UST UNIVERSITY OF SCIENCE & TECHNOLOGY

KiSTi Korea Institute of Science and Technology Information www.kisti.re.kr

Distributed Representations of Sentences and Documents

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

KISTI-UST
JUYEON YU

2021.03.26.FRI

목차

1. Introduction

2. Algorithms

- Learning Vector Representation of Words
- Paragraph Vector: A distrivuted memory model
- Paragraph Vector without word ordering: Distributed bag of words

3. Experiments

- Sentiment Analysis (with Stanford dataset)
- Sentiment Analysis (with IMDB dataset)
- Information Retrieval with Paragraph Vectors

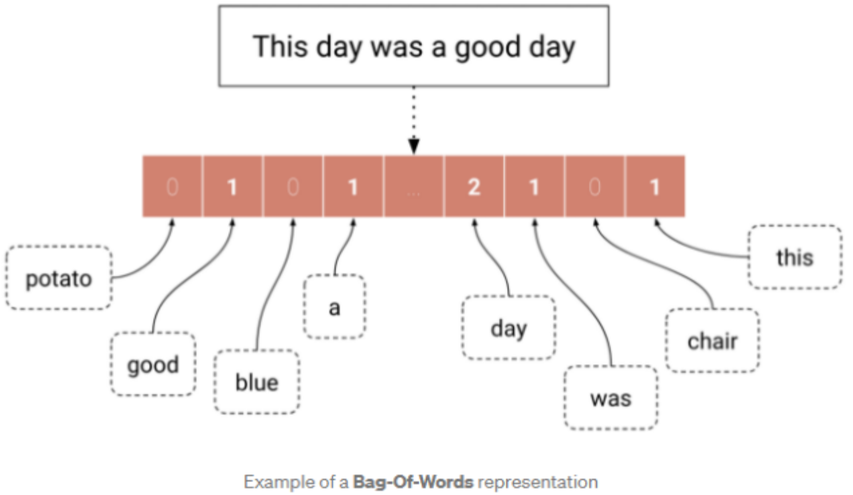
4. Related Work

5. Discussion

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 fro m 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 repres entations.

- 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

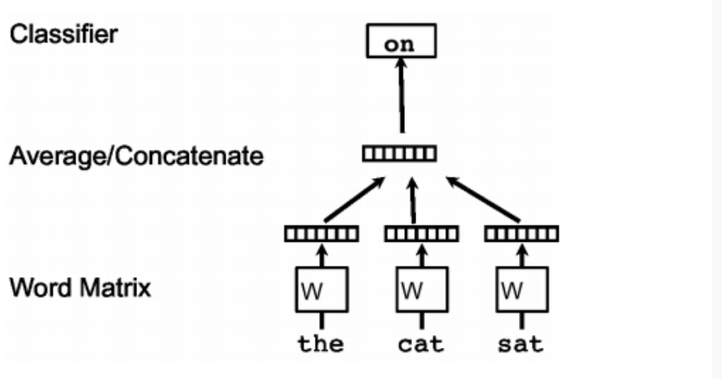


그림 1. 언어 모델 기반 단어 벡터 학습

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

수식 1

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

수식 2

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

수식 3

2. Algorithms

2.1 Learning Vector Representation of Words

수식 1

- 로그 확률의 평균을 최대화하는 과정에서 학습 됨
  - 수식 1의 값이 커진 다는 건 모델에 이전 k개 단어들을 입력하면 모델이 다음 단어(target)을 잘 맞춘다는 뜻
- T: 학습 데이터 문장 하나의 단어 개수

수식 2

- w_t: 문장의 t 번째 단어
- y_i: 말뭉치 전체 어휘 집합 가운데 i번째 단어에 해당하는 점수(score)
- h: 벡터 시퀀스가 주어졌을 때 평균을 취하거나 이어 붙여 고정된 길이의 벡터 하나를 반환하는 역할을 하는 함수

수식 3

- 이전 k 개 단어들을 W라는 단어 행렬에서 참조한 뒤 평균을 취하거나 이어 붙임
- 여기에 U 라는 행렬을 내적하고 바이어스 벡터 b를 더해줌
- 이렇게 만든 y에 소프트맥스를 취하면 확률 벡터가 됨
- U 의 크기는 어휘 집합 크기 X 임베딩 차원 수

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

수식 1

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

수식 2

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

수식 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 SGD and backpropagation.
- The paragraph vectors and word vectors are trained using stochastic g

radient descent and the gradient is obtained via backpropagation.

