KeyVec: Key-semantics Preserving Document Representations

Abstract

Previous studies have demonstrated the empirical success of word embeddings in various applications. In this paper, we investigate the problem of learning distributed representations for text documents which many machine learning algorithms take as input for a number of NLP tasks. We propose a neural network model, KeyVec, which learns document representations with the goal of preserving key semantics of the input text. It enables the learned low-dimensional vectors to retain the topics and important information from the documents that will flow to downstream tasks. Our empirical evaluations show the superior quality of KeyVec representations in two different document understanding tasks.

Introduction

In recent years, the use of word representations, such as word2vec BIBREF0 , BIBREF1 and GloVe BIBREF2 , has become a key “secret sauce” for the success of many natural language processing (NLP), information retrieval (IR) and machine learning (ML) tasks. The empirical success of word embeddings raises an interesting research question: Beyond words, can we learn fixed-length distributed representations for pieces of texts? The texts can be of variable-length, ranging from paragraphs to documents. Such document representations play a vital role in a large number of downstream NLP/IR/ML applications, such as text clustering, sentiment analysis, and document retrieval, which treat each piece of text as an instance. Learning a good representation that captures the semantics of each document is thus essential for the success of such applications.

In this paper, we introduce KeyVec, a neural network model that learns densely distributed representations for documents of variable-length. In order to capture semantics, the document representations are trained and optimized in a way to recover key information of the documents. In particular, given a document, the KeyVec model constructs a fixed-length vector to be able to predict both salient sentences and key words in the document. In this way, KeyVec conquers the problem of prior embedding models which treat every word and every sentence equally, failing to identify the key information that a document conveys. As a result, the vectorial representations generated by KeyVec can naturally capture the topics of the documents, and thus should yield good performance in downstream tasks.

We evaluate our KeyVec on two text understanding tasks: document retrieval and document clustering. As shown in the experimental section SECREF5 , KeyVec yields generic document representations that perform better than state-of-the-art embedding models.

Related Work

Le et al. proposed a Paragraph Vector model, which extends word2vec to vectorial representations for text paragraphs BIBREF3 , BIBREF4 . It projects both words and paragraphs into a single vector space by appending paragraph-specific vectors to typical word2vec. Different from our KeyVec, Paragraph Vector does not specifically model key information of a given piece of text, while capturing its sequential information. In addition, Paragraph Vector requires extra iterative inference to generate embeddings for unseen paragraphs, whereas our KeyVec embed new documents simply via a single feed-forward run.

In another recent work BIBREF5 , Djuric et al. introduced a Hierarchical Document Vector (HDV) model to learn representations from a document stream. Our KeyVec differs from HDV in that we do not assume the existence of a document stream and HDV does not model sentences.

KeyVec Model

Given a document INLINEFORM0 consisting of INLINEFORM1 sentences INLINEFORM2 , our KeyVec model aims to learn a fixed-length vectorial representation of INLINEFORM3 , denoted as INLINEFORM4 . Figure FIGREF1 illustrates an overview of the KeyVec model consisting of two cascaded neural network components: a Neural Reader and a Neural Encoder, as described below.

Neural Reader

The Neural Reader learns to understand the topics of every given input document with paying attention to the salient sentences. It computes a dense representation for each sentence in the given document, and derives its probability of being a salient sentence. The identified set of salient sentences, together with the derived probabilities, will be used by the Neural Encoder to generate a document-level embedding.

Since the Reader operates in embedding space, we first represent discrete words in each sentence by their word embeddings. The sentence encoder in Reader then derives sentence embeddings from the word representations to capture the semantics of each sentence. After that, a Recurrent Neural Network (RNN) is employed to derive document-level semantics by consolidating constituent sentence embeddings. Finally, we identify key sentences in every document by computing the probability of each sentence being salient.

