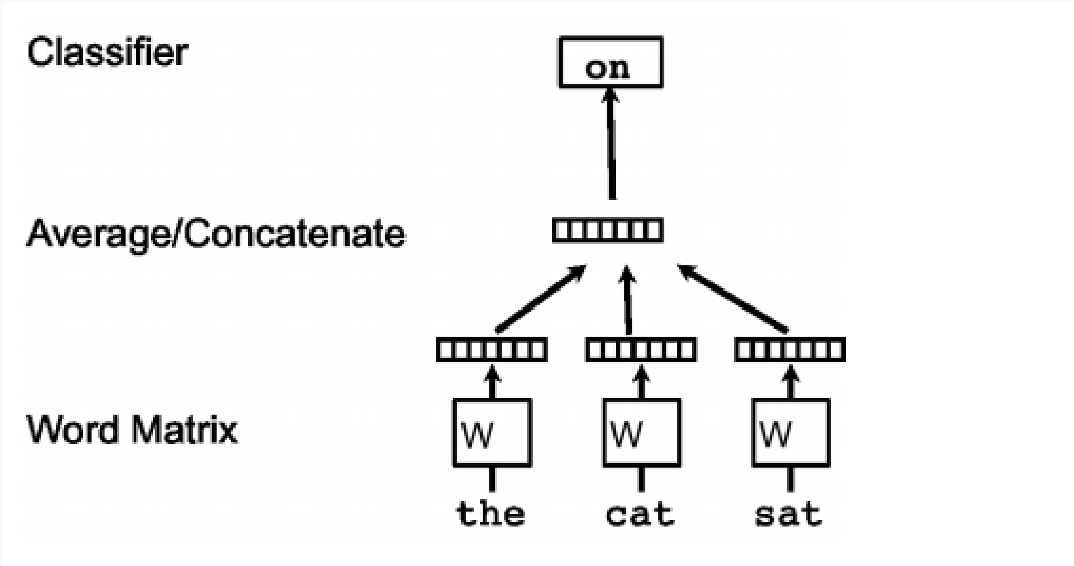
2.1 Learning Vector Representation of Words

- The task is to predict a word given the other words in a context.

- Given a sequence of training words w1,w2,w3, ⋯,wT , the objective of the word vector model is to maximize the average log probability:

- where U , b are the softmax parameters.

- h is constructed by a concatenation or average of word vectors extracted from W



<Table 1>

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

figure 1

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

figure 2

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

figure 3

2. Algorithms

2.1 Learning Vector Representation of Words

figure 1

- Learned in the process of maximizing the mean of log probabilities
  - The increase in the value of figure 1 means that if you enter the previous k words into the model, the model matches the next word (target) well.
- T: Number of words per sentence of training data

figure 2

- w_t: t-th word of the sentence
- y_i: Score corresponding to the i-th word in the entire vocabulary set of the corpus
- h: A function that takes an average or concatenates it when a vector sequence is given and returns a vector of fixed length

figure 3

- After referring to the previous k words in the word matrix W the average was taken or concatenated.
- A matrix called U is added and a bias vector b is added.
- It becomes a probability vector after taking a softmax on y created in this way.
- The size of U is the vocabulary set size X the number of embedding dimensions.

