Expert Systems With Applications 96 (2018) 103–114

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Incorporating knowledge into neural network for text representation

Yiming Li [a], Baogang Wei [a,*], Yonghuai Liu [b], Liang Yao [a], Hui Chen [a], Jifang Yu [a], Wenhao Zhu [c]

[a] College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China
[b] Department of Computer Science, Aberystwyth University, Aberystwyth SY23 3DB U.K
[c] School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China

ARTICLE INFO

Article history:
Received 4 July 2017
Revised 14 November 2017
Accepted 15 November 2017
Available online 22 November 2017

Keywords:
Text representation
Knowledge base
Representation learning
Network embedding

ABSTRACT

Text representations is a key task for many natural language processing applications such as document classification, ranking, sentimental analysis and so on. The goal of it is to numerically represent the unstructured text documents so that they can be computed mathematically. Most of the existing methods leverage the power of deep learning to produce a representation of text. However, these models do not consider about the problem that text itself is usually semantically ambiguous and reflects limited information. Due to this reason, it is necessary to seek help from external knowledge base to better understand text.

In this paper, we propose a novel framework named Text Concept Vector which leverages both the neural network and the knowledge base to produce a high quality representation of text. Formally, a raw text is primarily conceptualized and represented by a set of concepts through a large taxonomy knowledge base. After that, a neural network is used to transform the conceptualized text into a vector form which encodes both the semantic information and the concept information of the original text. We test our framework on both the sentence level task and the document level task. The experimental results illustrate the effectiveness of our work.

1. Introduction

Learning good representations of text plays an important role in many natural language processing (NLP) tasks, such as document classification, ranking, sentimental analysis and so on. Different representations may capture and disentangle different degrees of explanatory ingredients hidden in the text. Therefore, it has attracted considerable amount of attention from many researchers, and various types of models have been proposed for text representations.

The commonly used text representation is bag-of-words (BOW) (Harris, 1981). The BoW model regards text as unordered sets of words and words' interactions. Each word is a unique feature. However, the BOW model ignores the word order and syntactic features. Therefore, different sentences may have the same representation.

Recently, neural network based methods have provided new solutions for text representation and achieved remarkable results in many applications compared with BoW models. These models can extract features for variable-length text, among phrases, sentences and documents. Recurrent Neural Networks (RNN) is a kind of order-sensitive model by utilizing the sequence of the context words. Through its recurrent structure, RNN can naturally process sequence information, which takes the word order into consideration. However, the recurrent structure also leads to the problem that former words have less effect on the final representation of text than latter words. Different from RNN, Convolutional Neural Networks (CNN) regards each word fairly through a convolution layer and leverages sliding windows with different width and filters to perform the feature mapping. Afterwards, max pooling operation is utilized to obtain a fixed-length output.

The above methods have been proved their effectiveness (Johnson & Zhang, 2015; Kim, 2014; Zhang, Zhao, & LeCun, 2015). However, they don't consider about unique properties of texts. For example, image and speech understanding rely more on the information contained in the image and speech themselves than the background knowledge, while text understanding often needs to seek help from various external knowledge since text itself only reflects limited information and is sometimes ambiguous. In other words, the structures of neural network methods on the speech and image domains lies in its capability of discovering important

* Corresponding author.
E-mail addresses: liyiming90@zju.edu.cn (Y. Li), wbg@zju.edu.cn (B. Wei), yyl@aber.ac.uk (Y. Liu), yaoliang@zju.edu.cn (L. Yao), chhui@zju.edu.cn (H. Chen), 11521051@zju.edu.cn (J. Yu), whzhu@shu.edu.cn (W. Zhu).

https://doi.org/10.1016/j.eswa.2017.11.037
0957-4174/

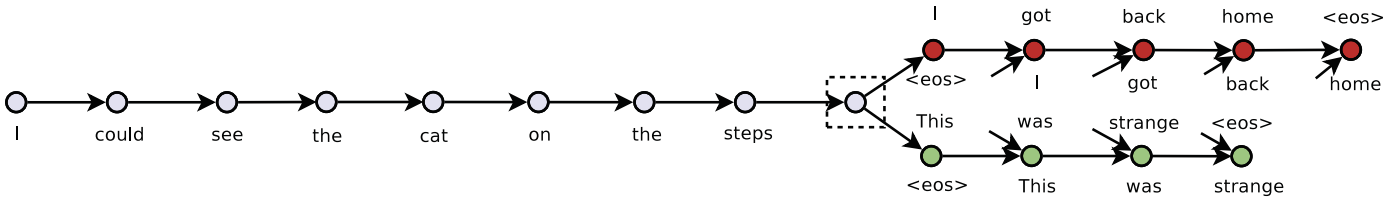


Fig. 1. The skip-thoughts model. The input is the sentence triplet *I got back home. I could see the cat on the steps. This was strange*. Unattached arrows are connected to the encoder output. Colors indicate which components share parameters. ⟨eos⟩ is the end of sentence token.

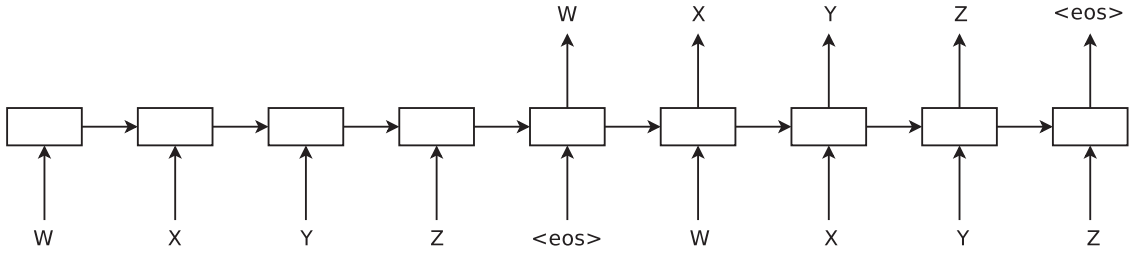


Fig. 2. The sequence autoencoder for the sequence "WXYZ". The sequence autoencoder uses a recurrent network to read the input sequence into the hidden state, which can then be used to reconstruct the original sequence.

signals from noisy input, while the major challenge for text understanding is instead the missing information and semantic ambiguity (Bian, Gao, & Liu, 2014).

To address these problems, we propose a new framework named Text Concept Vector (TCV) for the text representation which extracts the concept level information of text. For the first step, we recognize entities in text through Backward Maximum Matching (Sproat, Gale, Shih, & Chang, 1996) algorithm. Afterwards, we determine the sense (i.e., concept) of the entity by leveraging a large taxonomy knowledge base.

After that, raw text is transformed into conceptualized text which is composed of a set of concepts. Then the id of the text and its concepts are separately mapped to a unique vector. A neural network is used for training these two kinds of vectors through the stochastic gradient descent. After training, the text concept vectors can be regarded as the concept level features of the original text and used to natural language processing (NLP) tasks such as text classification.

To evaluate the performance of our framework, we test it on both the sentence level task: semantic relatedness of the sentence pair on SICK dataset (Marelli et al., 2014), and the document level task: document sentiment classification on IMDB reviews and Yelp reviews.

The main contributions of this paper are as follows:

- We propose a text concept extracting method to transform a raw text into a conceptualized text. This helps us to capture the background concept information of the original text and avoid the surface match.
- Based on the neural network, we achieve a vector representation of the conceptualized text. By incorporating the knowledge and neural network, our framework not only extracts the concept level information of the text but also takes term order into consideration.
- The framework we propose is a kind of architecture to produce knowledge based text features. Although we chose Probase (Wu, Li, Wang, & Zhu, 2012) as the running example, other knowledge bases also can be applied to our framework
- We test our framework on both the sentence level task and the document level task. The experimental results illustrate the effectiveness of our work.

The rest of the paper is organized as follows. Section 2 briefly reviews the related work. Section 3 introduces the knowledge base that we use in this paper. Section 4 describes the whole framework of Document Concept Vector. Experimental results are presented and analyzed in Section 5. Finally, Section 6 concludes the paper.

