Knowledge-Based Systems 182 (2019) 104842

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Learning short-text semantic similarity with word embeddings and external knowledge sources

Hien T. Nguyen [a], Phuc H. Duong [b], Erik Cambria [c,*]

[a] Faculty of Information Technology, Ho Chi Minh City University of Food Industry, Ho Chi Minh City, Viet Nam
[b] Artificial Intelligence Laboratory, Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Viet Nam
[c] School of Computer Science and Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 22 August 2018
Received in revised form 7 July 2019
Accepted 10 July 2019
Available online 25 July 2019

Keywords:
Paraphrase identification
Sentence similarity
Short text similarity
Semantic textual similarity

ABSTRACT

We present a novel method based on interdependent representations of short texts for determining their degree of semantic similarity. The method represents each short text as two dense vectors: the former is built using the word-to-word similarity based on pre-trained word vectors, the latter is built using the word-to-word similarity based on external sources of knowledge. We also developed a preprocessing algorithm that chains coreferential named entities together and performs word segmentation to preserve the meaning of phrasal verbs and idioms. We evaluated the proposed method on three popular datasets, namely Microsoft Research Paraphrase Corpus, STS2015 and P4PIN, and obtained state-of-the-art results on all three without using prior knowledge of natural language, e.g., part-of-speech tags or parse tree, which indicates the interdependent representations of short text pairs are effective and efficient for semantic textual similarity tasks.

1. Introduction

Measuring text semantic similarity is about determining the degree of semantic similarity between pairs of texts. Such measures of text semantic similarity have attracted much research attention in natural language processing (NLP), natural language understanding, and information retrieval communities [1–5]. The earliest methods were proposed to measure similarity between long texts, e.g., for document categorization, or between a short text and a long one, e.g., for a query and a document in a search system. The most convenient way to represent a document (and query) by those bag-of-words (BOW) methods is a vector space model where each document is considered as a BOW and represented by a vector. Cosine similarity, known as a traditionally vector-based measure used in information retrieval, is a popular measure to assess similarity between texts.

Recently, there have been emerging tasks that take advantage of short-text semantic similarity (STSS) and require to assess the degree of similarity between sentences or text snippets. Some example tasks getting benefit from STSS are question answering [6], short answer scoring [4], machine translation evaluation [7], plagiarism detection [3], sentiment analysis [8], text summarization [9], schema matching [10], etc.

STSS, however, is a difficult task. For instance, consider two sentences S_1 and S_2 below, they are similar in meaning but do share a few words, i.e., *trading*, *the*, *in*, *stock*, *Nasdaq*, and *Market*.
S_1: Comcast Class A shares were up 8 cents at $30.50 in morning trading on the Nasdaq Stock Market.
S_2: The stock rose 48 cents to $30 yesterday in Nasdaq Stock Market trading.

One can apply a BOW model to measure similarity between pairs of short texts. However, such a model is inefficient in computation since each text is represented by a vector in the high-dimensional vector space. In [11], the authors pointed out that BOW methods often ignore the important relationships in sentence structure, such as semantic and syntactic dependencies between words. For instance, they ignore function words, which are helpful in interpreting the meaning of texts, such as some transition words, e.g., *because of*, *as a result* do play an important role in analyzing discourse elements [12]. In addition, when contextual information is insufficient, particularly in the case of short texts, the BOW model suffers from the sparsity problem.

In order to overcome the drawbacks of BOW methods, many approaches have been proposed for STSS using lexical matching [13–16], syntactic information [5,8,11–13,16–19] in combination with lexical semantics. Lexical semantics is the study of word meanings in Linguistics. In NLP, approaches to lexical semantics include word-to-word similarities based on distributional representations of words or external sources of structured semantic knowledge such as WordNet. Some others make use of query expansion techniques by using Web search engines to enrich the

* Corresponding author.
E-mail addresses: hien@hufi.edu.vn (H.T. Nguyen), duonghuuphuc@tdtu.edu.vn (P.H. Duong), cambria@ntu.edu.sg (E. Cambria).

https://doi.org/10.1016/j.knosys.2019.07.013
0950-7051/

short texts to avoid the sparsity [20]. This work exploits lexical semantics using word embeddings (pre-trained word vectors) and external sources of knowledge.

To obtain good lexical semantics, word representations have become increasingly popular to capture the meaning of words [2]. Distributional representation are based on a word-document (or context) co-occurrence matrix F of the size $W \times C$, where each row of F is an initial representation of a word, W and C are the vocabulary size and the number of documents [21], respectively. A matrix factorization method such as singular value decomposition (SVD) can be used to reduce F from $W * C$ to $W * d$ where $d \ll C$. Notably distributional representation techniques are used in [11,13], and [22]. While [11] implemented latent semantic analysis (LSA) [23] and calculated STSS based on the similarity between words whose vectors are represented in a lower dimensional space, [13] and [22] proposed weighting schemes to re-weight features before factorizing the matrix.

Learning distributed representations of words, termed word embeddings, is another approach to word representations. Word embeddings models such as word2vec [24], Glove [25] or ELMo [26] have recently had tremendous success in many NLP tasks. Moreover, neural networks employed for STSS using word embeddings as pre-training have proved to achieve good performance [26,27]. The word embedding approaches are then generalized to coarser granularities, *i.e.*, phrase-level, sentence-level, or paragraph-level, which are usually considered as sentence embeddings or sentence representations in common [1,28–36].

One can categorized approaches to sentence representations as task-specific and general-purpose. While task-specific representations such as RAE [19] (henceforth *task-specific*) are for a specific task and untransferable, general-purpose representations are transferable by fine-tuning to adapt the parameters such as GenSen [36], OpenAI GPT [35] and BERT [1] or using as features such as Skip-thoughts [31], InferSent [33] in a downstream model. We call the former *fine-tuning* and the latter *feature-based*. Note that from now on if an approach to STSS is not task-specific, fine-tuning, or feature-based, we call it *feature engineering* in the mean that exploits lexical and syntactic information to obtain separate features for representing sentences other than learning sentence representations as a whole.

Even though many approaches have been proposed for STSS, the state-of-the-art method presented in [13] achieves not so high accuracy 80.4% on Microsoft Research Paraphrase (MRPC) corpus [37]. An other presented in [22] achieves 86.6% accuracy on MRPC but the model was trained using transductive setting instead of the inductive setting as in other state-of-the-art methods. However, when trained in the inductive setting, it achieves 68.7% accuracy on MRPC. Recently, pre-trained sentence representations, in particular BERT [1], have been proved to create breakthroughs in a wide range of NLP tasks when fine-tuned with just one additional output layer and achieves state-of-the-art results on MRPC with 89.3% accuracy, superior to all other approaches to STSS in literature. However, as reported in [1], BERT needs extremely vast resources to train. Indeed, $BERT_{LARGE}$ was trained on 16 Cloud TPUs (64 TPU chips total) and took 4 days to complete, putting it out of reach of low-resource researchers.