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

figure 1

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

figure 2

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

figure 3

2. Algorithms

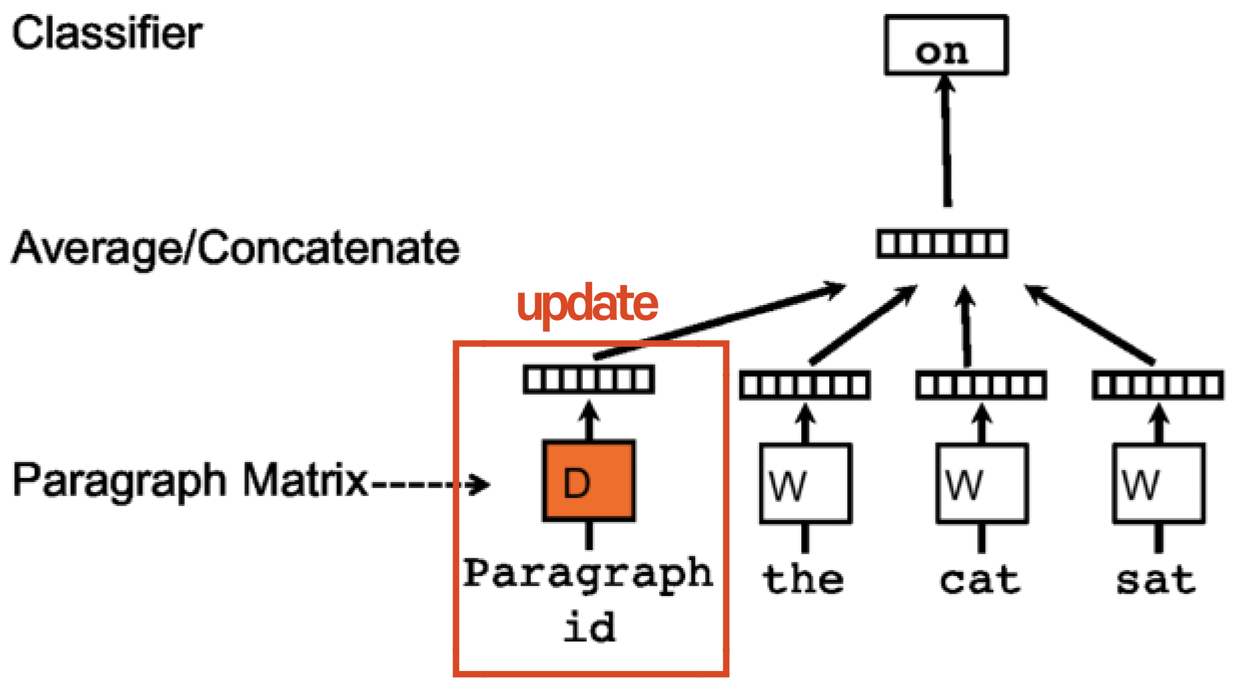
2.2 Paragraph Vector : A distributed memory model

PV-DM (Paragraph Vector with Distributed Memory)

- Assign and randomly initialize paragraph vector for each doc
- Predict next word using paragraph vector + context word
- Slide context window across doc but keep paragraph vec fixed

(hence distributed memory)

- Updating done via Stochastic Gradient Descent and backpropagation.



