Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Structural Sentence Similarity Estimation for Short Texts

Weicheng Ma
Computer Science Department
City University of New York
365 5th Ave., New York, NY
wm724@nyu.edu

Torsten Suel
Computer Science Department
Tandon School of Engineering
New York University
6 MetroTech Center, Brooklyn, NY
torsten.suel@nyu.edu

Abstract

Sentence similarity is the basis of most text-related tasks. In this paper, we define a new task of sentence similarity estimation specifically for short while informal, social-network styled sentences. The new type of sentence similarity, which we call Structural Similarity, eliminates syntactic or grammatical features such as dependency paths and Part-of-Speech (POS) tagging which do not have enough representativeness on short sentences. Structural Similarity does not consider actual meanings of the sentences either but puts more emphasis on the similarities of sentence structures, so as to discover purpose- or emotion-level similarities. The idea is based on the observation that people tend to use sentences with similar structures to express similar feelings. Besides the definition, we present a new feature set and a mechanism to calculate the scores, and, for the needs of disambiguating word senses we propose a variant of the Word2Vec model to represent words. We prove the correctness and advancement of our sentence similarity measurement by experiments.

该方法没有按照传统的方法依据句法和语法特征进行建模，例如依赖路径和词性标签信息。

该方法没有考虑句子的实际含义，而是更多的注重于句子结构的相似性，以便发现目的或情感上的相似性

Introduction

Proper estimation of sentence similarity is crucial to most text-related applications. Question Answering Systems require sentence similarity to match question-question or question-answer pairs(Wang and Ittycheriah 2015; Yan and Tian 2015). Similarity scores between the original sentence and that in the destination language manages the decision at each step in Machine Translation tasks(Tu et al. 2015; Xu et al. 2014). However the applications seldom follow the same standard in defining the sentence similarities. For example sentimental analysis tasks focus more on critical words while question answering systems rely more on syntactic features.

The difference in selections of sentence similarity measurement is due to the distinction of their goals, so it is hard to give a pervasive definition for all the text-related tasks. Due to the fact that most sentences on social-networks are short while highly informal in grammar and the use of words, it does not seem to be a valid solution to apply either syntactic or semantic analysis directly on social-network corpora. Some systems try to correct the grammars and words then treat them in the same way as grammatically correct sentences. But the elimination of the special expressions (e.g., emoticons, specially ordered sentences, etc.) and self-made words (e.g., Whaaaaat, sooooo, etc.) causes a clear loss of emotional information. Other methods, especially neural network based systems like Convolutional Neural Networks (CNNs)(He, Gimpel, and Lin 2015; Tu et al. 2015; Zhang, Zhao, and LeCun 2015; Hu et al. 2014) and Recurrent Neural Networks (RNNs)(Cho et al. 2014; Hill et al. 2015; Chung et al. 2014) are successful in avoiding building syntactic analysis pipelines, but the size and number of sliding windows in CNNs, as well as the recursion times in RNNs are strictly limited by the length of the social-network styled sentences, thus affecting the accuracy of sampling and parameter tuning. So new a new standard of sentence similarity estimation is urgently needed for discovering human emotions and feelings in short and informal oral sentences.

Previous work has been done in solving the short sentence similarity estimation problem(Metzler, Dumais, and Meek 2007; Tang et al. 2012; Islam and Inkpen 2009; Chen et al. 2012). (Xu et al. 2014) combined multiple features including lexicon, POS tagging and topic information in paraphrase detection task. Their work was evaluated on a SemEval task. (Li et al. 2006) combined word sequence features into a semantic analysis task and solved the classic "who bites who" problem (e.g., "The cat bites the dog" vs. "The dog bites the cat"). Though most of these previous work claimed high performance in specific tasks, they still concentrated more on the similarities of actual meanings of the sentences so still performed like variations of the standard Bag-of-Words methods. And since they still used WordNet plus POS tagging to disambiguate words in the sentences, their success could hardly be extended to social-network corpora.

To overcome the disadvantages of former works on purpose mining task over social-network corpora, we in this paper present a new definition to sentence similarities specifically for those short while casual sentences. We do not emphasize sentiment analysis in our work but it is completely fine to combine our mechanism with other text analysis methods to tackle different problems. Also it is possible that some applications such as sentiment analysis(Pang and Lee 2008; Rosenthal et al. 2015) or event extraction(Ritter

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

et al. 2012; Zhou, Chen, and He 2015) prefer simpler feature set, while the features we choose are for general purpose on short sentence analysis tasks.

One problem of dropping syntactic parsing completely lies in the word sense disambiguation task. To deal with this problem, we propose a new vector representation of words, which is a variation of the Word2Vec model. Different from using single vector, we cluster candidate vectors and keep multiple seed vectors when representing each word. In use, for a certain word we combine the word vectors of every other word in its context to form a new vector, calculate cosine similarity between the new vector with each vector representing the word, select the closest vector and return it with the similarity.

In the next section we give formal definition of the task on which our method is designed. We arrange feature selection and sentence similarity calculation metrics in Section 3. In Section 4 we describe the language model we use to represent the words. We design experiments to test our system and show the results in Section 5. We put the conclusion and future plans at the end of this paper.