Nevertheless, there is meaningful information for measuring semantic textual similarity that has not been exploited yet. In particular, back to the above example, we can see that there is much information which we need to capture before computing the similarity. For instance, named entities (NEs), e.g., *"Comcat Class A"*, *"Nasdaq Stock Market"*, should be recognized, since they will add noise to the measurement if we consider every token of them separately. Besides, the similar entities in S_1 and S_2, *"Nasdaq Stock Market"*, should also be chained together. However, the proposed methods in literature did not take coreference among entities into account. In addition, the tokens that constitute the meaning of a word are also important. For short English texts, much previous work considers each token as a word; but, that is not always true. For instance, in English grammar, *"pull out"* is a phrasal verb and has the same meaning with *"extract"*. If separating *"out"* from *"pull"*, we lose the word *"pull out"* and lose the chance to capture similarity between *"pull out"* and *"extract"* when they occur in two given texts.

In this paper, we propose a novel method for STSS and the contribution is three-fold. First, we represent interdependence between short texts in a pair based on lexical semantics using word embeddings and external sources of knowledge, i.e., Wiktionary[1] and WordNet.[2] The interdependent representations are used to compute features that then input to a support vector machine (SVM) to identify pairs as being paraphrases or not, or determine their degree of semantic similarity. By representing the interdependence between short texts, we model their interaction in the term of semantics. Second, our method exploits coreference among entities and performs *word segmentation* to preserve the meaning of phrasal verbs and idioms. Coreference among entities and word segmentation are performed in a *preprocessing step*. Note that the proposed method in [18] also exploits named entities (NEs) but differently in that it uses NE tags as features, ours takes coreference relationship among NEs into account. Finally, we conduct experiments to evaluate our method on three popular datasets —MRPC, STS2015[3] and P4PIN corpus[4] [38] —to show its effectiveness. The experiments are set up to show the contribution of the interdependent representations of short texts, as well as the preprocessing step. Experimental results show that the proposed method achieves comparable performance in three datasets MRPC, STS2015, and P4PIN. Overall, the findings are that the interdependent representations between short texts as proposed in this work significantly improve the performance of paraphrase identification. We note that our method can be considered as an improvement upon [39] and [40] in that we employ the same working process with [40] but using new features, one of which is based on [39].

The rest of this paper is organized as follows: Section 2 presents related work; Section 3 illustrates our proposed method; Section 4 describes datasets and experiments in detail; finally, Section 5 concludes the paper.

2. Related work

In this section, we review previous work relevant to aspects of our proposed method. De facto, a BOW method can be used to measure the similarity between pairs of short texts. However, experimental results in [11] showed that it produces poor prediction accuracy. In order to overcome the drawbacks of the BOW approach, many methods have been proposed for STSS such as exploiting lexical matching and syntactic information; learning distributional or distributed representations of words, as well as sentences. Lexical matching can be edit distance [13], lexical overlap [14], BLEU score [15], or string similarity [16]. Syntactic information can be word order [16,17], part-of-speech (POS) tags [11,18], discourse elements [12], parse trees [13,19], tree edit distance [5], or dependency grammars [18]. Distributed representations can be word embeddings [24,25], task-specific sentence representations [19] or general-purpose sentence representations [1,30–32,35,36].

[1] https://www.wiktionary.org/.

[2] https://wordnet.princeton.edu/.

[3] http://alt.qcri.org/semeval2015/task2/.

[4] Available at: http://ccc.inaoep.mx/~mmontesg/resources/corpusP4PIN.zip.

In order to easy following the rest of this section, keep in mind that when mentioning accuracy of a method, as well as a model or classifier, we mean the method achieves the accuracy on MRPC dataset; otherwise, we point out which dataset was used to evaluate the method.

In [11], STSS was determined by summing of WordNet-based similarities between words. In [17], the authors presented a method that exploits both similarity between words and word order information of two given sentences. They represented each sentence as two vectors: a semantic vector and a word-order vector. The length of each vector is equal to the size of the joint word set of two sentences. An entry of a semantic vector is the semantic similarity determined based on WordNet between a word in a sentence to a word in another one. An entry of a word-order vector is the position of a word in the corresponding sentence. Because different words have different contributions to the meaning of a sentence, each word is weighted by using information content based on Brown Corpus of American English.[5] Cosine between two semantic vectors or between two word-order vectors is considered as a feature. Overall STSS is a linear combination two features. The method in [16] improves [17] by taking string similarity (e.g., longest common subsequence) into account. In contrast, we train SVM models other than linear combination of features as methods presented in [11,16,17]. Moreover, we do preprocessing to obtain higher quality features. The work that is most relevant to ours is presented in [14], where the Jaccard coefficient and the Levenshtein edit distance are adapted by integrating the similarities between words based on word embeddings. Then, the coefficient and the distance are used as two features to train a J48 classifier. The authors reported the classifier achieves 83% and 93% accuracy on MRPC and P4PIN, respectively.

Having recognized that discourse structure may significantly constitute the meaning of a sentence, the proposed method in [12] divides a sentence into elementary discourse units (EDUs) and measures STSS based on the similarity between their elementary discourse units. The authors showed that the similarity between EDUs plays an important role in paraphrasing. However, recognition of EDUs is difficult due to the fuzziness of their boundaries and its dependence on parse trees of the input texts. Therefore, it is infeasible to adapt to low-resource languages. In [15], the authors evaluated eight different machine translation metrics for identifying paraphrase. They trained an ensemble classifier (i.e., using three constituent classifiers: logistic regression, SVM, and an instance-based classifier) that takes as input the metrics, each of which is considered as a feature. Experimental results show that this method achieves 77.4% accuracy. In [18], the authors presented a generative model that incorporates features consisting of both syntax and lexical semantics using quasi-synchronous dependency grammars. Those features are extracted from WordNet, a named entity recognizer, a POS tagger, and the dependency labels from the aligned trees. Then, they combine the model with a logistic regression classifier built from n-gram overlap features. They reported the model achieves 76.1% accuracy.

Representations of short texts and words as dense vectors, i.e., distributional representations and word embeddings, has become a direction of research of STSS in literature. LSA were implemented in [11,13,22,41]. While [22] and [11] sum up the latent vectors of all the constituent words re-weighted by a tf.idf weighting scheme, [41] and [13] showed that the performance is significantly improved by re-weighting features differently. In particular, the discriminative re-weighting scheme, termed TF-KLD, including the term frequency and the KL-divergence, is proposed in [13] to re-weight elements of the term-context matrix before factorization to obtain latent representations of short texts. TF-KLD increases discriminatively distributional features and decreases the others. The latent representations of a text pair is converted into a so-called sample vector that then fed into SVM to train a classifier. The authors reported the classifier achieves 80.4% accuracy when bigrams and dependency pairs are used as distributional features in combination with fine-grained features. [22] proved that LSA in combination with Abstract Meaning Representation (AMR) parsing significantly improves state-of-the-art results in paraphrase detection. Given two sentences, first, their AMR parsing graphs are merged; then PageRank's values of the nodes on the merged graphs are used to re-weight elements of the term-context matrix. The authors reported the classifier achieves 86.6% accuracy in transductive setting. In this work, to obtain easily adaptable models for other languages, especially lowresource languages, we avoid exploiting linguistic information like those used in [11,13,15,18,22,41].