2. Related work

In recent years, numerous methods have been proposed for learning text representation and used for various kinds of text mining tasks. Le and Mikolov (2014) proposed doc2vec method by extending the word2vec algorithm (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). A document id is treated as a word in every sliding window and train the same word2vec again to obtain the document vector. This extension further improves the understanding of long paragraphs. Kiros et al. (2015) proposed skip-thought model which applies the word2vec algorithm to the sentence level. This model reconstructs surrounding sentences of the input sentence by using an encoder-decoder structure. The encoder and decoder are RNNs. Given a tuple (s_{i-1}, s_i, s_{i+1}) of contiguous sentences, with s_i the ith sentence of a book, the sentence s_i is encoded and tries to reconstruct the previous sentence s_{i-1} and next sentence s_{i+1}. The model is shown in Fig. 1. Dai and Le (2015) proposed a semi-supervised sequence learning method. In this model, the sequence autoencoder uses a recurrent network to read the input sequence in to the hidden state, which can then be used to reconstruct the original sequence. The weights for the decoder network and the encoder network are the same and these weights are used as an initialization for standard LSTM to improve training and generalization. The output layer of this model predicts the document label from the LSTM output at the last time step. This model is shown in Fig. 2. Hierarchical Document Vector (HDV) (Djuric, Wu, Radosavljevic, Grbovic, & Bhamidipati, 2015) learned text representation through a document stream. The model first learns the document representation similar to doc2vec and then enriches the representations through exploiting the document sequence in the stream. Doc2Sent2Vec (Ganesh, Gupta, & Varma, 2016) is also a two-phase approach for learning document embedding. Firstly, it learns the sentence embeddings using the word sequence generated from the sentence. In the next phase, it learns the document representation from the sentence sequence generated from the document. The difference between HDV and Doc2Sent2Vec is that HDV doesn't model sentence while Doc2Sent2Vec doesn't assume the existence of a document stream. The deep learning based methods also make

a contribution to text representation. Kim (2014) used convolutional neural networks (CNN) to the sentence classification directly by defining an architecture with two channels for capturing local and global semantic. Johnson and Zhang (2015) improved the performance by utilising high-dimensional one-hot vectors as input. Zhang et al. (2015) proposed a character-level convolutional networks (ConvNets) which treats text as a kind of raw signal at character level. Huang and Wang (2016) introduced this method into Chinese corpus. Socher et al. (2013) proposed several kinds of recursive neural networks for sentence-level text classification. The extended recursive neural networks varied between deep recursive layer (İrsoy & Cardie, 2014), global feed backward (Paulus, Socher, & Manning, 2014), tuning feature weight (Li, Luong, Jurafsky, & Hovy, 2015), adaptive composition functions (Dong, Wei, Zhou, & Xu, 2014), and combining with Combinatory Categorial Grammar (Hermann & Blunsom, 2013). Most of these deep learning based techniques directly leverage the models that are used in other domains like computer vision and speech. However, these attempts ignore the unique properties of text.

Some works studied on the structure of models. Lai, Xu, Liu, and Zhao (2015) and Zhou, Sun, Liu, and Lau (2015) incorporated CNN and long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) structure. Tai, Socher, and Manning (2015) used tree-structured LSTMs to improve the semantic representations of text. These works were applied to sentence classification. Tang, Qin, and Liu (2015a) modeled documents through a hierarchical structure method. The sentence vectors of a document were initially produced by a CNN or a LSTM, then a gated recurrent neural network was utilized to compose the sentence vectors for the document vectors generating. Tang, Qin, and Liu (2015b) incorporated user and product information into a convolutional neural network for modeling documents. Gui, Xu, He, Lu, and Wei (2016) used a inter subjectivity network to obtain user representation and combines it with a CNN for document classification. Gui, Zhou, Xu, He, and Lu (2017) utilized a heterogeneous network to model the shared polarity in product reviews and learn representations of users, products they commented on and words they used simultaneously. These method leverage such text related information into consideration, however, they can be applied to the specific task such as sentiment text classification.

There are some works utilize knowledge based methods to understand text. Yao, Zhang, Wei et al. (2017) proposed KGE-LDA which incorporates knowledge graph embeddings into topic model. This method explicitly models document-level word co-occurrence in a corpus with knowledge encoded by entities embeddings automatically learned from an external knowledge base in a unified model, which could extract more coherent topics and better representation of a document in the topic space. Nakashole and Flauger (2017) proposed KBLSTM which utilizes knowledge graph embeddings to enhance the learning of recurrent neural networks for machine reading. At each time step, KBLSTM retrieves KB concepts that are potentially related to the current word. Comparing with our method, these two methods also uses external knowledge base to better understand text. Both of them learn distributed representations of WordNet (Miller, 1995) concepts using knowledge graph embedding methods. However, these models can't solve the ambiguous problem of the text. Wei, Zhang, Zeng, and Li (2016) proposed a model named TRMBK to establish background knowledge automatically and offer supports for the current text comprehension. In this method, each domain in datasets needs to build its own background knowledge and background knowledge is constructed through simply association rules. Different from TRMNK, our method can be applied to different kinds of text since Probase is a large scale general knowledge base.

3. Preliminary knowledge

In this section, we will have a brief description of the knowledge base and its two functions which we use to conceptualize text.

3.1. External knowledge base

For better representing text, we utilize external knowledge base to capture the background concept information of text. Existing large scale knowledge bases, such as Freebase (Bollacker, Evans, Paritosh, Sturge, & Taylor, 2008), YAGO (Suchanek, Kasneci, & Weikum, 2007), Wikipedia (Gabrilovich & Markovitch, 2006) and Probase (Wu et al., 2012), are available for this task. YAGO builds on entities and relations and currently contains more than 1 million entities and 5 million facts. The facts have been automatically extracted from Wikipedia and unified with WordNet. Besides, YAGO is compatible with RDFS. Freebase is a practical, scalable tuple database used to structure general human knowledge. The data in Freebase is collaboratively created, structured, and maintained. Freebase currently contains more than 125,000,000 tuples, more than 4000 types, and more than 7000 properties. Probase is an universal, general-purpose, probabilistic taxonomy automatically constructed from entire web and it integrates most of the worldly facts. It contains 2.7 million concepts harnessed automatically from a corpus of 1.68 billion web pages.

In this paper, we choose Probase as the external knowledge base. Comparing with Freebase, YAGO and Wikipedia, entities and concepts are associated with each other in the form of probability. This setting allows us to use Probase itself to disambiguate the entity in the text so that we can achieve more precise concepts of the text

In probase, an entity may belong to many concepts and a concept may contain many entities. Concepts and Entities are connected by various relations. The benefit is that such links are data-driven rather than handcrafted (Wu et al., 2012).

In our work, we use *isA* relation to describe the relation between an entity and a concept. The *isA* relation in Probase is harvested from 1.68 billion web pages and two-years' worth of Microsoft Bings search log using syntactic patterns (i.e., the Hearst patterns and the *isA* pattern) (Hearst, 1992). For an example, "Apple is an company". Here "company" is a concept c and "Apple" is an entity e.

3.2. Typical score

In probase, an entity may belong to many concepts and a concept may contain many entities. Each *isA* relation (e *isA* c) is presented as conditional probabilities $P(e|c)$ and $P(c|e)$ (a.k.a. Typical Score). The $P(c|e)$ means how typical a concept c is among the concepts which contain the entity e while $P(e|c)$ denotes the typical score of e when c is given:

$$P(c|e) = \frac{n(e,c)}{n(e)}, \quad P(e|c) = \frac{n(e,c)}{n(c)}.$$

Here, $n(e)$, $n(c)$ and $n(e, c)$ are the occurrences of e, c and co-occurrences of e and c in the Hearst extraction. With these two conditional probabilities, we can quantify the relationship between an entity e and a concept c so that typical concepts of an entity can be obtained.

3.3. Concept cluster

Since there are millions concepts in probase, many of them are similar to each other, such as "company" and "corporation",

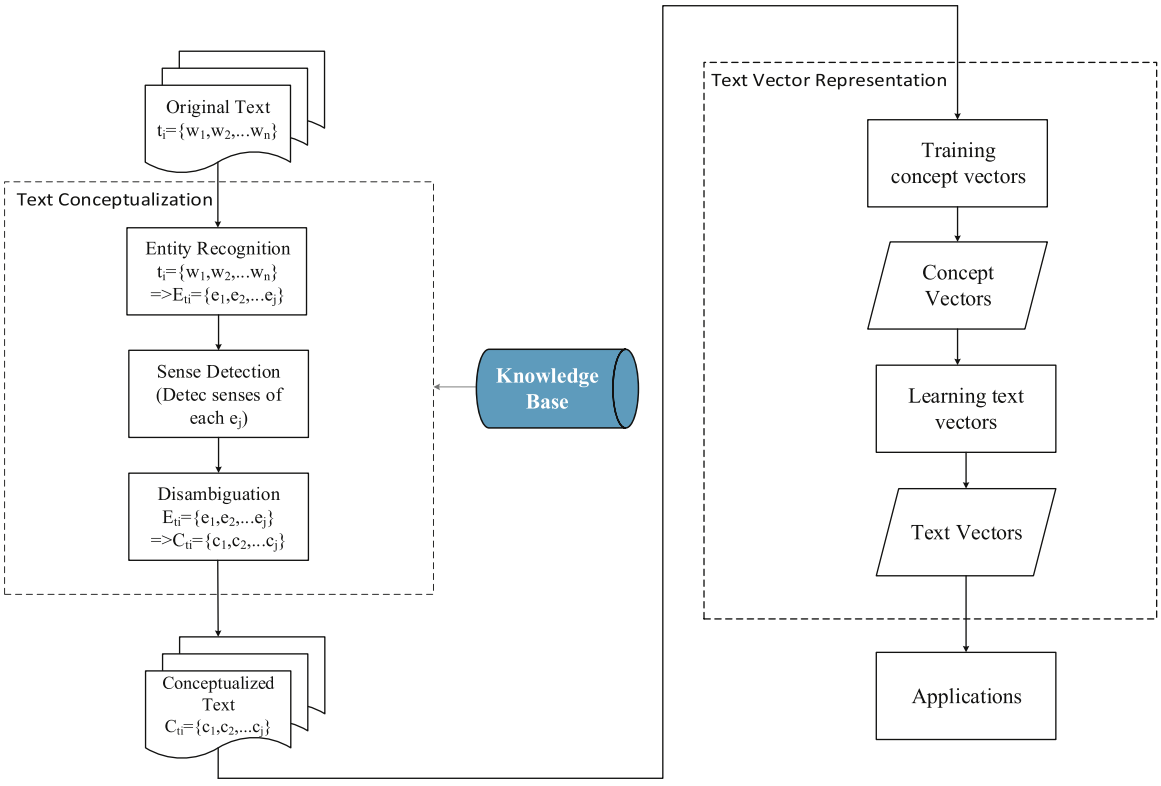


Fig. 3. The framework of Text Concept Vector.