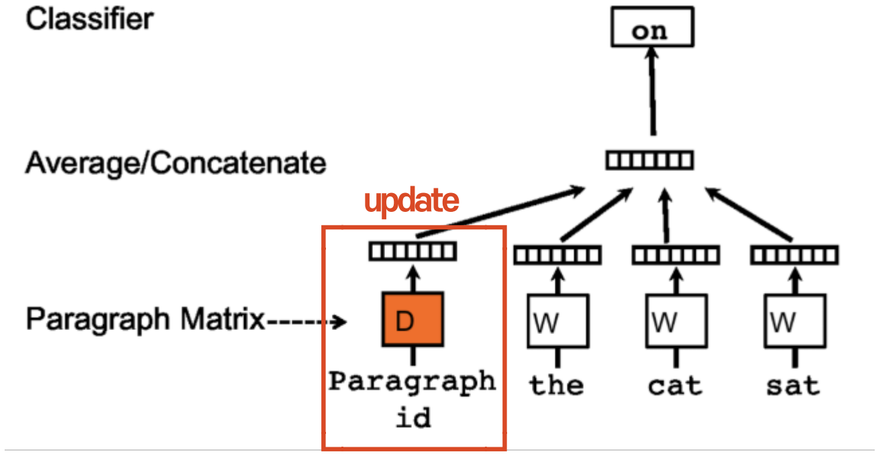


그림 2. Doc2Vec PV-DM

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

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.

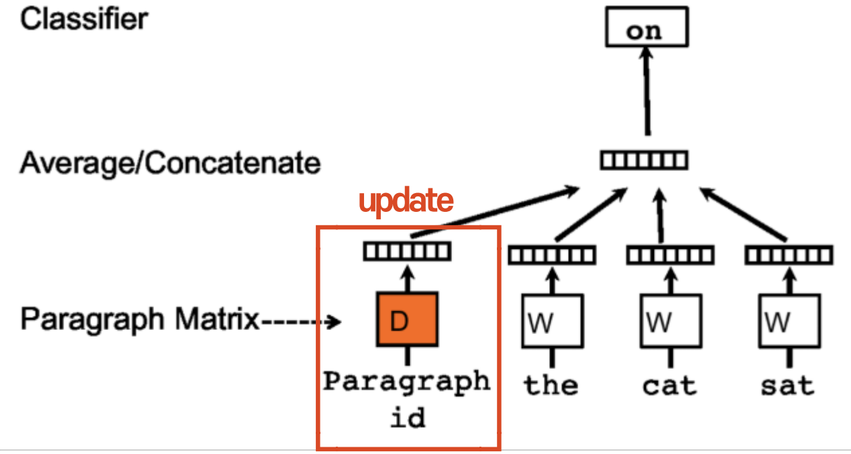


그림 2. Doc2Vec PV-DM

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.

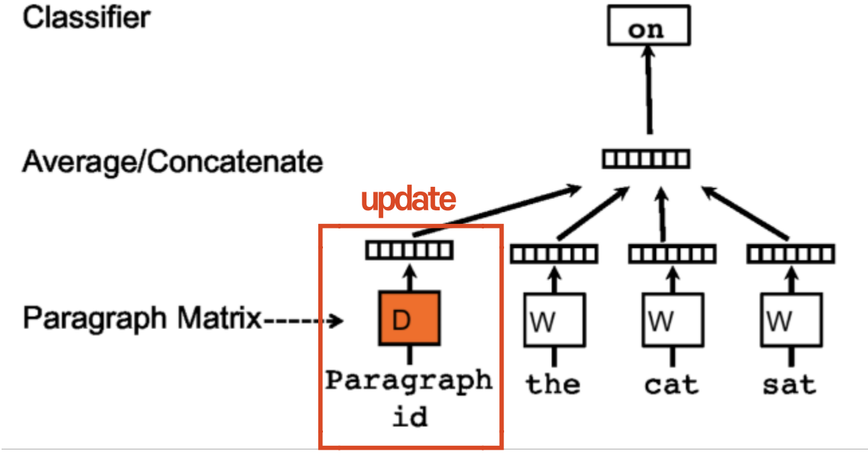


그림 2. Doc2Vec PV-DM

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.

2. Algorithms

2.2 Paragraph Vector : A distributed memory model

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]

2. Algorithms

2.3 Paragraph Vector : A distributed memory model

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

- Another way is to ignore the context words in the input, but force the model to predict words randomly sampled from the p aragraph 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.