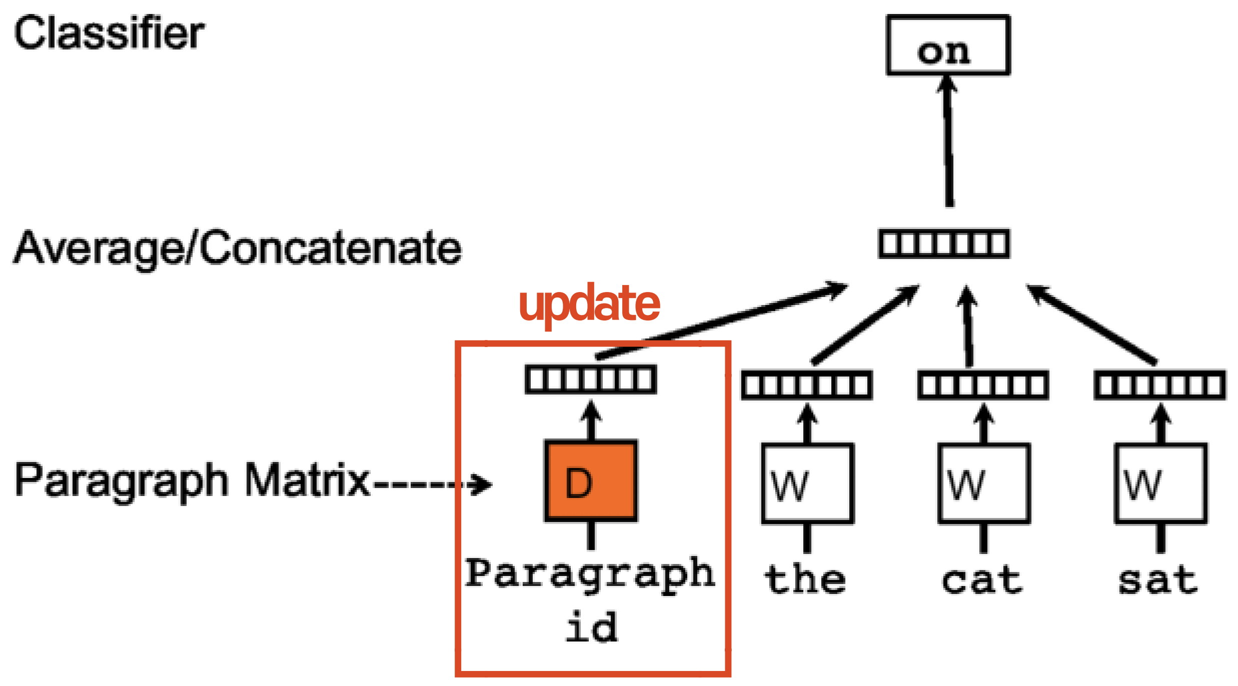
<Table 2> Doc2Vec PV-DM

PV-DM (Paragraph Vector with Distributed Memory)

- In our Paragraph Vector framework, every paragraph is mapped to a unique vector, represented by a column in matrix D and every word is also mapped to a unique vector, represented by a column in matrix W.

- 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.



<Table 2> Doc2Vec PV-DM

2. Algorithms

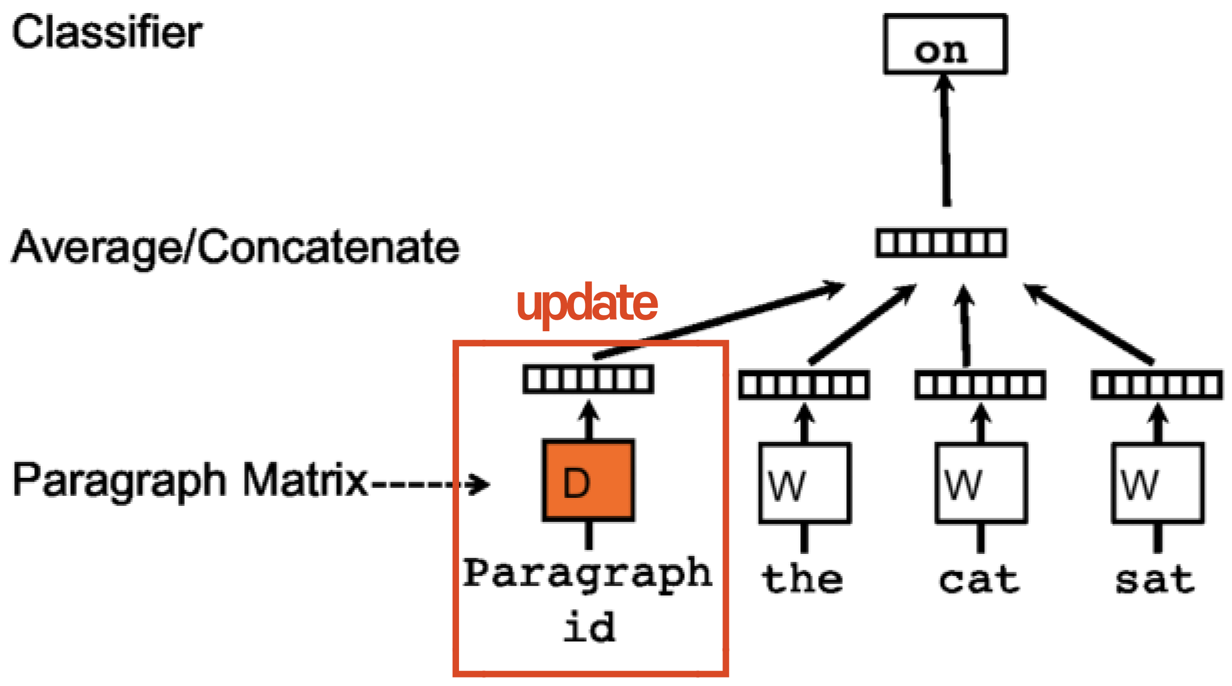
2.2 Paragraph Vector : A distributed memory model

PV-DM (Paragraph Vector with Distributed Memory)

- 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 stand ard classifier.



