Learning Document Embeddings with Crossword Prediction*

Junyu Luo[1], Min Yang[2], Ying Shen[3], Qiang Qu[2], Haixia Chai[4]
[1] Department of Computer Science, Sichuan University
[2] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences
[3] School of Electronics and Computer Engineering, Peking University Shenzhen Graduate School
[4] Data Center of SNG, Tencent

Abstract

In this paper, we propose a Document Embedding Network (DEN) to learn document embeddings in an unsupervised manner. Our model uses the encoder-decoder architecture as its backbone, which tries to reconstruct the input document from an encoded document embedding. Unlike the standard decoder for text reconstruction, we randomly block some words in the input document, and use the incomplete context information and the encoded document embedding to predict the blocked words in the document, inspired by the crossword game. Thus, our decoder can keep the balance between the known and unknown information, and consider both global and partial information when decoding the missing words. We evaluate the learned document embeddings on two tasks: document classification and document retrieval. The experimental results show that our model substantially outperforms the compared methods.[1].

Introduction

Inspired by the recent success of word embedding methods (Bengio et al. 2003; Mikolov et al. 2013), several approaches have been proposed to learn document representations by taking the semantics of text data into consideration Sentences that share semantic and syntactic properties are thus mapped to similar vector representations. Despite significant improvements have been achieved by the recent neural network based methods such as the paragraph vector (Le and Mikolov 2014) and skip-thought (Kiros et al. 2015), they are essentially designed for processing sentences rather than long documents.

To learn high-quality representations for long documents, we propose a Document Embedding Network (DEN). DEN is an unsupervised model that uses autoencoder as its backbone to learn the distributed document embeddings directly for documents instead of using a sliding window to catch local context information. Concretely, DEN first encodes the input document with the LSTM network. Then, a bidirectional decoding network is designed to reconstruct the input sequence from the encoded embedding, leveraging the information from both past and future contexts. Unlike the standard decoder for text reconstruction, we take the idea from crossword game to consider both the partial and global contexts, and keep the balance between the known and unknown information. We randomly block some words in the input document, and use the incomplete context information and the encoded document embedding to predict the blocked words in the document.

The experiments on the document classification and document retrieval tasks demonstrate that our model outperforms the compared methods and achieves the state-of-the-art results.

*Min Yang is corresponding author (min.yang@siat.ac.cn).
This work is supported by CAS Pioneer Hundred Talents Program.

[1]Code is available at `https://github.com/doucmentnet/doument-network`.

Document Embedding Network

Encoding Network We assume that the input document d consists of n words $d = [w_1, w_2, \ldots, w_n]$, where w_i denotes the i-th word in document. Each word w_i in the document is mapped to a low-dimensional embedding $x_i \in \mathbb{R}^d$ through a word embedding layer, where d denotes the dimension of the embedding. We then employ a LSTM network to obtain the hidden state z_i of each word x_i in the document. Finally, we feed the last hidden state h_n to a non-linear activation function to get final document embedding e.

Crossword Model in Semantic Level Most previous studies (e.g., skip-gram) only leverage local context information and are defective to learn document embeddings. To overcome this limitation, we design a crossword model to capture both the partial and global context, inspired by the crossword game. The crossword game is a word puzzle that usually takes the form of a square or a rectangular grid of white-and black-shaded squares. The goal of the crossword game is to fill the white squares with letters, forming words or phrases, which lead to the answers. In (Crossman and Crossman 1983), crossword was proved to be a useful tool in teaching the children to learn the key concepts.

Like the crossword game, our crossword model tries to keep the balance between the known information and unknown information. We use the incomplete global contextual information and the corresponding document embedding to predict the missing words in the document. Concretely, we randomly block the original input document x (set the embedding to the zero vector) with a probability of 0.5 and get the representations of the new input sequence as

$\{x_1^{'}, ..., x_n^{'}\}$. To further deal with the representations of synonym words, we compute the loss in semantic-level, rather than in word-level. More formally, the prediction of missing word embedding x_j is typically done via a bidirectional decoding network (denoted as DecodeNet):

$$\hat{x}_j = \text{DecodeNet}(x_1^{'}, ..., x_n^{'}, e) \quad (1)$$

where x_j represents the blocked word embedding and $\hat{x}_j$ is the predicted word embedding at position j in the document d with size n.