"movie" and "film", etc. These similar concepts are clustered into a same cluster through a K-Medoids algorithm (Li, Wang, Zhu, Wang, & Wu, 2013). A concept cluster has a central concept to present the general sense of this cluster. For example, the concepts around the central concept "company" are related to "company", such as "corporation", "large company", "international company", "manufacturer" etc. In our framework, concept clusters are used in sense detection and disambiguation.

4. The Text Concept Vector framework

In this section, we will introduce the Text Concept Vector (TCV), a framework for the text representation which consists of two steps: Text Conceptualizing and Text Vector Representation. The whole framework is shown in Fig. 3.

4.1. Text conceptualization

This part is inspired by BocSTC (Wang, Wang, Li, & Wen, 2014). The goal of text conceptualization is to transform a raw text into a set of concepts. For an original text which consists of n words $t_i = \{w_1, w_2, w_3, \ldots, w_n\}$, we primarily operate entity recognition of the text $E_{t_i} = \{e_1, e_2, e_3, \ldots, e_j\}$, and then capture the precise concepts of the entities $C_{t_i} = \{c_1, c_2, c_3, \ldots, c_j\}$. Here, c_j indicates the concept of *jth* entity in the original text. This step is composed of following three steps: Entity Recognition, Sense Detection and Disambiguation. For example, after entity recognition step, text "*Beyonce music and songs*" has entity list $E_t = \{Beyonce, music, songs\}$. After sense detection step, vague entity *Beyonce* has senses *{artist, designer}*, unambiguous entities *music* and *songs* have same sense *music*. After disambiguation step, vague entity *Beyonce* has precise sense *artist*.

4.1.1. Entity recognition

Entity recognition is the first stage for text conceptualization. With recognized entities, we can achieve their concepts through the knowledge base. In recognition stage, a raw text is first split into sentences, and then the Backward Maximum Matching (BMM) (Sproat et al., 1996) is leveraged to recognize entities in the sentences. BMM is proposed to deal with the Chinese segmentation as each Chinese word is composed of separate Chinese characters. We use BMM in English corpus for entity recognition purpose for the reason that an entity in Probase is composed of separate words, which is similar to Chinese. Comparing with other popular string matching algorithms such as Aho-Corasick String Matching, BMM is easy to implement and efficient enough. Specifically, BMM selects a string S with N words from right to left of text as the maximum string (N<=maxlen), and uses it to match entities in Probase. If there is such an entity Probase, the match is successful and the string is segmented from text. If there is no such an entity, the match is failed. Then the algorithm removes the first word in the string and matches the remaining strings again until the match is successful or the length of the remaining string is 0. The flow chart of BMM is show in Fig. 4. Due to the abundance of the entities in the Probase that we have used, all entities in Probase are utilized as the matching dictionary. Maxlen in BMM is set to 5. This is because that most of entities in Probase are composed of less than 5 words. After entity recognition, we obtain an entity list $E_{t_i} = \{e_1, e_2, e_3, \ldots, e_j\}$ of the original text.

4.1.2. Sense detection

As described in Section 3.2, each entity in a text may belong to many concepts. In this stage, the goal is to detect senses (i.e., concepts) of each entity in a text entity list E_{t_i} and determine whether an entity is ambiguous or not in the context. Specifically, for an entity e_j in a text entity list E_{t_i}, we choose top N_t concepts as its typical concepts ar first. The top N_t concepts are ranked by the Typical Score $P(c|e)$ (N_t=15 in this paper). After that, the stop con-

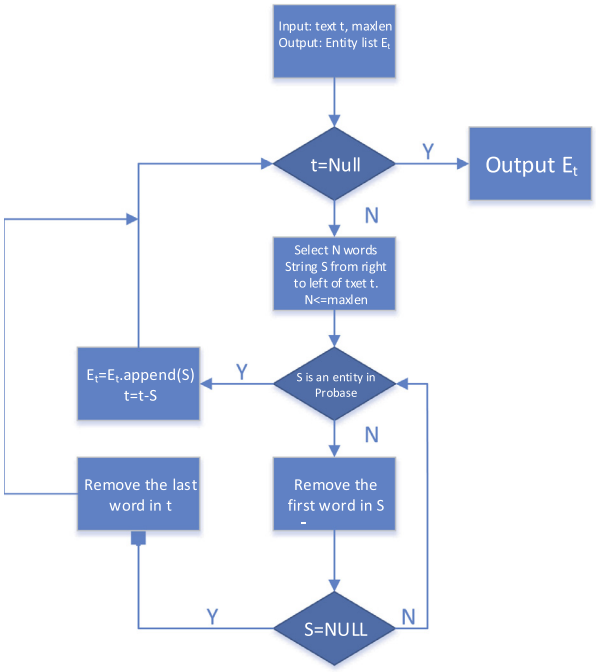


Fig. 4. The flow chart of Backward Maximum Matching.

cepts, which generally have many diverse instances and tend to be in the high level of concept hierarchy (these concepts are already pre-recognized with these two rules in Probase), are removed. For an entity e_j, we now have its typical concepts list $C_{e_j} = \{c_k, k = 1, 2, \ldots, N_t\}$. Since the concepts are clustered into different clusters, we also have a concept cluster set of an e_j entity $CCL_{e_j} = \{ccl_m, m = 1, 2, \ldots\}$. For example, entity *Beyonce* has the typical concepts list $C_{Beyonce} = \{singer, musician, artist, model, designer\}$ and the concept cluster set $CCL_{Beyonce} = \{artist, design\}$. After above steps, we calculate the entropy of the concept cluster distribution and regard it as the ambiguity of e_j:

$$H(e_j) = -\sum_{ccl_m \in CCl_{e_j}} P(ccl_m|e_j) \times log_2 P(ccl_m|e_j). \quad (1)$$

Here, $P(ccl_m|e_j)$ is the probability of e_j belonging to the concept cluster ccl_m. It is calculated by cumulating the typical scores of all e_j's concepts belonging to ccl_m. In our work, we assume the 30% lowest entropy entities are unambiguous entities.

After this stage, for each entity e_j in a text entity list E_{t_i}, we have its concept list C_{e_j}, concept cluster list CCL_{e_j} and entropy value $H(e_j)$.

4.1.3. Disambiguation

For vague entities in a text entity list E_{t_i}, the unambiguous entities in the context are used for disambiguation. Specially, for every concept cluster ccl_m in the concept cluster set of an vague entity $CCL_{e_i^v}$, $P(ccl_m|e_i^v)$ of a vague entity e_i^v is re-calculated as:

$$P'(ccl_m|e_i^v) = \sum_{ccl_n \in CCL_{e_j^u}, e_j^u \in E_{t_i}} CS(ccl_m, ccl_n). \quad (2)$$

Here e_j^u means the unambiguous entity and $CCL_{e_j^u}$ is the concept cluster set of an unambiguous e_j^u. $CS(ccl_m, ccl_n)$ is the concept cluster similarity which is calculated as follow:

$$CS(ccl_m, ccl_n) = \frac{1}{|ccl_m|} \sum_{c_k \in ccl_m} Max_{c_j \in ccl_n} \frac{\left|E_{c_k} \cap E_{c_j}\right|}{\left|E_{c_k} \cup E_{c_j}\right|}. \quad (3)$$

Here, c_k/c_j means concepts belong to the concept cluster ccl_m/ccl_n and E_{c_k}/E_{c_j} means all the entities belong to the concept c_k/c_j including the typical entities. For the vague entity, the central concept of the concept cluster that has the highest probability $P'(ccl_m|e_i^v)$ is regarded as the e_i^v's sense. For the unambiguous entity, the central concept of the domain cluster is selected as its sense. For example, $P(artist|Beyonce) = 0.5$, $P(design|Beyonce) = 0.5$, $P(music|music) = 1$ and $P(song|music) = 1$. *Beyonce* is vague entity while *music* and *song* are unambiguous entities. Recalculated $P'(artist|Beyonce) = 0.324$ and $P(design|Beyonce) = 0.036$ so that the sense of *Beyonce* is *artist*. After finishing the above steps, an original text $t_i = \{w_1, w_2, w_3, \ldots, w_n\}$ is conceptualized and represented by a set of concepts $C_{t_i} = \{c_1, c_2, c_3, \ldots, c_j\}$.