<Table 2> Doc2Vec PV-DM

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

Advantages of paragraph vectors

- An important advantage of paragraph vectors is that they are learned from unlabeled data and thus can work well for tasks that do not have enough labeled data.

- Paragraph vectors also address some of the key weaknesses of bag-of-words models.
  - First, they inherit an important property of the word vectors: the semantics of the words. In this space, "powerful" is closer to "strong" than to "Paris."
  - The second advantage of the paragraph vectors is that they take into consideration the word order.

PV-DM

"나는 배가 고파서 밥을 먹었다"

Paragraph Dictionary

ID	Paragraph
1	나는 배가 고파서 밥을 먹었다

Word Dictionary

ID	Word
1	나는
2	배가
3	고파서
4	밥을
5	먹었다

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

PV-DM

“나는 배가 고파서 밥을 먹었다”

Paragraph Embedding

ID	Paragraph Embedding
1	[0.5, 0.41, 0.55 ...]

Word Embedding

ID	Word Embedding
1	[0.2, 0.11, 0.55 ...]
2	[0.9, 0.41, 0.75 ...]
3	[0.4, 0.15, 0.53 ...]
4	[0.3, 0.78, 0.48 ...]
5	[0.6, 0.23, 0.12 ...]

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

PV-DM

“나는 배가 고파서 밥을 먹었다”

Step	Input	Label
1	[나는 배가 고파서 밥을 먹었다, 나는, 배가, 고파서]	밥을
2	[나는 배가 고파서 밥을 먹었다, 배가, 고파서, 밥을]	먹었다

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

PV-DM

“나는 배가 고파서 밥을 먹었다”

Step	Input	Label
1	[d1, w1, w2, w3]	w4
2	[d1, w2, w3, w4]	w5

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

PV-DM

“나는 배가 고파서 밥을 먹었다”

Step	Input	Label
1	[[0.5, 0.41, 0.55 ...], [0.2, 0.11, 0.55 ...], [0.9, 0.41, 0.75 ...], [0.4, 0.15, 0.53 ...]]	[0,0,0,0,1,0]
2	[[0.5, 0.41, 0.55 ...], [0.9, 0.41, 0.75 ...], [0.4, 0.15, 0.53 ...], [0.3, 0.78, 0.48 ...]]	[0,0,0,0,0,1]

PV-DBOW (Paragraph Vector with Distributed Bag of Words)