Word embeddings have been extremely successful in numerous NLP tasks. Many researchers successfully exploited word embeddings for learning sentence representations for modeling sentence pairs, as well as for STSS [1,19,28–32,36,42–44]. The recursive autoencoders based on unfolding objective proposed in [19] takes as input word vectors to learn feature vectors for phrases in two given sentences based on their parse trees. Then, the Euclidean distances between all word and phrase vectors of the sentences are calculated to form a similarity matrix. A novel dynamic pooling layer was proposed to transform the similarity matrix to a fixed-size one that will be fed into a softmax classifier to identify pairs as being paraphrases or not. The authors reported their model achieves 76.8% accuracy. The model proposed [45] showed that dissimilar parts of input sentences usually give some meaningful clues for accessing the degree of semantic similarity.

In [44], the authors investigated two CNN-based models, namely ARC-I and ARC-II, to learn representations of sentence pairs for matching and showed the effectiveness of modeling the pairs on the interaction space. In ARC-I, first, two sentences are separately transformed into fixed length vectors using multiple convolutional layers; then these vectors are concatenated. In ARC-II, an interaction representation of a sentence pair is obtained thorough "one-dimensional" convolutions by striding windows to capture all possible combinations of the two sentences. The interaction representation is continuous transformed using 2D-convolutions in following layers into a sole fixed length vector. Both ARC-1 and ARC-2 input the output vector of the CNN into a multi-layer perceptron network (MLP) to learn a binary classifier. The authors reported ARC-2 model achieves 69.9% accuracy.

BI-CNN-MI [28] separately learns the representation of each sentence by a CNN with multilayers and calculate similarity matrices of two sentence at four levels, each of which respectively represents the similarity between unigrams, short n-grams, long n-grams of the sentences and between the sentences. A similarity matrix at a level is built based on the Euclidean distance, followed by a dynamic pooling layer transforming it into a feature that then input to a logistic classifier to perform paraphrase detection.

ABCNN-3 [43] improves BI-CNN-MI in that it replaces the similarity matrices by attention ones. The role of the attention matrices is to capture the interdependence between two sentences range from word to phrase levels. The values of the attention matrices are the matching scores between values in the feature maps of the convolution layers using the Euclidean distance. Note that ABCNN-3 calculates attention matrices for both the input and output of a convolution layer with pooling, where the input stacks both the conventional feature map and attention feature map obtained by multiply the corresponding attention matrix with a parameter matrix learned during training. BI-CNN-MI and ABCNN-3 achieve 78.1% and 78.9% accuracy, respectively.

[5] http://www.nltk.org/nltk_data/.

In [46], the authors proposed a CNN based model with two main layers: sentence model and similarity measurement. The former learns sentence representations to capture different granularities of information of the input texts using multiple types of pooling and multiple convolution filters. The latter compares local regions of the sentence representations using multiple similarity measurements. The model achieves 78.6% accuracy. [42] combines CNN and LSTM to extract three kinds of features, then input them to a MLP with three layers to learn a classifier. The proposed model achieves 77.7% accuracy.

The word embedding approaches are also generalized to general-purpose sentence representations [1,28–36]. Among them, some representation models can be used as features such as Skip-thoughts [31], InferSent [33] or fine-tuned to adapt the parameters such as GenSen [36], OpenAI GPT [35] and BERT [1] in a downstream model.

In [29], the authors presented models that learn sentence embeddings for paraphrase identification. The simplest model generates the embedding vector of a sentence by averaging its word vectors. The second model learns projection in addition to the word embedding. The authors show that the generalization of these simple models is better than that of long short-term memory (LSTM) networks, deep averaging network and identity recurrent neural networks (RNN) and achieves the average of 75.8 Pearson's $\rho \times 100$ on SemEval 2015.

Skip-thoughts [31] is an unsupervised learning model of a distributed sentence encoder. Given a triple of continuous sentences s_{i-1}, s_i, s_{i+1}, the skip-thoughts model encodes the sentence s_i and tries to reconstruct its previous sentence s_{i-1} and the next sentence s_{i+1}. The skip-thoughts model is treated as an encoder–decoder model where both encoder and decoder are gated recurrent units (GRUs) [47]. Due to [31] limited the order of sentences, [30] improves the skip-thoughts model by using a sequential denoising autoencoder (SDAE) that can be trained on a set of sentences in arbitrary order and achieves 76.4% accuracy. Sent2Vec [32] is an unsupervised model for learning sentence embeddings using the word vectors along with n-gram embeddings. The supervised evaluation of Sent2Vec showed that it achieves 72.5% accuracy when using both unigrams and bi-grams.

The quick-thoughts in [34] is a kind of encoder–decoder based sequence models, but the decoder is replaced by a classifier which chooses the target sentence from a set of candidates. Source and candidate sentences are encoded by GRUs. The model achieves 76.9% accuracy. InferSent in [33] includes three components. The first one looks up pre-trained word vectors for each sentence. The second one consists of a BiLSTM, followed by a max-pooling layer. The output at a time step t, h_t, is the concatenation of a forward LSTM and a backward LSTM. The max-pooling layer in turn select the maximum value of each h_t to form a vector representing the input sentence. Given two sentences, after passing through the first and second components, the third takes as input the concatenation of the sentence vectors, their element-wise product and absolute element-wise difference; then the result vector is fed into a classifier consisting of multiple fully-connected layers culminating in a softmax layer. The authors show that the obtained transfer model achieves 76.2% accuracy.

In [35], the authors proposed a method combining unsupervised pre-training and supervised fine-tuning to learn a universal representation that transfers to a wide range of diverse tasks in natural language understanding with little adaptation. The proposed model architecture is based on the Transformers [48]. In particular, the authors employ a multi-layer Transformer decoder [49] to train a language model, followed by supervised fine-tuning to adapt the parameters for the supervised target tasks. Experimental results show that the proposed method is state-of-the-art. The multi-task learning framework in [36] combines the inductive biases of multiple training objectives of diverse NLP tasks into a single model to learn sentence representations generalizing across the tasks. The proposed model is a one-to-many multi-task sequence-to-sequence learning model that includes a single recurrent encoder shared across tasks and task-specific decoders for the target tasks. The authors reported the model trained on 100 million sentences with the training objectives of skip-thought next, French translation, German translation, natural language inference, large model, 2-layer large model, skip-thought previous and constituency parsing tasks is state-of-the-art and achieves 78.6% accuracy. While the language representation models in [35] and [36] are unidirectional, BERT proposed in [1] is a deep bidirectional representations. The model architecture of BERT is a multi-layer bidirectional Transformer encoder that learns representations jointly conditioned on both left and right context in all layers. BERT is pre-trained using two unsupervised prediction tasks: Masked language model and Next sentence prediction. In particular, the authors randomly mask a certain percentage of tokens (i.e., 15%) in the corpus and train a language representation model to predict only those masked tokens. Next sentence prediction is trained to predict if the second in a sentence pair is the next one of the first or not. The trained model is then fine-tuned for target tasks. The authors reported the model obtains state-of-the-art results on eleven NLP tasks and achieves 89.3% accuracy on MRPC dataset.