4.2. Text vector representation

With the conceptualized text $C_{t_i} = \{c_1, c_2, c_3, \ldots c_j\}$, we utilize a neural network based method to generate the text vector representation which is similar to Paragraph Vector model (Le & Mikolov, 2014). This method contains two steps. Firstly, we utilize Skip-gram model (Mikolov, Chen, Corrado, & Dean, 2013; Mikolov, Sutskever et al., 2013) to train the concept embeddings. Secondly, we put the document id into every sliding window as a concept and train the same skip-gram model again to achieve the text vector.

4.2.1. Concept embeddings

The well-known *word2vec* (Mikolov, Chen et al., 2013; Mikolov, Sutskever et al., 2013) (including the continuous bag-of-word (CBOW) model and the Skip-Gram model) is a kind of model for constructing distributed representations of words and Phrases. Benefiting from its structure, word2vec can encode the semantic and syntactic information into a continuous vector. Due to this advantage, similar words in a corpus are close to each other in the vector space. The continuous Bag-of-Words model utilizes the surrounding words of a word to maximize the classification of this word and does not take the word order into consideration. In contrast, the Skip-gram model uses the order of the sequence to sample the words that appear less frequently during training time (Franco-Salvador, Rangel, Rosso, Taulé, & Martít, 2015).

In this paper, we select the Skip-Gram model to train the concept embeddings due to its better performance on average especially at the semantic level (Yao, Zhang, Chen et al., 2017).

Formally, given a sequence of training concepts $c_1, c_2, c_3, \ldots c_T$, the objective of the Skip-gram is to maximize the average log probability of the central concept when given the surrounding concepts inside a window size

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-n \le j \le n, j \ne 0} \log p(c_{t+j}|c_t) \quad (4)$$

where c_t is the central concept and $c(t-n)$, $c(t-n+1)$,..., $c(t+n)$ are its surrounding concepts inside a window size $2n+1$.

Softmax function is used for estimate the conditional probability $p(c_{t+j}|c_t)$:

$$p(c_{t+j}|c_t) = \frac{\exp(\boldsymbol{v}_{c_{t+j}}^{\mathrm{T}} \boldsymbol{v}_{c_{t+j}})}{\sum_{c=1}^{C} \exp(\boldsymbol{v}_{c_t}^{\mathrm{T}} \boldsymbol{v}_{c_t})} \quad (5)$$

where $\boldsymbol{v}_{c_{t+j}}$ and $\boldsymbol{v}_{c_t}$ are the vector representations of c_{t+j} and c_t. Vectors are initialized randomly. C is the total number of concepts in all conceptualized text.

The average log probability is optimized through stochastic gradient descent. Updating of $\boldsymbol{v}_{c_{t+j}}$ and $\boldsymbol{v}_{c_t}$ are as follows:

$$\boldsymbol{v}_{c_{t+j}} - \alpha_s \left(\frac{\exp\{f(c_p)\}}{\exp\{f(c_p)\} + 1} - I\left[c_{t+j} = c_p\right] \right) \boldsymbol{v}_{c_t} \quad (6)$$

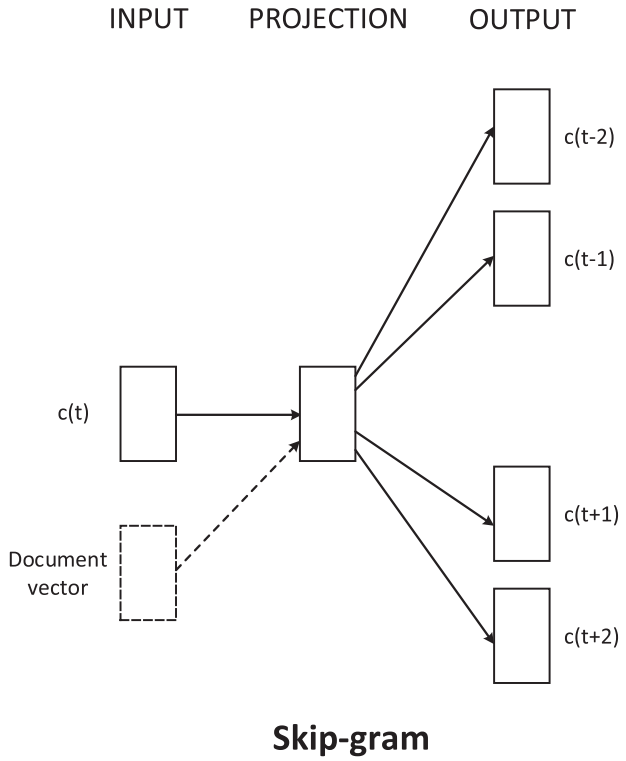


Fig. 5. The Skip-gram architecture we adapted for generating the concept embeddings. The goal is to predict concepts within a certain window size around the current concept. Dashed part is used for replacing $c(t)$ only when learning text vectors.

$$\boldsymbol{v}_{c_t} - \alpha_s \sum_{-n\le j\le n, j\neq 0} \left(\frac{\exp\{f(c_p)\}}{\exp\{f(c_p)\}+1} - I\left[c_{t+j} = c_p\right] \right) \boldsymbol{v}_{c_{t+j}} \tag{7}$$

where:

- $\boldsymbol{v}_{c_{t+j}}$ and $\boldsymbol{v}_{c_t}$ are the vector representations of c_{t+j} and c_t.
- c_p is the central concept predicted by the model.
- α_S is the learning rate.
- $f(c_p) = \boldsymbol{v}_{c_{t+j}}{}^{\mathrm{T}} \boldsymbol{v}_{c_t}$
- $I[x]$ equals 1 if x is true.

4.2.2. Learning text vector

After obtaining the concept embeddings, the adapted version of the Skip-gram architecture is used to produce the text vector *tv* for text representation. Specifically, before each context window movement, $c(t)$ is replaced by the current text id t_{id} with the objective of maximizing the average log probability (see Fig. 5).

$$\frac{1}{T}\sum_{t=1}^{T} \sum_{-n\le j\le n, j\neq 0} \log p(c_{t+j}|t_{id}). \tag{8}$$

Softmax function that is used for estimate the conditional probability $p(c_{t+j}|t_{id})$ is shown as follow:

$$p(c_{t+j}|t_{id}) = \frac{\exp(\boldsymbol{v}_{c_{t+j}}^{\mathrm{T}} \boldsymbol{v}_{c_{t+j}})}{\sum_{c=1}^{C} \exp(\boldsymbol{v}_{t_{id}}{}^{\mathrm{T}} \boldsymbol{v}_{t_{id}})}. \tag{9}$$

Here, $\boldsymbol{v}_{t_{id}}$ is the vector representation of current text (i.e., $\boldsymbol{tv}$).

4.3. Applications

After being trained, the text concept vector $\boldsymbol{tv}$ can be regarded as the feature of the original text and used for nlp tasks.

5. Experiments

To evaluate the effectiveness of the text representation, our model will be tested both on the sentence level task and the document level task. For the sentence level task, we test our model by predicting the semantic relatedness of sentence pairs. For the document level task, our model will be tested on sentiment classification.

5.1. Semantic relatedness

This experiment aims to measure the semantic relatedness of two sentences. For a given pair of sentences, the goal is to predict a real-valued score which means how semantically related these two sentences are.

5.1.1. Dataset

The dataset we use in this experiment is Sentences Involving Compositional Knowledge (SICK) dataset (Marelli et al., 2014). This dataset contains 4500 training pairs, 500 development pairs and 4927 testing pairs. Each sentences pair has a relatedness score which takes values between 1 and 5. A score of 1 means that two sentences are unrelated while a score of 5 indicates two sentences are very related. Each score is the average of 10 ratings labeled by 10 different human annotators. All sentences in the dataset are derived from existing image and video annotation datasets.