Specifically, for the INLINEFORM0 -th sentence INLINEFORM1 with INLINEFORM2 words, Neural Reader maps each word INLINEFORM3 into a word embedding INLINEFORM4 . Pre-trained word embeddings like word2vec or GloVe may be used to initialize the embedding table. In our experiments, we use domain-specific word embeddings trained by word2vec on our corpus.

Given the set of word embeddings for each sentence, Neural Reader then derives sentence-level embeddings INLINEFORM0 using a sentence encoder INLINEFORM1 :

DISPLAYFORM0

where INLINEFORM0 is implemented by a Convolutional Neural Network (CNN) with a max-pooling operation, in a way similar to BIBREF6 . Note that other modeling choices, such as an RNN, are possible as well. We used a CNN here because of its simplicity and high efficiency when running on GPUs. The sentence encoder generates an embedding INLINEFORM1 of 150 dimensions for each sentence.

Given the embeddings of sentences INLINEFORM0 in a document INLINEFORM1 , Neural Reader computes the probability of each sentence INLINEFORM2 being a key sentence, denoted as INLINEFORM3 .

We employ a Long Short-Term Memory (LSTM) BIBREF7 to compose constituent sentence embeddings into a document representation. At the INLINEFORM0 -th time step, LSTM takes as input the current sentence embedding INLINEFORM1 , and computes a hidden state INLINEFORM2 . We place an LSTM in both directions, and concatenate the outputs of the two LSTMs. For the INLINEFORM3 -th sentence, INLINEFORM4 is semantically richer than sentence embedding INLINEFORM5 , as INLINEFORM6 incorporates the context information from surrounding sentences to model the temporal interactions between sentences. The probability of sentence INLINEFORM7 being a key sentence then follows a logistic sigmoid of a linear function of INLINEFORM8 :

DISPLAYFORM0

where INLINEFORM0 is a trainable weight vector, and INLINEFORM1 is a trainable bias scalar.

Neural Encoder

The Neural Encoder computes document-level embeddings based on the salient sentences identified by the Reader. In order to capture the topics of a document and the importance of its individual sentences, we perform a weighted pooling over the constituent sentences, with the weights specified by INLINEFORM0 , which gives the document-level embedding INLINEFORM1 through a INLINEFORM2 transformation:

DISPLAYFORM0

where INLINEFORM0 is a trainable weight matrix, and INLINEFORM1 is a trainable bias vector.

Weighted pooling functions are commonly used as the attention mechanism BIBREF8 in neural sequence learning tasks. The “share” each sentence contributes to the final embedding is proportional to its probability of being a salient sentence. As a result, INLINEFORM0 will be dominated by salient sentences with high INLINEFORM1 , which preserves the key information in a document, and thus allows long documents to be encoded and embedded semantically.

Model Learning

In this section, we describe the learning process of the parameters of KeyVec. Similarly to most neural network models, KeyVec can be trained using Stochastic Gradient Descent (SGD), where the Neural Reader and Neural Encoder are jointly optimized. In particular, the parameters of Reader and Encoder are learned simultaneously by maximizing the joint likelihood of the two components:

DISPLAYFORM0

where INLINEFORM0 and INLINEFORM1 denotes the log likelihood functions of Reader and Encoder, respectively.

Reader's Objective: $\mathscr{L}$ \mathcal {L}_{\tt read}

To optimize Reader, we take a surrogate approach to heuristically generate a set of salient sentences from a document collection, which constitute a training dataset for learning the probabilities of salient sentences INLINEFORM0 parametrized by INLINEFORM1 . More specifically, given a training set INLINEFORM2 of documents (e.g., body-text of research papers) and their associated summaries (e.g., abstracts) INLINEFORM3 , where INLINEFORM4 is a gold summary of document INLINEFORM5 , we employ a state-of-the-art sentence similarity model, DSSM BIBREF9 , BIBREF10 , to find the set of top-INLINEFORM6 sentences INLINEFORM8 in INLINEFORM9 , such that the similarity between INLINEFORM10 and any sentence in the gold summary INLINEFORM11 is above a pre-defined threshold. Note that here we assume each training document is associated with a gold summary composed of sentences that might not come from INLINEFORM12 . We make this assumption only for the sake of generating the set of salient sentences INLINEFORM13 which is usually not readily available.

