Distributed Representations of Sentences and Documents

1 Citation

Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." International Conference on Machine Learning. 2014.

https://arxiv.org/pdf/1405.4053.pdf

2 Abstract

Paragraph vector is an unsupervised technique that can turn a paragraph into a fixed-size vector.

3 Introduction

Bag of Words (BOW) is limited because it loses word order and does not account for semantically similar words. Our model learns paragraph vectors by using the paragraph vector and some word vectors to predict words in the paragraph.

4 Algorithms

Let's look at learning word vectors first. Given a corpus of words $w_1, w_2, ..., w_T$, we aim to maximize:

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

where we model the probability with a softmax (in practice a hierarchical softmax is better):

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

where

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

where h represents concatentation or averaging and U, b are learnable.

The result is an end-to-end trainable neural network that we optimize with stochastic gradient descent.

How can we apply this to learn paragraph vectors? Let's first consider the Distributed Memory model of paragraph vectors.

Map each paragraph to a vector using a learnable embedding matrix D. We also map each word to a vector using a learnable embedding matrix W. At prediction time, concatentate (or average) the paragraph vector with the word vectors for $w_{t-k}, ..., w_{t-1}$ and try to predict w_t (note that $w_{t-k}...w_t$ must be in the paragraph). For N paragraphs mapped to p-length vectors and M words mapped to q-length vectors, this yields $Np + Mq$ parameters (excluding softmax params), but it works fine in practice because gradient updates are sparse (i.e. most words/paragraphs are untouched when training on a single example).

At test time, we create a new D for the new set of paragraphs. Then, we hold W, U, b fixed and learn D using the technique described above. The resulting Distributed Memory Paragraph Vectors (PV-DM) can be used for other tasks.

We also experiment with a Bag of Words Paragraph Vector (PV-DBOW), where we learn D by feeding the paragraph vector through the softmax and try to predict random words in the paragraph. PV-DM usually gets better accuracy than PV-DBOW on prediction tasks, but it is more compute-intensive. To get maximum accuracy, concatenate or average them.

5 Experiments

We test our paragraph vectors by trying to do sentiment analysis. The Stanford Sentiment Tree Bank dataset has 11K sentences, where every node in the parse tree is labeled with a 5-level sentiment label (very negative, negative, neutral, positive, very positve).

We learn unsupervised and use a logistic regression on paragraph vectors at test time. Hyperparameters are chosen with cross validation, special characters are treated as regular words, and if the sentence has < 9 words, we pad with NULL words.

We set state of the art.

The sentiment analysis dataset only contains sentences, so we might do even better on datasets that have many sentences. We use 100K example IMDB movie review dataset, which has multiple sentences in each review.

We learn our paragraph vectors unsupervised and use a fully-connected layer and logistic regression at test-time.

We set state of the art.

We also look at another task, where we consider several examples of 10 paragraph snippets (one per link) returned from a search query. We create a dataset of paragraph triples where two paragraphs are results of the same query. We aim to predict the odd-one-out. We compare our method against BOW, Bag of Bigrams, and averaging word vectors. Our Paragraph Vectors beat all of them.

We make some interesting observations. First, PV-DM is better than PV-DBOW, but it is best to use them both. Second, the word window k should be picked through cross validation. With 16 cores, learning paragraph vectors for 25K (230 word each) IMDB documents takes 30 minutes.

6 Related Work

Word vectors have been used for many tasks. People have used autoencoders to try and create paragraph vectors. Using parse trees can help model sentences, but is hard when you want to model paragraphs (i.e. many consecutive sentences).

7 Discussion

Our Paragraph Vectors set state of the art on sentiment analysis of both sentences and documents.