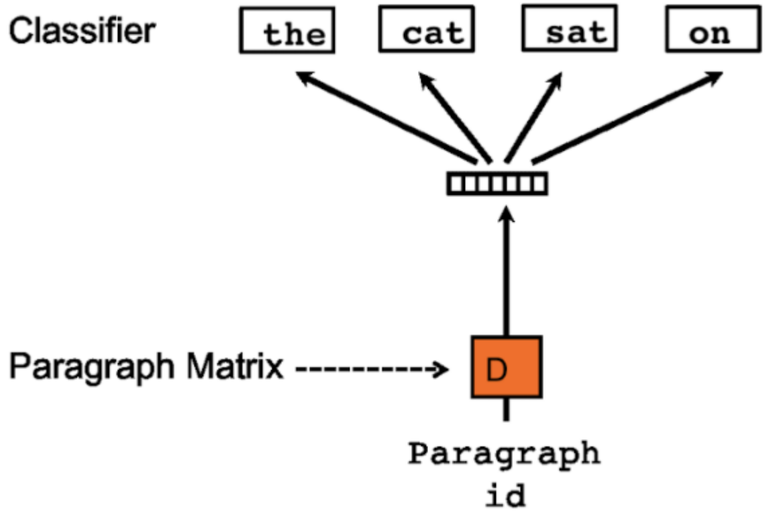


그림 3. Doc2Vec PV-DBOW

2. Algorithms

2.3 Paragraph Vector : A distributed memory model

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

- ONLY use paragraph vec(no word vecs!)
- Take window of words in praragraph and randomly sample which one to predict using paragraph vec(igores word ordering)
- Simpler, more memory effcient
- DM typically outperforms DBOW (but DM + DBOW is even better)

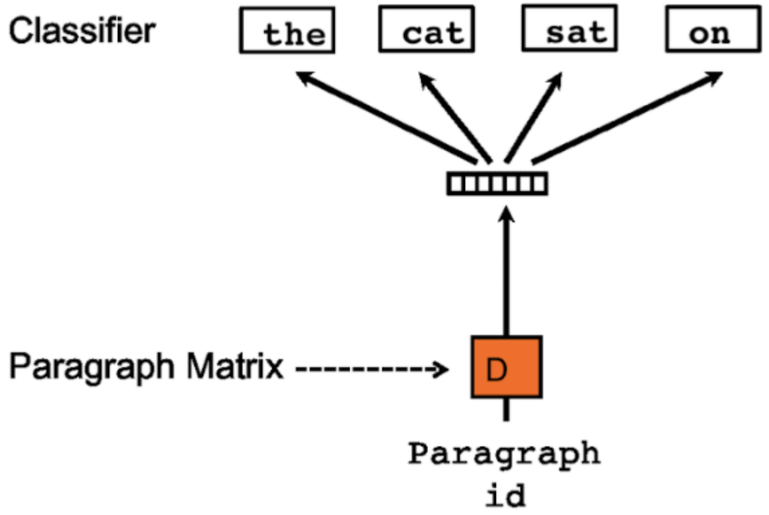


그림 3. Doc2Vec PV-DBOW

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

• In our experiments, each paragraph vector is a combination of two vectors: one learned by the standard paragraph vector with distributed memory (PV-DM) and one learned by the paragraph vector with distributed bag of words (PV- DBOW).

• We perform experiments to better understand the behavior of the paragraph vectors. To achieve this, we benchmark Paragraph Vector on a text understanding problem that require fixed-length vector representations of paragraphs: sentiment analysis.

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).

• 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)

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

Tasks and Baselines

- {Very Negative, Negative, Neutral, Positive, Very Positive} 5-way fine-grained classification task or
  {Negative, Positive} 2-way coarse-grained classification task

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

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 indepe ndent 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 t hem to a logistic regression to learn a predictor of the movie rating.

• The vector presented to the classifier is a concatenation of two vectors, one from PV-DBOW an d one from PV-DM.

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 3.1 Sentiment Analysis with the Stanford Sentiment Treebank Dataset

Results

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.
- One key aspect of this dataset is that 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 classifier to learn to predict the sentiment.

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

Results

- Once again, 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

• 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 achie ve 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 s ame 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

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(Rumelhart et al., 1986) -> statistical language modeling (Elman, 1990; B engio 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, 20 10; 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; Grefenstet te et al., 2013; Mikolov et al., 2013c)

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

• (Socher et al., 2011a;c; 2013b)

• computing the paragraph vectors via gradient descent(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 datas ets 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