We represent interdependence between short texts using lexical semantics based on word embeddings and external sources of knowledge. The interdependent representations are encoded as dense vectors based on the joint word set of two short texts under consideration. The size of a vector is equal to the number of distinct words in two input texts. In other word, our method takes advantages of word-to-word similarity based on the word context built by using word embeddings and semantic relatedness based on external sources of knowledge. It learns SVM models that takes as input cosine similarities between vectors representing input texts and accepts the input texts with variable length.

3. Proposed method

In this section, we present our proposed method. The overall working process shown in Fig. 1 consists of training and testing phases, each of which contains two main modules: preprocessing and feature extraction. The training phase includes a step where one can use any supervised learning algorithm to train a model for accessing the degree of semantic similarity of text pairs. In this work, we train SVM models using two features based on cosine similarities between dense vectors representing the input texts. The preprocessing and feature extraction modules will be presented in Sections 3.1 and 3.2, respectively.

3.1. Preprocessing

A sentence often contains special characters, e.g., colon, dash, that do not contribute much information in measuring the similarity [17]. Therefore, first, the preprocessing is to remove all special characters, but still maintains the structure of input texts.

Second, the preprocessing performs named entity recognition (NER) and NE coreference resolution. Considering two concepts *"Artificial Intelligence"* and *"AI"* in the context of *"Computer science"*; they are similar in meaning but do not share any term in common. Or *"The Pentagon"* is similar to *"United States Department of Defense"*, though they are different in words. Another important feature of NE is *alias*, that means, a NE can be represented by many other aliases. For example, consider *"IBM"* in the *"technology company"* context, at least two aliases different from words but similar in meaning are *"International Business Machines*

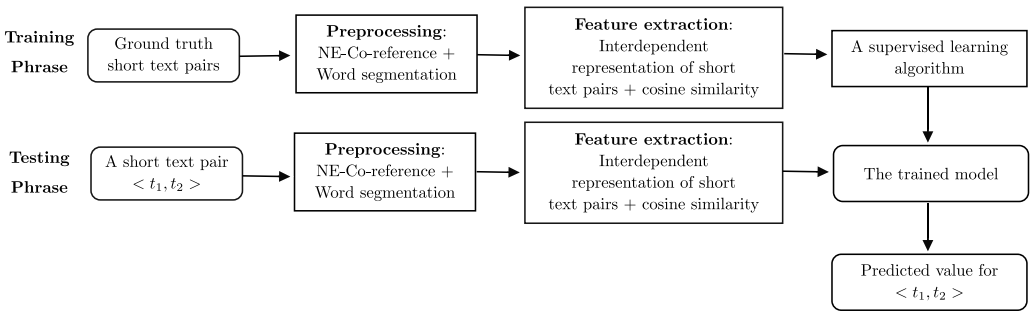

Fig. 1. The overall working process for short-text semantic similarity.

Corporation" and *"Big Blue"*. In other case, the *"United States"* has aliases: *"USA"*, *"U.S."*, *"United States of America"*. By recognizing aliases of a named entity, we can chain them together if they appear in two short texts and assign each coreference chain an identifier (ID). We use Stanford CoreNLP[6] for NER and coreference resolution with default setting. Since the Stanford CoreNLP achieves low performance, we propose to exploit Wikipedia[7] to extract alias of entities using their redirect pages and use OrthoMatcher rules in [50] to adjust some coreference chains produced by the Stanford CoreNLP. For instance, given two sentences S_3 and S_4, after resolving coreference NEs, coreference chains {*"Justice Dept"*$_1$, *"DOJ"*$_2$}, {*"Trump"*$_1$, *"Trump"*$_2$} are identified and each of which is assigned an ID.

S_3: Justice Dept says Trump to replace travel ban order in near future.

S_4: Trump promises new immigration order as DOJ holds off appeals court.

Finally, the preprocessing performs word segmentation to group tokens of the same phrasal verbs or idioms. For instance, in the case of *"blow up"* and *"explode"*, when considering *"blow up"* as two distinct tokens, it will not express the real meaning since *"blow up"* is a phrasal verb of *"explode"*. Besides, the English idioms is also recognized in the preprocessing. For example, *"baker's dozen"* and *"thirteen"* do not share any token in common, but they are similar in meaning. Therefore, word segmentation to recognize phrasals help maintain the meaning of original texts. In practice, we exploit Wiktionary[8] to group the tokens of recognized phrases together by looking up Wiktionary using *n*-grams. For instance, after performing word segmentation, the phrasal verb*"holds off"* in the sentence S_4 will become*"holds_off"*.

In summary, the preprocessing performs removing special characters, NER, NE coreference resolution, and word segmentation. Therefore, after the preprocessing, the sentences S_3 and S_4 are as follows:

S_3^{ws}: ID_1 says ID_2 to replace travel ban order in near future.

S_4^{ws}: ID_2 promises new immigration order as ID_1 holds_off appeals court.

3.2. Feature extraction

In this section, we introduce interdependent representations of short text pairs based on word embeddings and external sources of knowledge. We also present how to calculate features used to train SVM models for measuring semantic similarity between text pairs. Note that with this interdependent representations of input texts, we obtain vectors that are dense, low-dimensional, and real-valued. In particular, the size of a vector is the total number of all distinct words in two input texts under consideration.

3.2.1. Interdependent representation based on word embeddings

In [51], the authors present two word representation models: (1) continuous BOW (CBOW), and (2) continuous skip-gram. The difference between these two models is on the input and output. For CBOW, the model can predict the output word given its context, and otherwise, skip-gram model can predict the context from a given word. In both model, each word is represented by a one-hot vector and the output layer is softmax. Based on the approach of the skip-gram model, we can figure out that two words are similar when their contexts are similar. Given a word, we lookup its pre-trained vector using word2vec[9] [24] and build its context by searching for top k most similar words using the distance tool of word2vec. In this work, k is set to 20 in experiments.