5.1.2. Implementation details

We take similar method for predicting the relatedness score of two sentences as (Tai et al., 2015). Specifically, we first use TCV to obtain the vector representation for each sentence in a sentence pair $(\boldsymbol{v}_{s_L}, \boldsymbol{v}_{s_R})$. The dimension of TCV is 200 and the window size is 3. We have tested different vector dimension and different window size. However, the results don't have statistically significant differences. These two parameters are not sensitive. With the vector representations, the relatedness score $\hat{y}$ can be calculated through a neural network. Specifically, we first calculate the element-wise product of $\boldsymbol{v}_{s_L}$ and $\boldsymbol{v}_{s_R}$, and their absolute difference and then we concatenate together. With the concatenate vector, we use a fully connected layer to transform it to real-valued vector whose length is 5. After that, we add a softmax layer to calculate the conditional probabilities and utilize an integer vector which ranges from 1 to 5 to produce the relatedness score.

$$\begin{aligned}
\boldsymbol{v}_{\times} &= \boldsymbol{v}_{s_L} \odot \boldsymbol{v}_{s_R}, \\
\boldsymbol{v}_{+} &= |\boldsymbol{v}_{s_L} - \boldsymbol{v}_{s_R}|, \\
\boldsymbol{v}_c &= concatenate\{\boldsymbol{v}_{\times}; \boldsymbol{v}_{+}\} \\
\boldsymbol{v}_h &= \boldsymbol{W}^{(h)}\boldsymbol{v}_c + \boldsymbol{b}^{(h)}, \\
\hat{\boldsymbol{p}}_{\boldsymbol{\theta}} &= softmax(\boldsymbol{v}_h), \\
\hat{y} &= \boldsymbol{r}^T \hat{\boldsymbol{p}}_{\boldsymbol{\theta}}.
\end{aligned} \tag{10}$$

Here, $\boldsymbol{v}_{\times}$ and $\boldsymbol{v}_{+}$ mean the element-wise product of $\boldsymbol{v}_{s_L}$ and $\boldsymbol{v}_{s_R}$, and their absolute difference. What's more, $\boldsymbol{r}^T = [1, \ldots, 5]$ is an integer vector which ranges from 1 to 5. The predicted rating $\hat{y}$ is expected to be close to the gold rating y:

$$\hat{y} = \boldsymbol{r}^T \hat{\boldsymbol{p}}_{\boldsymbol{\theta}} \approx y, y \in [1, 5] \tag{11}$$

Therefore, the target distribution p is constructed as follow:

$$p_i = \begin{cases} y - \lfloor y \rfloor, & i = \lfloor y \rfloor + 1 \\ \lfloor y \rfloor - y + 1, & i = \lfloor y \rfloor \\ 0 & otherwise \end{cases} \tag{12}$$

The KL-divergence between distribution $\hat{\boldsymbol{p}}_{\theta}$ and distribution $\boldsymbol{p}$ is used as the cost function:

$$J(\theta) = \frac{1}{m}\sum_{j=1}^{m} KL(\boldsymbol{p}^{(j)} || \hat{\boldsymbol{p}}_{\theta}^{(j)}) \tag{13}$$

where m is the total number of sentence pairs in the training pairs and the superscript j means the jth sentence pair. The cost function is optimized through AdaGrad (Duchi, Hazan, & Singer, 2011) with respect to the whole set of parameters $\theta = \left[\boldsymbol{W}^{(h)}; \boldsymbol{b}^{(h)}\right]$.

Pearson's r:

$$r = \frac{\sum_{i=1}^{n}(predicted_{s_i} - \overline{predicted_s})(real_{s_i} - \overline{real_s})}{\sqrt{\sum_{i=1}^{n}(predicted_{s_i} - \overline{predicted_s})^2}\sqrt{\sum_{i=1}^{n}(real_{s_i} - \overline{real_s})^2}}, \tag{14}$$

where:

- n is the total number of sentence pairs in the test set.
- $predicted_{s_i}$ indicates the predicted score of ith sentence pair in the test set.
- $real_{s_i}$ indicates the real score of ith sentence pair in the test set.
- $\overline{predicted_s}$ is the mean of predicted scores and analogously for $\overline{predicted_s}$.

Spearman's ρ:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2-1)}$$
$$d_i = rg(real_{s_i}) - rg(predicted_{s_i}) \tag{15}$$

where:

- $rg(real_{s_i})$ is the rank of ith sentence pair in test set sorted by the real score.
- $rg(predicted_{s_i})$ is the rank of ith sentence pair sorted by the predicted score.
- d_i means the difference between the two ranks.

Mean Squared Error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(predicted_{s_i} - real_{s_i})^2 \tag{16}$$

where n, $predicted_{s_i}$ and $real_{s_i}$ are defined as above.

5.1.3. Baselines

For this experiment, previous submissions in the SemEval 2014 competition and several state-of-the-art methods are served as baselines.

1. Tree-LSTM. The Tree-LSTM (Tai et al., 2015) is a kind of tree-structured network topologies. This network composes its state from an input vector and the hidden states of arbitrarily many child units. The updating of memory cell and gating vectors are dependent on the states of many child units.
2. Skip-Thought Vectors. The Skip-thought framework (Kiros et al., 2015) remodel the input sentence's surrounding sentences by leveraging an encoder-decoder structure. The encoder and decoder are RNNs.
3. CNN-LSTM encoder-decoder. The CNN-LSTM encoder-decoder method (Dai & Le, 2015) uses a CNN to perform pooling operations on the input sentence and a fully-connected layer is used to produce a fixed-length vector representation of the sentence. This vector is fed into a LSTM to produce target sentence.
4. Text Concept Vector with CBOW. This method is an adaption of original TCV. The Skip-gram is replaced by CBOW.

Table 1
The results of semantic relatedness on the SICK dataset. The first group are the results of SemEval 2014 submissions. The second group are the results of sequential LSTMs. The third group and fourth group are the baselines reported by Tai et al. (2015) and the results of its own methods (i.e., Constituency Tree-LSTM and Dependency Tree-LSTM). The fifth group are the results of Skip-Thought Vectors (Kiros et al., 2015). The sixth group and seventh group are the results of CNN-LSTM encoder-decoder (Dai & Le, 2015) and results of its different version with different vocabulary expansion method. Comparing with best results, improvements of Pearsons r ($p < 0.002$) and MSE ($p < 0.001$) are significant based on students t-test.

Method	Pearsons r	Spearmans ρ	MSE
Illinois-LH	0.7993	0.7538	0.3692
UNAL-NLP	0.8070	0.7489	0.3550
Meaning Factory	0.8268	0.7721	0.3224
ECNU	0.8414	–	–
LSTM	0.8528	0.7911	0.2831
Bidirectional LSTM	0.8567	0.7966	0.2736
2-layer LSTM	0.8515	0.7896	0.2838
2-layer Bidirectional LSTM	0.8558	0.7965	0.2762
Mean vectors	0.7577	0.6738	0.4557
DT-RNN	0.7923	0.7319	0.3822
SDT-RNN	0.7900	0.7304	0.3848
Constituency Tree-LSTM	0.8582	0.7966	0.2734
Dependency Tree-LSTM	0.8676	0.8083	0.2532
Uni-skip	0.8477	0.7780	0.2872
Bi-skip	0.8405	0.7696	0.2995
Combine-skip	0.8584	0.7916	0.2687
Combine-skip+COCO	0.8655	0.7995	0.2561
Autoencoder	0.8284	0.7577	0.3258
Future predictor	0.8132	0.7342	0.3450
Hierarchical model	0.8333	0.7646	0.3135
Composite model	0.8434	0.7767	0.2972
Combine	0.8533	0.7891	0.2791
Hierarchical model + emb	0.8352	0.7588	0.3152
Composite model + emb	0.8425	0.7742	0.3005
Combine + emb	0.8554	0.7893	0.2789
Our Method			
Text Concept Vector	**0.8691**	0.8044	**0.2497**
Text Concept Vector with CBOW	0.8689	0.8044	0.2501

5.1.4. Results

The results of semantic relatedness on the SICK are shown in Table 1. The first group is the results of SemEval 2014 submissions. The second group is the results of sequential LSTMs. The third group and fourth group are the baselines reported by Tai et al. (2015) and the results of its own methods (i.e., Constituency Tree-LSTM and Dependency Tree-LSTM). The fifth group is the results of Skip-Thought Vectors (Kiros et al., 2015). The sixth group and seventh group are the results of CNN-LSTM encoder-decoder (Dai & Le, 2015) and results of its different version with different vocabulary expansion method. The last one is our own method and we report mean results over 5 runs, which is the same as previous works.

From the table we can see that our framework outperforms Skip-Thought Vectors, CNN-LSTM encoder-decoder and all the previous submissions in SemEval 2014 submissions. The CNN-LSTM encoder-decoder based methods do not outperform the sequential LSTMs models. It may be that these two kinds of methods capture same level features. The extension version of Skip-Thought Vectors (combine-skip + COCO) achieves remarkable result. It is because that it uses features derived from an image-sentence embedding model trained on the Microsoft COCO dataset (Lin et al., 2014). This external features help the original model to improve the performance, however, its result is still slightly worse than our method. This is because that our framework utilizes the text background concept information which is more appropriate for the text representation. Comparing with the state-of-the-art tree-LSTM methods, our framework obtains better result than the constituency tree-LSTM and performs about the same with the dependency tree-LSTM. The dependency tree-LSTM has higher Spearman's ρ while

Table 2
Statistical information of the datasets. #s/d, #w/d and #e/d indicate average number of sentences, average number of words and average number of entities in one document.

Corpus	Training samples	Test samples	Class	Vocabulary	#s/d	#w/d	#e/d
Yelp 2013	268,013	33,504	5	211,245	8.9	151.6	130.2
Yelp 2014	900,363	112,549	5	476,191	9.2	156.9	131.9
Yelp 2015	1,255,409	156,928	5	612,636	9.0	151.9	131.5
IMDB	280,593	34,029	10	115,831	14.02	325.6	268.0

Table 3
Run time of Yelp 2013, Yelp 2014, Yelp 2015 and IMDB in text conceptualization stage and text vector representation stage.