The objective of our DEN model is to minimize the L_2 norm between the original blocked word embeddings and the predicted blocked word embeddings in the document d_i. In the next section, we will elaborate the bidirectional decoding network (DecodeNet) in details.

Bidirectional Decoding Network Given the blocked word embeddings $\{x_1^{'}, ..., x_n^{'}\}$ of document d and the document embedding e, the forward hidden state h_t^f and the backward hidden state h_t^f of bidirectional LSTM at time step t can be updated as:

$$h_t^f = \text{LSTM}_{\theta_d^f}(h_{t-1}^f, [x_t^{'}, e]); h_t^b = \text{LSTM}_{\theta_d^b}(h_{t+1}^b, [x_t^{'}, e]) \quad (2)$$

where θ_d^f and θ_d^b are parameters of the forward and backward LSTM networks, respectively. Finally, the prediction of the word embedding at position t is:

$$\hat{x}_t = W_p[h_t^f, h_t^b] + b_p \quad (3)$$

Here, W_p and b_p are learnable parameters.

Experiments

We apply the learned document representations to two applications: document classification and document retrieval.

Dataset We conduct extensive experiments on a publicly available dataset: 20Newsgroups[2]. It consists of 18,845 newsgroup documents from 20 different newsgroups, where each newsgroup corresponds to a specific topic. Following the preprocessing in (Xu et al. 2015), the dataset is partitioned into 10,443 training documents and 6,973 test documents. We randomly choose 10% of training data as validation set to set the hyperparameters. The average length of all the documents is 92.8, and the vocabulary size is 41,877.

Implementation Details We use 100-dimensional word2vec vectors to initialize the word embeddings for words in the corpus. The hidden state size of each encoder LSTM unit is 512. For both forward and backward decoder LSTMs, the hidden state sizes are set to 256. We set the size of the document embedding as 64. We conduct mini-batch (with size 32) training using Adam optimization algorithm to train the model. The learning rate is 0.001.

Baseline Methods We evaluate and compare our model with five strong competitors, including paragraph vector (PV) (Le and Mikolov 2014), skip-thought (ST) (Kiros et al. 2015), TF-IDF, latent Dirichlet allocation (LDA), and latent semantic indexing (LSA). We use the same parameter settings as in the papers where these methods were introduced.

[2] http://kdd.ics.uci.edu/databases/20newsgroups/20newsgroups.html

Evaluation Metrics For document classification task, we adopt classification accuracy as the evaluation metric. For document retrieval task, the precision at n (or $P@n$, $n \in [25, 50]$) and mean average precision (MAP) are used to estimate the performance of our model, which are widely adopted in retrieval task.

Methods	Classification	Retrieval task (MAP)		Retrieval task ($P@n$)	
	Accuracy	MAP@25	MAP@50	P@25	P@50
DEN	**0.757**	**0.671**	**0.632**	**0.585**	**0.554**
PV	0.707	0.602	0.567	0.523	0.501
ST	0.486	0.455	0.402	0.309	0.277
TF-IDF	0.647	0.593	0.555	0.499	0.472
LDA	0.307	0.253	0.218	0.139	0.126
LSA	0.362	0.369	0.328	0.244	0.220

Table 1: Extensive experiment results.

Document Classification After learning the document embeddings, we feed them to a logistic regression classifier to predict the document categories. The classification accuracy results are summarized in Table 1 (the second column). The results show that more advanced methods (e.g., PV) perform much better than the conventional models (e.g., LDA and LSA). Our method outperforms all these baseline methods.

Document Retrieval Following the setting in (Le and Mikolov 2014), documents in the training set are used as a database, while the test set is used as queries. For each query, documents in the database are ranked using cosine distance as the similarity metric. Table 1 (columns 3-6) reports the document retrieval results (MAP and $P@n$ values) that are averaged. Our model consistently and substantially outperforms the strong baseline methods by a large margin on all the evaluation metrics, especially when the value of n is small.

References

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *JMLR* 3:1137–1155.

Crossman, E. K., and Crossman, S. M. 1983. The crossword puzzle as a teaching tool. *Teaching of Psychology*.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *NIPS*.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, 3111–3119.

Xu, J.; Wang, P.; Tian, G.; Xu, B.; Zhao, J.; Wang, F.; and Hao, H. 2015. Convolutional neural networks for text hashing. In *IJCAI*.