Let wc_1 and wc_2 be the contexts of two word w_1 and w_2 respectively, the similarity between w_1 and w_2 is computed by the following formula:

$$Sim(w_1, w_2) = \frac{|wc_1 \cap wc_2|}{max(wc_1, wc_2)} \quad (1)$$

Let consider S_3^{ws} and S_4^{ws} above-mentioned, their vector representations are as follows. First, each word in S_3^{ws} and S_4^{ws} is normalized by applying stemming algorithm, i.e., transforming "promises" to "promise" using the Porter Stemming Algorithm (PSA).[10] If the result stem is not contained in WordNet,[11] we try another possible stem and if it is actually contained in Wordnet, we choose the new stem. Second, we construct a joint word set S which contains all distinct words from them. In particular, for the case S_3^{ws} and S_4^{ws}, S = {ID_1, say, ID_2, to, replace, travel, ban, order, in, near, future, promise, new, immigration, as, hold_off, appeal, court}. Finally, we form two dense vectors, denoted by V_1 and V_2, of which the dimension is the size of S. Each entry value of a dense vector is the word-to-word similarity calculated using Eq. (1). These dense vectors are built based on the following rules:

- Case 1: if a word w in S appears in S_i, where $i \in \{1, 2\}$, then assign $v_w = 1$.
- Case 2: if w does not appear in S_i, we calculate the similarity between w in S and each word in S_i. Then, we take the result which is highest and exceeds the preset threshold h, otherwise, $v_w = 0$. By defining a threshold, we can handle the case when v_w is very small, thus it should be assigned by 0.

After constructing two dense vectors, we compute cosine between these vectors and use it as the first feature of the SVM

6 https://stanfordnlp.github.io/CoreNLP/.
7 https://en.wikipedia.org/.
8 https://en.wiktionary.org/.
9 https://code.google.com/archive/p/word2vec/.
10 http://snowball.tartarus.org/algorithms/porter/stemmer.html.
11 http://wordnet.princeton.edu.

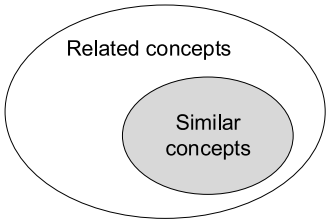

Fig. 2. Relatedness and similarity concepts [52].

models. Since the result from Eq. (2) is between 0 and 1, thus, two sentences are similar when the result is near 1, otherwise.

$$Sim_{F1}(S_1, S_2) = \cos(\theta) = \frac{V_1 \cdot V_2}{\|V_1\| \cdot \|V_2\|} \tag{2}$$

3.2.2. Interdependent representation based on external knowledge

In this Subsection, we introduce knowledge-based and corpus-based approaches in measuring the relatedness between two concepts. The term *"relatedness"* is more general than *"similarity"*, as presented in Fig. 2. For example, *"car"* and *"wheel"* are not similar, but there is a *part-of* relationship between them.

Let c_1 and c_2 be two given concepts, for measuring the relatedness between them, we apply Rel_{Res95} [53] based on WordNet and the Brown Corpus of American English. In [54], the authors presented a Path Length method, which takes into account the distance of (c_1, c_2) in the taxonomy. This approach, however, may lead to the unreliability [53]. To overcome this problem, we consider the probability of concepts subsuming both (c_1, c_2). This is based on information theory argumentation, the *information content* of a concept is computed by the negative log likelihood, $-\log pr(c)$, thus, two concepts are similar when they share more information determined by the subsume concepts. Eq. (3) presents the formula, where $LCS(c_1, c_2)$ is the set of *least common subsumer* concepts of (c_1, c_2).

$$Rel_{Res95}(c_1, c_2) = \max_{c \in LCS(c_1, c_2)} [-\log pr(c)] \tag{3}$$

A problem in exploiting WordNet is that it does not cover latest concepts, e.g., *"emoji"*, *"selfie"*. Therefore, when a concept does not exist in WordNet, we combine Wiktionary[12] and gloss-based approach [55]. Based on the definitions of concepts, each of which is called a *gloss text*, the method in [55] measures the relatedness between concepts as the intersection between their gloss texts. Formally, Eq. (4) defines the measurement.

$$Rel_{Lesk86}(c_1, c_2) = gloss(c_1) \cap gloss(c_2) \tag{4}$$

Eq. (5) is an extended version of Eq. (4), which normalizes the result in range of [0, 1].

$$Rel_{Lesk86}(c_1, c_2) = \frac{|gloss(c_1) \cap gloss(c_2)|}{\min(|gloss(c_1)|, |gloss(c_2)|)} \tag{5}$$

Combining Eqs. (3) and (5), we form the measurement for computing the relatedness between two concepts, as shown in Eq. (6). Plugging it into the vector space approach, as presented in Section 3.2.1, we can conclude the similarity between two sentences by using knowledge-based and corpus-based semantic relatedness, denoted by Sim_{F2}. The similarity Sim_{F2} is then used as the second feature of SVM models.

$$Rel_{KB-CB}(c_1, c_2) = \begin{cases} \max\limits_{c \in LCS(c_1, c_2)} [-\log pr(c)] & c_1, c_2 \in \text{WordNet} \\ \frac{|gloss(c_1) \cap gloss(c_2)|}{\min(|gloss(c_1)|, |gloss(c_2)|)} & c_1, c_2 \in \text{Wiktionary}, \notin \text{WordNet} \\ 0 & otherwise \end{cases} \tag{6}$$

[12] http://en.wiktionary.org.

Table 1
Statistics about the training set of STS2015.

Datasources	# pairs of sentences
STS2012-test/STS.input.MSRpar	750
STS2012-test/STS.input.MSRvid	750
STS2012-test/STS.input.SMTeuroparl	459
STS2012-test/STS.input.surprise.SMTnews	399
STS2012-train/STS.input.MSRpar	750
STS2012-train/STS.input.MSRvid	750
STS2012-train/STS.input.SMTeuroparl	734
sts2013-test/STS.input.FNWN	189
sts2013-test/STS.input.headlines	750
sts-en-test-gs-2014/STS.input.deft-forum	450
sts-en-test-gs-2014/STS.input.deft-news	300
sts-en-test-gs-2014/STS.input.headlines	750
sts-en-test-gs-2014/STS.input.images	750
sts-en-test-gs-2014/STS.input.tweet-news	750
Total	**8,531**

4. Evaluation

4.1. Datasets

We conduct the experiments on three datasets: (1) MRPC, (2) STS2015, and (3) P4PIN.

- MRPC is a popular dataset built for evaluation of method proposed for paraphrase identification task. It contains labeled sentence pairs in two classes {0, 1}. The dataset is extracted from 32,408 news clusters from World Wide Web and manually judged by humans. The training set consists of 4076 pairs (2753 true and 1323 false paraphrase pairs); the test set consists of 1725 pairs (1147 true and 578 false paraphrase pairs).
- SemEval is a series of evaluation of computational semantic analysis systems. STS2015 dataset is a benchmark dataset for task-2 of SemEval-2015[13] [56]. The training set we use to train our model consists of 8,531 labeled sentence pairs from SemEval-2014, SemEval-2013, and SemEval-2012. Table 1 presents statistics about and data-sources of the training set in detail. The test set consists of labeled sentence pairs in five domains, i.e., news headlines (HDL), images, answers-student, answers-forum, and belief. It consists of 3,000 sentence pairs. The statistics about the test set of STS2015 is presented in Table 2 of [56]. Each pair in the training and test sets is scored in range of [0, 5], as explained in Table 2.
- The P4PIN dataset consists of 3,354 instances, 847 positives and 2,507 negatives. Note that P4PIN is not inherently divided into training and test sets like MRPC and STS2015. Therefore, one can split the dataset into training/test sets or use cross validation to train and evaluate a model. In this work, we use 10-fold cross validation in experiments.