Corpus	Yelp 2013	Yelp 2014	Yelp 2015	IMDB
Text conceptualization stage	5.2 h	17.3 h	23.5 h	9.7 h
Text vector representation stage	2.3 h	8.2 h	10 h	2.4 h

our framework has higher Pearson's r and lower mean squared error. Improvements of Pearsons r ($p < 0.002$) and MSE ($p < 0.001$) are significant based on students t-test. It is due to that the dependency tree-LSTM produces dependency parses of each sentence, which makes the model more suitable for the task. Results of Text Concept Vector with CBOW illustrate that knowledge information of text is more important than the term order since CBOW doesn't consider order information.

5.2. Sentimental classification

For the document level task, our framework will be tested on sentiment document classification.

5.2.1. Datasets

Datasets of this experiment are IMDB reviews and Yelp Dataset Challenge.

1. IMDB Reviews. IMDB reviews dataset is obtained from Diao et al. (2014). Each review is marked with a rating (ranges from 1 to 10) which indicates the sentimental level. Reviews with higher scores tend to more positive.
2. Yelp Reviews. The Yelp reviews dataset comes from the Yelp Dataset Challenge in 2013, 2014 and 2015. We use the version provided by Tang et al. (2015a)!'£Reviews are labeled with 5 different levels of sentiment (the higher the score the more positve the review).

The statistical information of these two datasets are summarized in Table 2. The datasets are divided into training, development and testing sets with 80/10/10. Stanford CoreNLP is leveraged for sentence splitting and tokenization on all these datasets. The run time of these four datasets in text conceptualization stage and text vector representation stage is shown in Table 3.

5.2.2. Implementation details

For predicting the sentiment label of reviews, we use TCV to obtain the vector representation of each review and treat it as the input feature. Then we use a neural network to perform the classification. Fig. 3 shows the detail. Given a review vector $\boldsymbol{v_r}$, a fully connect layer is added to transform it to a real-valued vector $\hat{\boldsymbol{r}}$ whose length is same as the number of sentiment class labels. After that, a softmax layer is used to predict probabilities of each sentiment class label:

$$\hat{\boldsymbol{r}} = \boldsymbol{W}^{(r)}\boldsymbol{v_r} + \boldsymbol{b}^{(r)} \tag{17}$$

$$l_c = \frac{\exp(e(\hat{r}_c))}{\sum_{k=1}^{C}\exp(e(\hat{r}_k))}, c \in [1, C] \tag{18}$$

Here, C is the number of sentiment class labels and l_c is predicted probability of sentiment class label c.

The training objective is to minimize the categorical cross-entropy loss between the ground truth label distribution $l^g(r)$ and the predicted sentiment class label distribution $l(r)$.

$$J(\theta) = -\sum_{r\in M}\sum_{c=1}^{C} l_c^g(r)\log(l_c(r)) \tag{19}$$

where M is the total number of reviews in the training data and r represents a review. The ground truth label distribution $l^g(r)$ is one-hot represented scheme with ground truth being 1 and others being 0. The derivative of loss function is taken through back-propagation with respect to the whole set of parameters $\theta = \left[W^{(r)}; b^{(r)}\right]$ and parameters are updated by stochastic gradient descent.

The evaluation metric is classification accuracy (ACC):

$$ACC = \frac{\sum_{predicted_{l_i}=correct_{l_i}} 1}{N_{test}}, \tag{20}$$

where $predicted_{l_i}$ and $correct_{l_i}$ indicate the predicted label and correct label of ith review in test set. N_{test} means the total number of reviews in test set.

5.2.3. Baselines

Our framework will be compared to several methods for document level sentiment classification as follows:

1. Majority. Majority is a kind of heuristic method. In this method, the prominent sentiment label in the training set is assigned to each document in test set.
2. Unigrams and Bigrams. These two methods use bag-of-unigrams and bag-of-bigrams as features respectively and feed them into a SVM classifier.
3. TextFeature. TextFeature method (Kiritchenko, Zhu, & Mohammad, 2014) extracts several kinds of text features including character n-grams, word n-grams and sentiment lexicon features. Afterwards, a SVM classifier is trained for classification.
4. AverageSG. AverageSG first learns word embeddings of a document through the word2vec and then word embeddings are averaged to obtain the representation of the document, which is regarded as features and fed into a SVM classifier.
5. SSWE. This method generates sentiment-specific word embeddings (SSWE) (Tang et al., 2014)(SSWE) as features. After that, max/min/average pooling is utilized to obtain the document representation , and then a SVM is trained.
6. Paragraph Vector. The original distributed representation of documents (Le & Mikolov, 2014).
7. CNN. Directly uses the convolutional neural network for sentiment classification (Kim, 2014).

Table 4
Sentiment classification results of all methods. Results of Majority, Unigrams, Bigrams, SSWE, Paragraph Vector, CNN, Conv-GRNN and LSTM-GRNN are reported in Tang et al. (2015a). Results of Heterogeneous Embedded CNN and ISN are reported in Gui et al. (2017). Comparing with best results, improvements of Yelp2013 ($p < 0.0002$), Yelp2014 ($p < 0.0002$) and Yelp2015 ($p < 0.00015$) are significant based on students t-test.

Method	Yelp2013	Yelp2014	Yelp2015	IMDB
Majority (Tang et al., 2015a)	35.6	36.1	36.9	17.9
SVM + Unigrams (Tang et al., 2015a)	58.9	60.0	61.1	39.9
SVM + Bigrams (Tang et al., 2015a)	57.6	61.6	62.4	40.9
SVM + TextFeatures (Tang et al., 2015a)	59.8	61.8	62.4	40.5
SVM + AverageSG (Tang et al., 2015a)	54.3	55.7	56.8	31.9
SVM + SSWE (Tang et al., 2015a)	53.5	54.3	55.4	26.2
SVM + Paragraph Vector (Tang et al., 2015a)	57.7	59.2	60.5	34.1
CNN (Tang et al., 2015a)	59.7	61.0	61.5	37.6
Conv-GRNN (Tang et al., 2015a)	63.7	65.5	66.0	42.5
LSTM-GRNN (Tang et al., 2015a)	65.1	67.1	67.6	45.3
UPNN (Gui et al., 2017)	57.7	58.5	–	40.5
ISN (Gui et al., 2017)	62.3	63.5	–	47.6
Heterogeneous Embedded CNN (Gui et al., 2017)	65.6	66.2	–	**50.9**
Our Method				
Text Concept Vector	**67.8**	**69.2**	**71.5**	50.5

8. Conv-GRNN and LSTM-GRNN. These two models use a CNN or a LSTM to generate the sentence vectors, and then feed them into a recurrent neural network (GRNN) for producing the document representation (Tang et al., 2015a).
9. UPNN. UPNN (Tang et al., 2015b) incorporates user and product information into a convolutional neural network for to modeling documents.
10. Heterogeneous Embedded CNN. This method utilizes a heterogeneous network to model the shared polarity in product reviews and learn representations of users, products they commented on and words they used simultaneously (Gui et al., 2017).
11. ISN. Inter subjectivity Network Embedding (ISN) (Gui et al., 2016), uses a inter subjectivity network to obtain user representation and combines it with a CNN for document classification.

5.2.4. Results

Table 4 gives the sentiment classification results of all baselines. Results of Majority, Unigrams, Bigrams, SSWE, Paragraph Vector, CNN, Conv-GRNN and LSTM-GRNN are reported in Tang et al. (2015a). Results of Heterogeneous Embedded CNN and ISN are reported in Gui et al. (2017).

From the Table 3, we can see that method Majority gives the worst performances because it doesn't take any text features into consideration . Among the SVM classifier based methods, Unigrams, Bigrams and TextFeatures are the traditional features while AverageSG and SSWE are features generated by deep learning methods. However, the traditional features give better results than the features generated by the deep learning methods. What's more, unigram and bigram features are almost the strongest baselines. Original neural network based models without composition methods, such as CNN and Paragraph vectors, have little improvement for the document level sentiment classification. This may be caused by the fact that for the large scale text, bag-of-ngram features are no worse than simple neural network methods for the text representation.

Models leveraging composition methods can obviously improve the classification accuracy. Conv-GRNN and LSTM-GRNN use a hierarchical structure to model the document which is similar to the hierarchical structure of document. This setting helps the model better represent the document. The classification accuracy of Conv-GRNN and LSTM-GRNN are much higher than CNN and Paragraph vectors. UPNN, ISN and Heterogeneous Embedded CNN are other kinds of composition methods. Their models take the document related information into consideration such as writers of reviews and words they used. These external features enhance the classification performance of their methods. Heterogeneous Embedded CNN performs best on IMDB dataset. This may be because that IMDB has a large average number of words per review and this can help Heterogeneous Embedded CNN model to better capture words-writer relations for sentiment classification.