Task Definition

In this section we formally define the task of structural sentence similarity estimation for short while grammatically informal sentences specifically. Our sentence similarity does not necessarily reveal similarities in meaning, but discovers emotional feelings. Our original idea was based on the observation that people tend to use similar sentence structures to express similar emotions. This definition can also work with other methods such as semantic parsing or RNNs with no negative effect to the results.

Applicable Corpora

Intended use of this definition of sentence similarity estimation is on short and informal sentences which can not be accurately parsed by grammar based syntactic parsers. Examples of such datasets include social-network posts such as Twitter tweets, SMS texts and a portion of questions in question answering systems.

Structural Sentence Similarity

The kind of sentence similarity we present here is different from other measurements in the sense that it does not induce sentences to grammar rules but tries to extract the skeletons of both sentences. Another difference between structural sentence similarity and semantic similarity is that the lexical meanings of words do not govern but the contexts do. These characteristics make structural sentence similarity less sensitive to distinctions in actual meaning such as subject-object differences. Instead minor difference in sentence skeletons, for example reversely ordered subject-predicate pairs, may result in big difference in similarity scores. Still using the cat-mouse example, semantic sentence similarity estimation systems would say that the sentence "A cat chases a mouse" is more similar to the sentence "A mouse is chased by a cat" than the sentence "A mouse chases a cat". However, for the same sentence ("A cat chases a mouse"), our structural sentence similarity estimation method would prefer the latter if the words "mouse" and "cat" appear frequently in similar contexts.

Structural sentence similarity can be easily combined with semantic analysis or sentimental analysis for some specific tasks.

Essential Elements

Word Vector A good representation of words is critical for a text analysis system. The use of vector representation for words attracted much attention these days. The advantages of applying vector representation of words is that the vectors reveal context information, which is the core of our task.

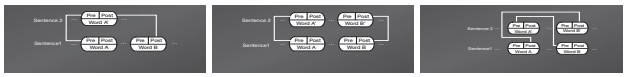
Famous word representation methods include the Word2Vec model by Dr. Mikolov(Mikolov et al. 2014), and the recently published SkipThought model(Kiros et al. 2015). We choose the SkipGram model since the documents we research, the bodies of tweets, do not contain that many sentences.

The main drawback of the Word2Vec model is that since each word in SkipGram model has only one vector, it is hard to differentiate different meanings of a word. Moreover, preceding and following contexts are not distinguished in the Word2Vec model, which are important in determining the borders of a phrase. So in our experiment we use a variated version of Word2Vec to overcome these shortcomes.

Sentence Division Most of our former works still regard sentences as a set of words, and sentence similarities as a combination of word similarities. (Cho et al. 2014; Hill et al. 2015; Kiros et al. 2015) claimed that phrases should be put more emphasis when analyzing text documents. To go one step further, we do not regard phrases to be groups of words that tend to appear together in a corpora. Instead, we think of every sequence of words that expresses certain meaning to be a phrase, thus defining the sentences to be sequences of phrases. Moreover, according to our definition even the division of the same sentence may not be the same when compared with different sentences, since the corresponding phrases in the other sentence may not be dividable.

To properly divide sentences, purely relying on word similarity overlaps is not enough, with the risk that the whole sentence being treated as one phrase. Thus we try to find a good division by balancing the scores of features including word-level similarities, phrase formation possibilities, word adjacencies and sum distances, phrase length and existence reward with entropy and reorder penalties. We will discuss the features in detail in the next section.

Longest Similar Phrase Sequence (LSPS) On sentence-level we require a one to one correspondence of phrases when calculating the sentence similarity scores. We fill up a correspondence matrix of two sentences by the score of match for each pair of phrases, one from each sentence. Here we set a threshold to eliminate scores that are too low in case that sentence similarity is affected by irrelevant terms. For a selected pair of phrases, each component blocks the phrase following it from matching the phrases preceding the



(a) Two words correspond to a same word

(b) Word correspondence in order

(c) Word correspondence out of order

Figure 1: Possible phrase boundaries

other component, so for each element M[a,b] in the matrix M with r rows and c columns, we recursively pick the one minimizing the sum values of matrices M[0:a,b+1:c] and M[a+1:r,0:b], then set both matrices to be zero matrices. It then quits the loop once the whole similarity matrix becomes zero matrix. This operation helps preserve longer correspondence chains and will not be bothered by identical but ill-ordered sentence pairs.

Feature Selection and Calculation

Feature Design

In this part we introduce the features we choose to use in more details. We divide the scoring of a match into two levels, namely, phrase-level and sentence-level and we use different sets of features on each level. On phrase-level we evaluate the candidates in two distinct directions, one being the phrase formation probability and the other being the phrase correspondence score. On sentence-level we consider only score of matches and disorderedness. Moreover, we give some tolerance to out-of-order words when evaluating phrases but ignore ill-ordered phrases completely since the order of sentence components decides the structure of the sentence.