[13] http://alt.qcri.org/semeval2015/task2/index.php?id=semantic-textual-similarity-for-english.

Table 2
Explaining similarity scores of STS2015.

Score	*Meaning*
0	Two sentences are not similar.
1	Two sentences mention the same topic.
2	Two sentences share some details.
3	Two sentences differ from some important information.
4	Two sentences differ from some unimportant information.
5	Two sentences have the same meaning.

4.2. Experimental setup

In order to evaluate our proposed method, we focus on the preprocessing and two features used to train SVM models for STSS. In particular, we employ the SVM algorithms implemented in Libsvm[14] with default setting to train SVM models using features presented in Section 3. For MRPC and P4PIN, we train SVM classifiers. For STS2015, we train a SVM regression model.

Let F_1 and F_1' be features calculated using Formula 2 presented in Section 3.2.1 without/with preprocessing, respectively; F_2 be the feature based on external knowledge sources as presented in Section 3.2.2 and F_3 be the features based on average of pre-trained word vectors of a input text as the first model in [29] does. To highlight the contribution of these, we perform four experiments using F_1, F_1', $F_1'+F_2$, $F_1'+F_3$ as input features, respectively.

For the final experiment using $F_1' + F_3$ features, we represent each input text in a 300-dimensional vector space, which can be computed by *word2vec*[15] model on the Google News dataset. We conduct this experiment to evaluate how does directly using pre-trained word vectors contribute the performance. The Eq. (7) shows how we calculate the vector v_S representing the input text S, where $|w|$ is the number of words in S, v_i is the ith 300-dimensional pre-trained word vector. Given a short text pair, F_3 feature is the cosine between vectors calculated using Eq. (7).

$$v_S = \frac{1}{|w|}\sum_{i=1}^{|w|} v_i \tag{7}$$

4.3. Experimental results

To normalize the performance results, with MRPC and P4PIN, we use the *accuracy* measurement; with STS2015, we use the Pearson(ρ)/Spearman(r) correlation coefficient measurement. Table 3 presents our results on three datasets. For the case of STS2015, they are the average results of five subsets. The models trained in the first set-up (using only F_1) slightly outperforms those trained in the second (using only F_1') on two datasets STS2015 and P4PIN. We can explain this phenomena that before training the Skip-gram to obtain word2vec, the training corpus was constructed phrases, such as replacing "New York Times" and "Toronto Maple Leafs" by unique tokens [24], the models trained using only F_1 (without preprocessing) get benefit from the phrase embeddings included in word2vec. Meanwhile, the Stanford CoreNLP produced wrong coreference chains, which hurts the performance of models trained in the second set-up. The experimental results also show that the models trained in the third set-up (combining F_1' and F_2) slightly outperforms those trained in the fourth (combining F_1' and F_3) on two datasets MRPC and STS2015. Overall, the model trained by combining F_1' and F_2 performs best on two datasets MRPC and STS2015, while the model trained using only F_1 performs best on P4PIN dataset.

14 https://www.csie.ntu.edu.tw/~cjlin/libsvm/.
15 https://code.google.com/archive/p/word2vec/.

Table 3
Experimental results on MRPC, STS2015, and P4PIN in combination of features.

Features	Dataset		
	MRPC (*accuracy*)	STS2015 (ρ/r)	P4PIN (*accuracy*)
F_1	0.8133	0.8263/0.8249	**95.22**
F_1'	0.8392	0.8189/0.8161	93.94
$F_1' + F_2$	**0.8417**	**0.8373**/**0.8297**	93.82
$F_1' + F_3$	0.7949	0.7824/0.7809	93.79

Table 4
Experimental results in detail on STS2015. Results in previous work are reprinted.

Dataset	Sultan et al. [57] (ρ)	Hänig et al. [58] (ρ)	Han et al. [59] (ρ)	Ours (ρ)
answers-forums	0.7390	0.6946	0.6589	**0.7992**
belief	0.7491	0.7482	0.7029	**0.8074**
answers-students	0.7725	0.7784	0.7827	**0.8265**
headlines	0.8250	0.8245	0.8342	**0.8662**
images	0.8644	0.8527	0.8701	**0.8870**
Average	0.79	0.7797	0.7698	**0.8373**

Table 4 presents experimental results of the model trained by combining F_1' and F_2 in comparison with those of Sultan et al. [57], Hänig et al. [58], and Han et al. [59]. We sort results in ascending order. Experimental results shows that ρ/r values of our method are consistent increasing with ρ values of Sultan et al. [57], Hänig et al. [58], and Han et al. [59] and $\rho \times 100$ of our method are better than those approximately 5, 6, 7 points, respectively.

Table 5 presents the best performance of our method in comparison with the methods in literature on MRPC, STS2015, and P4PIN datasets. The table shows that our method outperforms the other benchmarks, achieving 84.17% accuracy and 15.83% error rate on the test set, except transfer learning approach using BERT pre-trained representation model presented in [1]. Overall, our method outperforms others in literature on the same datasets: MRPC, STS2015, and P4PIN, except BERT-based approach presented in [1]. However, $BERT_{LARGE}$ was trained on 16 Cloud TPUs (64 TPU chips total) and took 4 days to complete, putting it out of reach of low-resource researchers.

Experimental results show that interdependent representations of short text pairs are effective and efficient in paraphrase identification. In interdependent representations, our proposed method represents input words by two approaches. First, by applying word embeddings, we can capture a various dimensional vector of both semantic and syntactic information for representing a word. Second, instead of representing words in high dimensional vector, we exploit a semantic network and large corpus to measure semantic relatedness between concepts. By taking advantage of these two approaches, our method gains a better performance than almost all other approaches on the three datasets.

5. Conclusion

In this paper, we presented a novel method for determining the degree of semantic similarity between pairs of short texts. The proposed method exploits the similarity between word contexts built by using word embeddings and semantic relatedness between concepts based on external sources of knowledge. It also performs both coreference resolution of named entities in two short texts to chain entities together and word segmentation to preserve the meaning of phrasal verbs and idioms. We evaluate our method on three popular datasets: MRPC, STS2015, and P4PIN. Experimental results show that all presented features contribute to the semantic similarity and the proposed

Table 5
Experimental results on MRPC, STS2015, and P4PIN in comparison with those of previous work. Results in previous work are reprinted. In the Rep. column, *Rep.* stands for sentence representation approach, *fe* stands for feature engineering, *ts* stands for task-specific, *fb* stands for feature-based, *ft* stands for fine-tuning.