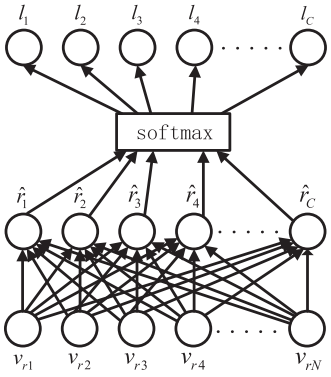


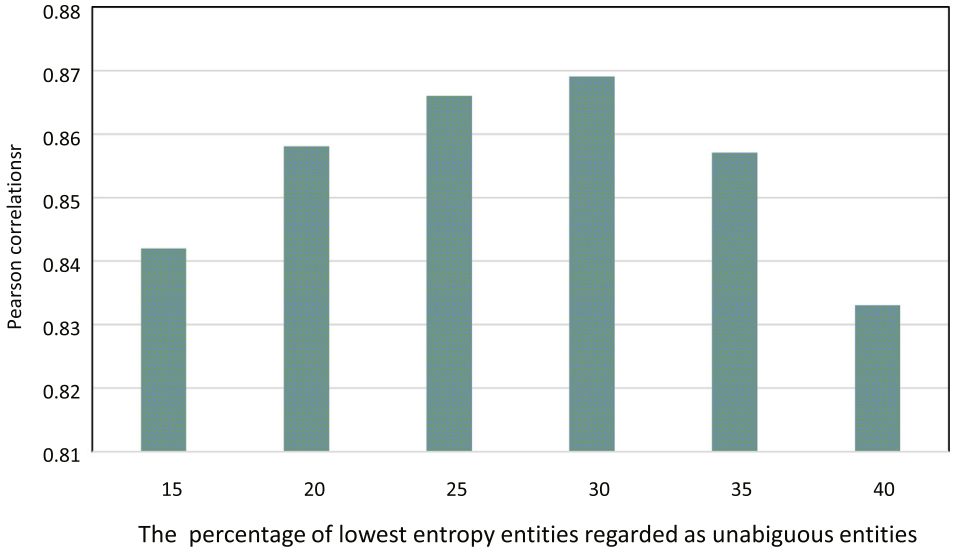
Fig. 6. The neural network used to perform the classification. The input is the review vector $\boldsymbol{v_r}$ whose length is N. Then a fully connected layer is used to transform the review vector to a real-valued vector $\hat{\boldsymbol{r}}$ whose length is same as the number of sentiment class labels C. After that, a softmax layer is used to predict probabilities of each sentiment class label l_c.

Comparing with above baselines, our proposed framework TCV achieves the best results on three datasets. Results of Heterogeneous Embedded CNN and ISN are reported in Gui et al. (2017). Comparing with best results, improvements of Yelp2013 ($p < 0.0002$), Yelp2014 ($p < 0.0002$) and Yelp2015 ($p < 0.00015$) are significant based on students t-test. The original Paragraph Vector method is same as TCV without knowledge base. Our TCV outperforms it obviously. It is because with the help of external knowledge base Probase, TCV can solve the semantic ambiguity in sentimental text. Results of TCV demonstrate the effectiveness of our framework.

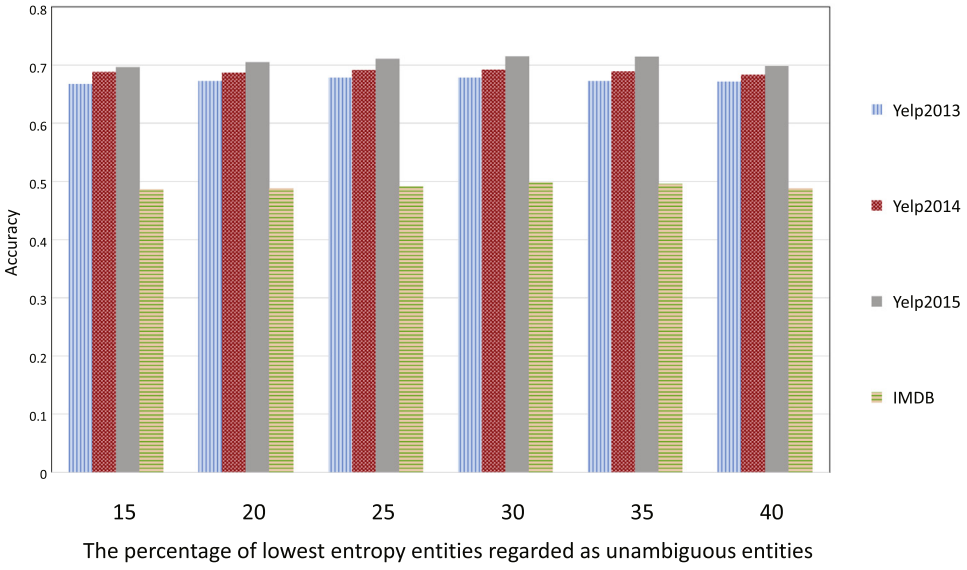
5.3. Parameter sensitivity

5.3.1. Effect of the percentage of lowest entropy entities

As we described in Section 4.1.3, one of the important step is using unambiguous entities to disambiguate vague entities. So, it is important to set the percentage of lowest entropy entities as unambiguous entities. In this experiment, we test the effect of setting the different percentage of lowest entropy entities as unambiguous entities. The results are shown in Fig. 7a and b. From figures



(a) Pearson correlations r of semantic relatedness with setting the different percentage of lowest entropy entities as unambiguous entities



(b) Classification accuracy of four datasets with setting different percentage of lowest entropy entities as unambiguous entities

Fig. 7. The effect of setting the different percentage of lowest entropy entities.

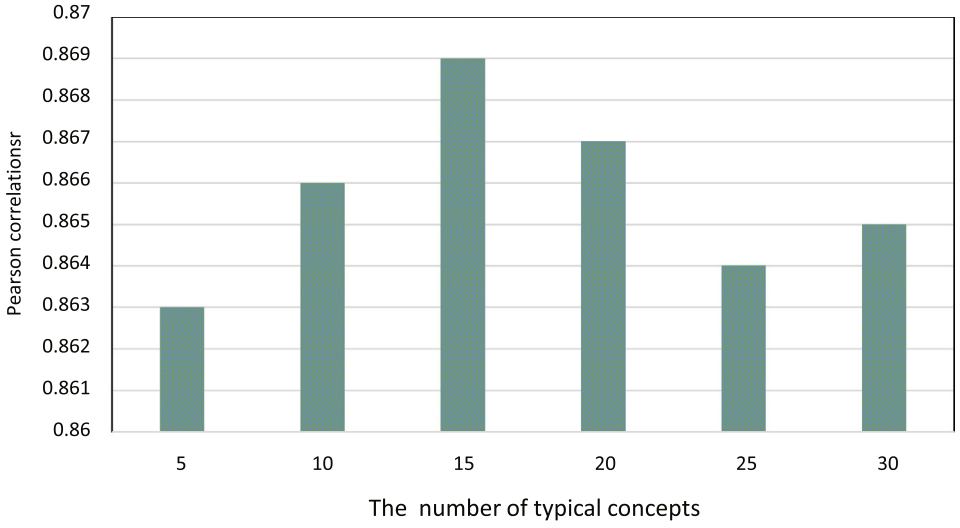
we can see that for both semantic relatedness task and sentiment classification, TCV achieves best result when percentage is set to be 25% to 30%. Performances of TCV decay rapidly when percentage is over 35%. This is because if we set large percentage of lowest entropy entities as unambiguous entities, vague entities will be included in them.

5.3.2. Effect of the number of typical concepts

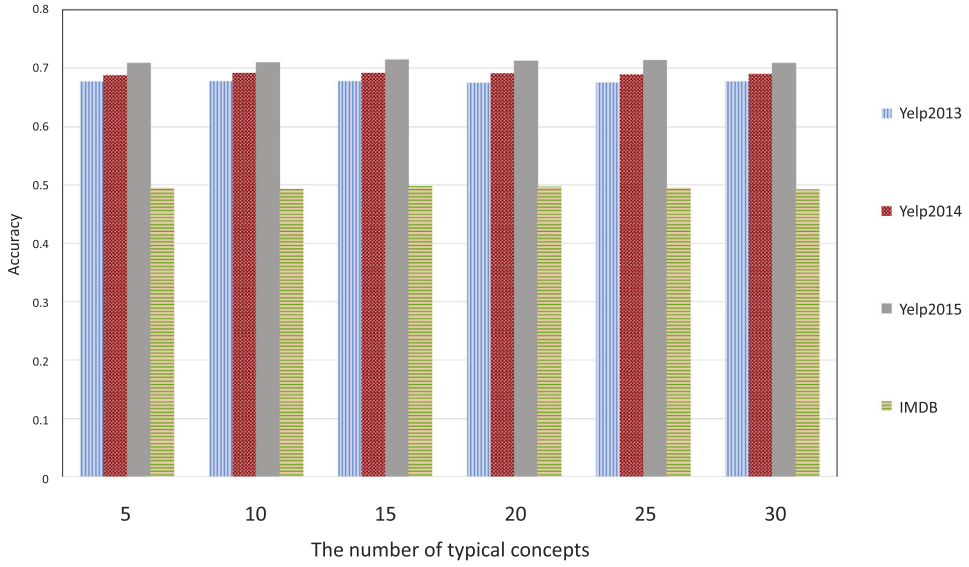
Another critical parameter for TCV is the number of typical concepts of an entity e_j (eg., N_t). This parameter can effect the concept cluster of the entity e_j which is important to determine the sense of the entity. Fig. 8a and b show the results. It can be seen from figures that the number of typical concepts has smaller influence on sentiment classification than on semantic relatedness. The sentiment classification results do not have much fluctuations. This is because the data used in sentiment classification are larger than the data used in semantic relatedness. Semantic relatedness obtains best results when the number of typical concepts is 15. Too many typical concepts will contain noise.