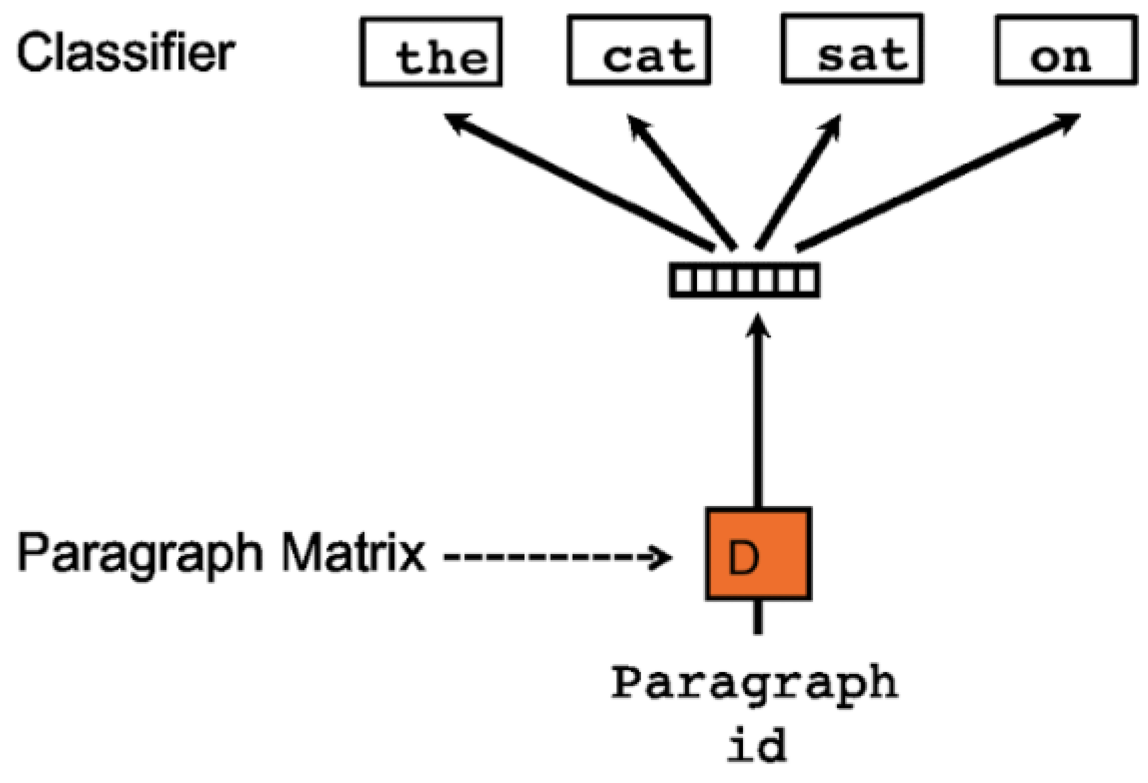
- Only use paragraph vectors(no word vectors)

- Another way is to ignore the context words in the input, but f orce 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 wind ow and form a classification task given the Paragraph Vector.

- In this version, the paragraph vector is trained to predict the words in a small window.

- Simpler, more memory effcient



<Table 3> Doc2Vec PV-DBOW

2. Algorithms

2.3 Paragraph Vector : A distributed memory model

PV-DBOW

"나는 배가 고파서 밥을 먹었다"

Step	Input	Label
1	나는 배가 고파서 밥을 먹었다	나는
2	나는 배가 고파서 밥을 먹었다	배가
3	나는 배가 고파서 밥을 먹었다	고파서
4	나는 배가 고파서 밥을 먹었다	밥을
5	나는 배가 고파서 밥을 먹었다	먹었다

3. Experiments

Sentiment analysis

- Stanford sentiment treebank dataset (Socher et al., 2013b)

  -single sentence

- IMDB dataset (Maaset et al., 2011)

  -several sentences

Information retrieval

3. Experiments 3.1 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

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.

• The dataset consists of three sets: 8544 sentences for training, 2210 sentences for test and 1101 sentences for validation (or development).

3. Experiments 3.1 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

Tasks and Baselines

- 5-way fine-grained classification task

  {Very Negative, Negative, Neutral, Positive, Very Positive}

- 2-way coarse-grained classification task

  {Negative, Positive}

- In this work we only consider labeling the full sentences.

- Recursive Neural Tensor Network works much better than bag-of-words model.

  - movie reviews are often short.

  - compositionality plas an importtant role in deciding whether the review is positive or negative.

3. Experiments 3.1 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

Experimental protocols

- To make use of the available labeled data, in our model, each subphrase is treated as an independent sentence and we learn the representations for all the subphrases in the training set.

- After learning the vector representations for training sentences and their subphrases, we feed them to a logistic regression to learn a predictor of the movie rating

- The optimal window size is 8.

- The vector presented to the classifier is a concatenation of PV-DBOW and PV-DM.

3. Experiments 3.1 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

Results

• Simply averaging the word vectors (in a bagof-words fashion) does not improve the results.

- NB, SVM, BiNB
- bag-of-words models do not consider word order.
- therefore fail to recognize many sophisticated linguistic phenomena (sarcasm).

• 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.