Method	MRPC	STS2015	P4PIN	Rep.
Mihalcea et al. (2006) [11]	70.3%	–	–	fe
Das and Smith (2009) [18]	73.9%	–	–	fe
Socher et al. (2011) [19]	76.8%	–	–	ts
Madnani et al. (2012) [15]	77.4%	–	–	fe
Wang et al. (2016) [45]	78.4%	–	–	ts
Ji and Eisenstein (2013) [13]	80.4%	–	–	ts +fe
Issa et al. (2018) [22]	68.7%	–	–	ts
Hu et al. (2014) [44]	69.9%	–	–	ts
Yin and Schütze (2015) [28]	78.1%	–	–	ts
Yin et al. (2015) [43]	78.9%	–	–	ts
He et al. (2015) [46]	78.6%	–	–	ts
Agarwal et al. (2018) [42]	77.7%	–	–	ts
Wieting et al. (2015) [29]	–	78.5%	–	fb
Sultan et al. (2015) [57]	–	79.00%	–	fe
Hänig et al. (2015) [58]	–	77.97%	–	fe
Han et al. (2015) [59]	–	76.98%	–	fe
Kajiwara et al. (2017) [60]	–	63.6%	–	fe
Kiros et al. (2015) [31]	73.0%	–	–	fb
Hill et al. (2016) [30]	76.4%	–	–	fb
Pagliardini et al. (2018) [32]	72.5%	–	–	fb
Logeswaran and Lee (2018) [34]	76.9%	–	–	fb
Conneau et al. (2018) [33]	76.2%	–	–	fb
Radford et al. (2018) [35]	82.3%	–	–	ft
Subramanian et al. (2018) [36]	78.6%	–	–	ft
Devlin et al. (2018) [1]	**89.3%**	–	–	ft
Álvarez-Carmona et al. (2018) [14]	83.0%	–	93.0%	fe
This paper	**84.17%**	**83.73%**	**95.22%**	fe

method outperforms others, except one, in literature without using linguistic information.

Note that, even though we use vector representations of short texts, each vector is dense, low-dimensional, and real-valued. However, although beating almost all state-of-the-art systems, our learning model is supervised and needs feature engineering; whereas, unsupervised representation models such as those proposed in [32,43] can automatically learn features for modeling pairs of sentences using neural networks with or without attention mechanisms. The performance of such models is currently lower than that of state-of-the-art supervised learning systems but they have strong potential in the current scenario, where availability of data and labels is shrinking. Moreover, language model pre-training [61] and data augmentation [62] are also new directions of research. In the near future, we plan to explore how these unsupervised models can be improved by exploiting external knowledge.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.07.013.

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.

[2] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, IEEE Comput. Intell. Mag. 13 (3) (2018) 55–75.

[3] A. Abdi, N. Idris, R.M. Alguliyev, R.M. Aliguliyev, PDLK: Plagiarism detection using linguistic knowledge, Expert Syst. Appl. 42 (22) (2015) 8936–8946.

[4] M. Mohler, R. Bunescu, R. Mihalcea, Learning to grade short answer questions using semantic similarity measures and dependency graph alignments, in: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, 2011, pp. 752–762.

[5] M. Heilman, N.A. Smith, Tree edit models for recognizing textual entailments, paraphrases, and answers to questions, in: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2010, pp. 1011–1019.

[6] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, X. Cheng, A deep architecture for semantic matching with multiple positional sentence representations, in: AAAI, 2016, pp. 2835–2841.

[7] D. Kauchak, R. Barzilay, Paraphrasing for automatic evaluation, in: Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, 2006, pp. 455–462.

[8] E. Cambria, S. Poria, D. Hazarika, K. Kwok, SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings, in: AAAI, 2018, pp. 1795–1802.

[9] R.M. Aliguliyev, A new sentence similarity measure and sentence based extractive technique for automatic text summarization, Expert Syst. Appl. 36 (4) (2009) 7764–7772.

[10] J. Madhavan, P.A. Bernstein, A. Doan, A. Halevy, Corpus-based schema matching, in: Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on, 2005, pp. 57–68.

[11] R. Mihalcea, C. Corley, C. Strapparava, Corpus-based and knowledge-based measures of text semantic similarity, in: AAAI, AAAI Press, 2006, pp. 775–780.

[12] N.X. Bach, M.L. Nguyen, A. Shimazu, Exploiting discourse information to identify paraphrases, Expert Syst. Appl. 41 (6) (2014) 2832–2841.

[13] Y. Ji, J. Eisenstein, Discriminative improvements to distributional sentence similarity, in: EMNLP, ACL, 2013, pp. 891–896.

[14] M.A. Álvarez-Carmona, M. Franco-Salvador, E. Villatoro-Tello, M. Montes-y Gómez, P. Rosso, L. Villaseñor-Pineda, Semantically-informed distance and similarity measures for paraphrase plagiarism identification, J. Intell. Fuzzy Systems 34 (5) (2018) 2983–2990.

[15] N. Madnani, J.R. Tetreault, M. Chodorow, Re-examining machine translation metrics for paraphrase identification, in: HLT-NAACL, The Association for Computational Linguistics, 2012, pp. 182–190, URL http://www.aclweb.org/anthology/N12-1019.

[16] A. Islam, D. Inkpen, Semantic text similarity using corpus-based word similarity and string similarity, ACM Trans. Knowl. Discov. Data 2 (2) (2008) Article 10.

[17] Y. Li, D. McLean, Z. Bandar, J. O'Shea, K.A. Crockett, Sentence similarity based on semantic nets and corpus statistics, IEEE Trans. Knowl. Data Eng. 18 (8) (2006) 1138–1150.

[18] D. Das, N.A. Smith, Paraphrase identification as probabilistic quasi-synchronous recognition, in: K.-Y. Su, J. Su, J. Wiebe (Eds.), ACL/IJCNLP, The Association for Computer Linguistics, 2009, pp. 468–476.

[19] R. Socher, E.H. Huang, J. Pennington, A.Y. Ng, C.D. Manning, Dynamic pooling and unfolding recursive autoencoders for paraphrase detection, in: J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, K. Q. Weinberger (Eds.), NIPS, 2011, pp. 801–809.

[20] M. Sahami, T.D. Heilman, A web-based kernel function for measuring the similarity of short text snippets, in: L. Carr, D.D. Roure, A. Iyengar, C.A. Goble, M. Dahlin (Eds.), WWW, ACM, 2006, pp. 377–386.

[21] J. Turian, L. Ratinov, Y. Bengio, Word representations: a simple and general method for semi-supervised learning, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2010, pp. 384–394.

[22] F. Issa, M. Damonte, S.B. Cohen, X. Yan, Y. Chang, Abstract meaning representation for paraphrase detection, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, 2018, pp. 442–452.

[23] T.K. Landauer, P.W. Foltz, D. Laham, An introduction to latent semantic analysis, Discourse Process. 25 (2–3) (1998) 259–284.

[24] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 3111–3119.

[25] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation, in: EMNLP, Vol. 14, 2014, pp. 1532–1543.