6. Conclusion and future works

We describe a knowledge-based framework (TCV) to obtain concept level of text. Distinct from the previous works, our framework does not simply focus on text itself which reflects limited information and is usually semantically ambiguous but includes concepts that are extracted from a external knowledge base. This framework utilizes an external knowledge base to encode the background knowledge and semantic information of text into representation. It first uses Probase to capture the concepts of text. Then a neural network is employed to generate the concept level representation of text. Our framework is applied to both sentence level task and document level task. For the sentence level task, we test our model by predicting the semantic relatedness of sentence



(a) Pearson correlationsr of semantic relatedness with different number of typical concepts



(b) Classification accuracy of four datasets with different number of typical concepts

Fig. 8. The effect of the number of typical concepts.

pairs. For the document level task, our model will be tested on sentiment classification. Experimental results show that our proposed framework achieves state-of-the-art performances on both tasks. We chose Probase as the running example, other knowledge bases also can be applied to our framework. For an example, one can use Wikipedia via Wikification to generate text concepts.

In the future, we will further modify our framework which will be more unified and elegant. What's more, due to the fact that text is constructed by human based on morphological and grammatical rules, it already contains well defined morphological and syntactic knowledge. We will try to integrate other kinds of knowledge (such as morphological, syntactic, and semantic knowledge) to represent text.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 61572434), China Knowledge Centre for Engineering Sciences and Technology (No. CKCEST-2017-1-2).

References

Bian, J., Gao, B., & Liu, T.-Y. (2014). Knowledge-powered deep learning for word embedding. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 132–148). Springer.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on management of data* (pp. 1247–1250). ACM.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Proceedings of the 28th international conference on neural information processing systems* (pp. 3079–3087). MIT Press.

Diao, Q., Qiu, M., Wu, C.-Y., Smola, A. J., Jiang, J., & Wang, C. (2014). Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 193–202). ACM.

Djuric, N., Wu, H., Radosavljevic, V., Grbovic, M., & Bhamidipati, N. (2015). Hierarchical neural language models for joint representation of streaming documents and their content. In *Proceedings of the 24th international conference on world wide web* (pp. 248–255). ACM.

Dong, L., Wei, F., Zhou, M., & Xu, K. (2014). Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *Proceedings of the twenty-eighth AAAI conference on artificial intelligence* (pp. 1537–1543). AAAI Press.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research, 12*, 2121–2159.

Franco-Salvador, M., Rangel, F., Rosso, P., Taulé, M., & Martít, M. A. (2015). Language variety identification using distributed representations of words and documents. In *International conference of the cross-language evaluation forum for European languages* (pp. 28–40). Springer.

Gabrilovich, E., & Markovitch, S. (2006). Overcoming the brittleness bottleneck using wikipedia: Enhancing text categorization with encyclopedic knowledge. In *Proceedings of the 21st national conference on artificial intelligence-volume 2* (pp. 1301–1306). AAAI Press.

Gui, L., Xu, R., He, Y., Lu, Q., & Wei, Z. (2016). Intersubjectivity and sentiment: From language to knowledge. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 2789–2795). AAAI Press.

Gui, L., Zhou, Y., Xu, R., He, Y., & Lu, Q. (2017). Learning representations from heterogeneous network for sentiment classification of product reviews. *Knowledge-Based Systems, 124*, 34–45.

Ganesh, J., Gupta, M., & Varma, V. (2016). Doc2sent2vec: A novel two-phase approach for learning document representation. In *Proceedings of the 39th international ACM SIGIR conference on research and development in information retrieval* (pp. 809–812). ACM.

Harris, Z. S. (1981). *Distributional structure* (pp. 3–22)). Springer Netherlands.

Hearst, M. A. (1992). Automatic acquisition of hyponyms from large text corpora. *Coling 1992 volume 2: The 15th international conference on computational linguistics*.

Hermann, M. K., & Blunsom, P. (2013). The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 894–904). Association for Computational Linguistics.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Huang, W., & Wang, J. (2016). Character-level convolutional network for text classification applied to chinese corpus. arXiv:1611.04358.

İrsoy, O., & Cardie, C. (2014). Deep recursive neural networks for compositionality in language. In *Proceedings of the 27th international conference on neural information processing systems* (pp. 2096–2104). MIT Press.

Johnson, R., & Zhang, T. (2015). Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 103–112). Association for Computational Linguistics. doi:10.3115/v1/N15-1011.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1746–1751). Association for Computational Linguistics. doi:10.3115/v1/D14-1181.

Kiritchenko, S., Zhu, X., & Mohammad, S. M. (2014). Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research, 50*, 723–762.

Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., et al. (2015). Skip-thought vectors. In *Proceedings of the 28th international conference on neural information processing systems* (pp. 3294–3302). MIT Press.

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the twenty-ninth AAAI conference on artificial intelligence* (pp. 2267–2273). AAAI Press.

Le, Q., & Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st international conference on international conference on machine learning-volume 32* (pp. II–1188). JMLR. org.

Li, J., Luong, T., Jurafsky, D., & Hovy, E. (2015). When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 2304–2314). Association for Computational Linguistics. doi:10.18653/v1/D15-1278.

Li, P., Wang, H., Zhu, K. Q., Wang, Z., & Wu, X. (2013). Computing term similarity by large probabilistic ISA knowledge. In *Proceedings of the 22nd ACM international conference on conference on information & knowledge management* (pp. 1401–1410). ACM.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755). Springer.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., & Zamparelli, R. (2014). Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th international workshop on semantic evaluation (semeval 2014)* (pp. 1–8). Association for Computational Linguistics. doi:10.3115/v1/S14-2001.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv:1301.3781.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th international conference on neural information processing systems* (pp. 3111–3119). Curran Associates Inc.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the ACM, 38*(11), 39–41.

Nakashole, N., & Flauger, R. (2017). Knowledge distillation for bilingual dictionary induction. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2487–2496).

Paulus, R., Socher, R., & Manning, C. D. (2014). Global belief recursive neural networks. In *Proceedings of the 27th international conference on neural information processing systems* (pp. 2888–2896). MIT Press.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, D. C., Ng, A., et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642). Association for Computational Linguistics.

Sproat, R., Gale, W., Shih, C., & Chang, N. (1996). A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics, 22*(3), 377–404.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago: A core of semantic knowledge. In *Proceedings of the 16th international conference on world wide web* (pp. 697–706). ACM.

Tai, S. K., Socher, R., & Manning, D. C. (2015). Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* (pp. 1556–1566). Association for Computational Linguistics. doi:10.3115/v1/P15-1150.

Tang, D., Qin, B., & Liu, T. (2015a). Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing* (pp. 1422–1432). Association for Computational Linguistics. doi:10.18653/v1/D15-1167.

Tang, D., Qin, B., & Liu, T. (2015b). Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* (pp. 1014–1023). Association for Computational Linguistics. doi:10.3115/v1/P15-1098.

Tang, D., Wei, F., Yang, N., Zhou, M., Liu, T., & Qin, B. (2014). Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1555–1565). Association for Computational Linguistics. doi:10.3115/v1/P14-1146.

Wang, F., Wang, Z., Li, Z., & Wen, J.-R. (2014). Concept-based short text classification and ranking. In *Proceedings of the 23rd ACM international conference on conference on information and knowledge management* (pp. 1069–1078). ACM.

Wei, X., Zhang, J., Zeng, D. D., & Li, Q. (2016). A multi-level text representation model within background knowledge based on human cognitive process for big data analysis. *Cluster Computing, 19*(3), 1475–1487.

Wu, W., Li, H., Wang, H., & Zhu, K. Q. (2012). Probase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data* (pp. 481–492). ACM.

Yao, L., Zhang, Y., Chen, Q., Qian, H., Wei, B., & Hu, Z. (2017). Mining coherent topics in documents using word embeddings and large-scale text data. *Engineering Applications of Artificial Intelligence, 64*, 432–439.

Yao, L., Zhang, Y., Wei, B., Jin, Z., Zhang, R., Zhang, Y., et al. (2017). Incorporating knowledge graph embeddings into topic modeling. In *AAAI* (pp. 3119–3126).

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Proceedings of the 28th international conference on neural information processing systems* (pp. 649–657). MIT Press.

Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A c-lstm neural network for text classification. arXiv:1511.08630.