The log likelihood objective of the Neural Reader is then given by maximizing the probability of INLINEFORM0 being the set of key sentences, denoted as INLINEFORM1 :

DISPLAYFORM0

where INLINEFORM0 is the set of non-key sentences. Intuitively, this likelihood function gives the probability of each sentence in the generated key sentence set INLINEFORM1 being a key sentence, and the rest of sentences being non-key ones.

Encoder's Objective: $\mathscr{L}$ \mathcal {L}_{\tt enc}

The final output of Encoder is a document embedding INLINEFORM0 , derived from LSTM's hidden states INLINEFORM1 of Reader. Given our goal of developing a general-purpose model for embedding documents, we would like INLINEFORM2 to be semantically rich to encode as much key information as possible. To this end, we impose an additional objective on Encoder: the final document embedding needs to be able to reproduce the key words in the document, as illustrated in Figure FIGREF1 .

In document INLINEFORM0 , the set of key words INLINEFORM1 is composed of top 30 words in INLINEFORM2 (i.e., the gold summary of INLINEFORM3) with the highest TF-IDF scores. Encoder's objective is then formalized by maximizing the probability of predicting the key words in INLINEFORM4 using the document embedding INLINEFORM5 :

DISPLAYFORM0

where INLINEFORM0 is implemented as a softmax function with output dimensionality being the size of the vocabulary.

Combining the objectives of Reader and Encoder yields the joint objective function in Eq (EQREF9). By jointly optimizing the two objectives with SGD, the KeyVec model is capable of learning to identify salient sentences from input documents, and thus generating semantically rich document-level embeddings.

Experiments and Results

To verify the effectiveness, we evaluate the KeyVec model on two text understanding tasks that take continuous distributed vectors as the representations for documents: document retrieval and document clustering.

Document Retrieval

The goal of the document retrieval task is to decide if a document should be retrieved given a query. In the experiments, our document pool contained 669 academic papers published by IEEE, from which top-INLINEFORM0 relevant papers are retrieved. We created 70 search queries, each composed of the text in a Wikipedia page on a field of study (e.g., https://en.wikipedia.org/wiki/Deep_learning). We retrieved relevant papers based on cosine similarity between document embeddings of 100 dimensions for Wikipedia pages and academic papers. For each query, a good document-embedding model should lead to a list of academic papers in one of the 70 fields of study.

Table TABREF15 presents P@10, MAP and MRR results of our KeyVec model and competing embedding methods in academic paper retrieval. word2vec averaging generates an embedding for a document by averaging the word2vec vectors of its constituent words. In the experiment, we used two different versions of word2vec: one from public release, and the other one trained specifically on our own academic corpus (113 GB). From Table TABREF15 , we observe that as a document-embedding model, Paragraph Vector gave better retrieval results than word2vec averagings did. In contrast, our KeyVec outperforms all the competitors given its unique capability of capturing and embedding the key information of documents.

Document Clustering

In the document clustering task, we aim to cluster the academic papers by the venues in which they are published. There are a total of 850 academic papers, and 186 associated venues which are used as ground-truth for evaluation. Each academic paper is represented as a vector of 100 dimensions.

To compare embedding methods in academic paper clustering, we calculate F1, V-measure (a conditional entropy-based clustering measure BIBREF11), and ARI (Adjusted Rand index BIBREF12). As shown in Table TABREF18 , similarly to document retrieval, Paragraph Vector performed better than word2vec averagings in clustering documents, while our KeyVec consistently performed the best among all the compared methods.

Conclusions

In this work, we present a neural network model, KeyVec, that learns continuous representations for text documents in which key semantic patterns are retained.

In the future, we plan to employ the Minimum Risk Training scheme to train Neural Reader directly on original summary, without needing to resort to a sentence similarity model.