[26] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of NAACL-HLT, 2018.

[27] T. Kenter, M. de Rijke, Short text similarity with word embeddings, in: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, ACM, 2015, pp. 1411–1420.

[28] W. Yin, H. Schütze, Convolutional neural network for paraphrase identification, in: HLT-NAACL, 2015, pp. 901–911.

[29] J. Wieting, M. Bansal, K. Gimpel, K. Livescu, Towards universal paraphrastic sentence embeddings, Clin. Orthop. Relat. Res. (2015) abs/1511.08198.

[30] F. Hill, K. Cho, A. Korhonen, Learning distributed representations of sentences from unlabelled data, in: K. Knight, A. Nenkova, O. Rambow (Eds.), HLT-NAACL, The Association for Computational Linguistics, 2016, pp. 1367–1377.
[31] R. Kiros, Y. Zhu, R.R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, S. Fidler, Skip-thought vectors, in: Advances in Neural Information Processing Systems, 2015, pp. 3294–3302.
[32] M. Pagliardini, P. Gupta, M. Jaggi, Unsupervised learning of sentence embeddings using compositional n-gram features, in: NAACL 2018 - Conference of the North American Chapter of the Association for Computational Linguistics, 2018.
[33] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, A. Bordes, Supervised learning of universal sentence representations from natural language inference data, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, 2017, pp. 670–680.
[34] L. Logeswaran, H. Lee, An efficient framework for learning sentence representations, in: Proceedings of ICLR, 2018.
[35] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, 2018, [Online] Available: https://blog.openai.com/language-unsupervised/.
[36] S. Subramanian, A. Trischler, Y. Bengio, C.J. Pal, Learning general purpose distributed sentence representations via large scale multi-task learning, in: Proceedings of ICLR, 2018.
[37] W.B. Dolan, C. Brockett, Automatically constructing a corpus of sentential paraphrases, in: Proc. of IWP, 2005.
[38] J.F. Sánchez-Vega, Identificación de plagio parafraseado incorporando estructura, sentido y estilo de los textos Ph.D. thesis, 2016.
[39] P.H. Duong, H.T. Nguyen, V.P. Nguyen, Evaluating semantic relatedness between concepts, in: IMCOM, ACM, 2016, 20:1–20:8.
[40] H.T. Nguyen, P.H. Duong, T.Q. Le, A multifaceted approach to sentence similarity, in: V.-N. Huynh, M. Inuiguchi, T. Denoeux (Eds.), IUKM, in: Lecture Notes in Computer Science, vol. 9376, Springer, 2015, pp. 303–314.
[41] W. Guo, M. Diab, Modeling sentences in the latent space, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers, Vol. 1, Association for Computational Linguistics, 2012, pp. 864–872.
[42] B. Agarwal, H. Ramampiaro, H. Langseth, M. Ruocco, A deep network model for paraphrase detection in short text messages, Inf. Process. Manage. 54 (6) (2018) 922–937.
[43] W. Yin, H. Schütze, B. Xiang, B. Zhou, ABCNN: Attention-based convolutional neural network for modeling sentence pairs, Trans. Assoc. Comput. Linguist. 4 (2016) 259–272.
[44] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional neural network architectures for matching natural language sentences, in: Advances in Neural Information Processing Systems, 2014, pp. 2042–2050.
[45] Z. Wang, H. Mi, A. Ittycheriah, Sentence similarity learning by lexical decomposition and composition, in: N. Calzolari, Y. Matsumoto, R. Prasad (Eds.), COLING, ACL, 2016, pp. 1340–1349.
[46] H. He, K. Gimpel, J.J. Lin, Multi-perspective sentence similarity modeling with convolutional neural networks, in: L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, Y. Marton (Eds.), EMNLP, The Association for Computational Linguistics, 2015, pp. 1576–1586.
[47] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: A. Moschitti, B. Pang, W. Daelemans (Eds.), EMNLP, ACL, 2014, pp. 1724–1734.
[48] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[49] P.J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer, Generating wikipedia by summarizing long sequences, in: Proceedings of ICLR, 2018.
[50] K. Bontcheva, M. Dimitrov, D. Maynard, V. Tablan, H. Cunningham, Shallow methods for named entity coreference resolution, in: Chaınes de références et résolveurs d'anaphores, Workshop TALN, 2002.
[51] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, Clin. Orthop. Relat. Res. (2013) abs/1301.3781. URL http://arxiv.org/abs/1301.3781.
[52] T. Zesch, I. Gurevych, Wisdom of crowds versus wisdom of linguists - measuring the semantic relatedness of words, Nat. Lang. Eng. 16 (1) (2010) 25–59.
[53] P. Resnik, Using information content to evaluate semantic similarity in a taxonomy, in: IJCAI, Morgan Kaufmann, 1995, pp. 448–453.
[54] R. Rada, H. Mili, E. Bicknell, M. Blettner, Development and application of a metric on semantic nets, IEEE Trans. Syst. Man Cybern. 19 (1) (1989) 17–30.
[55] M. Lesk, Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone, in: V. DeBuys (Ed.), SIGDOC, ACM, 1986, pp. 24–26.
[56] E. Agirre, C. Banea, C. Cardie, D.M. Cer, M.T. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigau, L. Uria, J. Wiebe, SemEval-2015 Task 2: Semantic textual similarity, English, Spanish and Pilot on interpretability, in: D.M. Cer, D. Jurgens, P. Nakov, T. Zesch (Eds.), Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015, The Association for Computer Linguistics, 2015, pp. 252–263, URL http://aclweb.org/anthology/S/S15/.
[57] M.A. Sultan, S. Bethard, T. Sumner, DLS@CU: Sentence similarity from word alignment and semantic vector composition, in: D.M. Cer, D. Jurgens, P. Nakov, T. Zesch (Eds.), Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, The Association for Computer Linguistics, 2015, pp. 148–153.
[58] C. Hänig, R. Remus, X. de la Puente, Exb themis: Extensive feature extraction from word alignments for semantic textual similarity, in: D.M. Cer, D. Jurgens, P. Nakov, T. Zesch (Eds.), Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, The Association for Computer Linguistics, 2015, pp. 264–268.
[59] L. Han, J. Martineau, D. Cheng, C. Thomas, Samsung: Align-and-differentiate approach to semantic textual similarity, in: D.M. Cer, D. Jurgens, P. Nakov, T. Zesch (Eds.), Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT, The Association for Computer Linguistics, 2015, pp. 172–177.
[60] T. Kajiwara, D. Bollegala, Y. Yoshida, K.-i. Kawarabayashi, An iterative approach for the global estimation of sentence similarity, PLoS One 12 (9) (2017) e0180885.
[61] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, 2019, URL https://openai.com/blog/better-language-models/.
[62] Q. Xie, Z. Dai, E. Hovy, M.-T. Luong, Q.V. Le, Unsupervised data augmentation, 2019, https://arxiv.org/pdf/1904.12848.pdf.