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)
Naïve Bayes (Socher et al., 2013b)	18.2 %	59.0%
SVMs (Socher et al., 2013b)	20.6%	59.3%
Bigram Naïve 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%**

3. Experiments 3.2 Beyond One Sentence: Sentiment Analysis with IMDB Dataset

Dataset

- The dataset consists of 100,000 movie reviews taken from IMDB
- Each movie review has several sentences
- The 100,000 movie reviews are divided into three datasets:
  - 5,000 labeled training instances, 25,000 labeled test instances and 50,000 unlabeled training instances
- There are two types of labels: Positive and Negative

3. Experiments 3.2 Beyond One Sentence: Sentiment Analysis with IMDB Dataset

Experimental protocols

- 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 c
lassifier to learn to predict the sentiment.

- The optimal window size is 10.

- The vector presented to the classifier is a concatenation of PV-DBOW and PV-DM.

3. Experiments 3.2 Beyond One Sentence: Sentiment Analysis with IMDB Dataset

Results

- Simply using the word vectors (in a bagof-words fashion) does not improve the results.

- The combination of two models (Restricted Boltzmann Machines model + bag-of-words.) yields an improvement approximately 1.5% in terms of error rates.

- NBSVM on bigram features works the best

- paragraph vectors outperform the prior state of the art, with a 15% relative improvement

Table 2. The performance of Paragraph Vector compared to other approaches 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 (bΔt'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. Experiments 3.3 Information Retrieval with Paragraph Vectors

Dataset

- For their third experiment, the authors 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.

Paragraph 1:
calls from (000) 000 - 0000. 3913
calls reported from this number. according to 4 reports the identity of this caller is american airlines

Paragraph 2:
do you want to find out who called you from +1 000 - 000 - 0000 , +1 0000000000 or (000) 000 - 0000 ?
see reports and share information you
have about this caller

Paragraph 3:
allina health clinic patients for your convenience , you can pay your allina health clinic bill online . pay your
clinic bill now , question and answers...

3. Experiments 3.3 Information Retrieval with Paragraph Vectors

Experimental protocols

- The triplets are split into three sets: 80% for training, 10% for validation, and 10% for testing.

- We benchmark four methods to compute features for paragraphs

  - bag-of-words, bag-of-bigrams, averaging word vectors and Paragraph Vector.

3. Experiments 3.3 Information Retrieval with Paragraph Vectors

Results

- 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 semant ics 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. Experiments 3.4 Some further observations

- PV-DM is consistently better than PV-DBOW. PVDM 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 work consistently better (7.42% in IMDB) and therefore recommended.

- 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.

- 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%.

- 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).

4. Related Work

- Distributed representations for words were first proposed (Rumelhart et al., 1986)

- Statistical language modeling (Elman, 1990; Bengio et al., 2006; Mikolov, 2012)

- Word vectors have been used in NLP applications
  - such as word epresentation, named entity recognition, word sense disambiguation, parsing, tagging and machine translation
  (Collobert & Weston, 2008; Turney & Pantel, 2010; Turian et al., 2010; Collobert et al., 2011; Socher et al., 2011b; Huang et al., 2012; Zou et al., 2013)

- Representing phrases (Mitchell & Lapata, 2010; Zanzotto et al., 2010; Yessenalina & Cardie, 2011; Grefenstette et al., 2013; Mikolov et al., 2013c)

- autoencoder-style models(Maas et al., 2011; Larochelle & Lauly, 2012; Srivastava et al., 2013)

- Distributed representations of phrases and sentences (Socher et al., 2011a;c; 2013b)

- Our approach of computing the paragraph vectors via gradient descent bears resemblance to a successful paradigm in computer vision (Perronnin & Dance, 2007; Perronnin et al., 2010) known as Fisher kernels (Jaakkola & Haussler, 1999).

5. Discussion

- We described Paragraph Vector, an unsupervised learning algorithm that learns vector representations for variable length pieces of texts such as sentences and documents.

- The vector representations are learned to predict the surrounding words in contexts sampled from the paragraph.

- Our experiments on several text classification tasks such as Stanford Treebank and IMDB sentiment analysis datasets show that the method is competitive with state-of-the-art methods. The good performance demonstrates the merits of Paragraph Vector in capturing the semantics of paragraphs.

- In fact, paragraph vectors have the potential to overcome many weaknesses of bag-of-words models.

THANK YOU

Q & A