Phrase Boundary Match According to our definition of phrases, the division of phrases should be specific to the sentence pair. Based on empirical experiences, words with similar contexts in one direction are likely to be the boundaries on the same side of two corresponding phrases. As we assume vector representation of words, the boundaries of a possible phrase can be represented by matching the first word's preceding words vector and vector of following words for the last word in the first string to those of the words at corresponding positions in the second string. As is shown in Figure 1, there exists three types of phrase boundary matching.

To evaluate the similarity of two boundary words, we take the word vector on a single direction for each word and score the matching by calculating cosine similarity using

$$p_{boundary} = \frac{\mathbf{pre} \bullet \mathbf{pre}^{'}}{\|\mathbf{pre}\|\|\mathbf{pre}^{'}\|} \bullet \frac{\mathbf{post} \bullet \mathbf{post}^{'}}{\|\mathbf{post}\|\|\mathbf{post}^{'}\|} \quad (1)$$

where $p_{boundary}$ represents the similarity of boundary words on both sides for a pair of phrases, $\mathbf{pre}$ and $\mathbf{post}$ denote the word vectors of the front and back boundaries of the first string respectively while $\mathbf{pre}^{'}$ and $\mathbf{pre}^{'}$ are the corresponding vectors in the second string.

Phrase Validation According to the definition of SkipGram Model, when given the context it is possible to predict a missing word. We use this property to measure the possibility that a word fits in a phrase with known boundaries. Take the sentence pair in Figure 2 as an example, if without skipGram model prediction only "NYC restaurant" will be matched to "Chicago theater", which causes information loss. This feature of SkipGram model is also the core of WSD phase. Still using the example sentences in Figure 2, the word fish in sentence 1 can only be of its noun meaning considering its context.

Moreover, we give common phrases more credits by rewarding existing word sequences in history. For example we may prefer grouping the words "have been to" instead of "I have" in Figure 2 since they appear together more usual in the corpora.

Phrase Length When other conditions hold, we want the phrases to be longer for lower sentence entropy and stronger connections between phrases. So phrase lengths should be positively related to phrase similarities. Like in Figure 2, "have been to" will be grouped as one phrase instead of just the first two words according to this rule. This is beneficial to our task since if we exclude the word "to" from the phrase it will have no match at all so unnecessarily lowering the similarity score of the sentences.

Word Distance in Phrase From observation, the further two words are apart in the sentence, the less likely they are related. And the connection should get weakened with the increase of distance, which agrees with our experimental results. For example in the sentence "A new store is about to be built in York.", the words "new" and "york" are not likely to be bonded together, nor are other distant word pairs.

Also there is a special case in evaluating word distances, which is adjacency. Words separated by 3 or 6 words may not be that different in the possibility of forming a phrase, but it is a big difference for adjacent words versus a one word separated pair. For instance, we prefer "be built" to "be in York" in the former sentence even though the latter is longer.

Word Reorder Cost Sometimes ill-ordered correspondences are too informative to be simply ignored. So instead of removing all randomly ordered phrases, we apply penalty to each reordering operation. Consider the case in Figure 3, we prefer matching "Yesterday the concert" to "the concert yesterday" rather than incorporating the word "great" since the more reordering operations there are, the less likely the words will form a phrase.

Phrase Entropy Though dynamic sentence division system can divide sentences smarter, the accuracy may be harmed by the trend to combine all the words in a sentence into one phrase if only positive weights are given for phrase formation. To tackle this problem we import entropy on phrase-level to maintain the stability of phrases. For each word w_i in the phrase ph, the entropy $H(ph)$ can be calculated by

$$H(ph) = -\sum_{w_i \in ph} p(w_i|ph) \log p(w_i|ph) \quad (2)$$

$$r(s_1, s_2) = \frac{\prod_{ph \in s_1, ph' \in s_2} sim(ph, ph') \times \sum p(ph) \times \sum p(ph') \times \sum len(ph) \times \sum len(ph') \times \sum adj(ph) \times \sum adj(ph')}{[\sum_{i=1}^{s_1.length} H(p_i^1) + \sum_{i=1}^{s_2.length} H(p_i^2)] \times \sum dist(ph) \times \sum dist(ph') \times \sum reorder(ph, ph')} \quad (7)$$

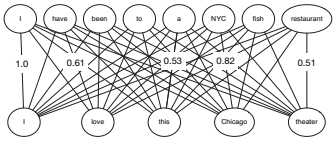


Figure 2: Word match example

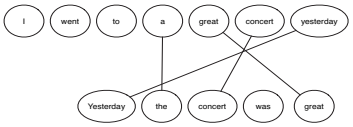


Figure 3: Word reorder example

where $p(w_i|ph)$ here denotes the validity of w_i to be at position i in the phrase ph.

Sentence Entropy

For the same reason of incorporating entropy in phrasal analysis, on sentence-level we also introduce entropy to avoid over-fusion of phrases. The calculation methods of sentence-level entropy and phrase-level entropy share the same formulation while the $p(w_i|ph)$ is changed to $sim(ph_i, ph_j^{'})$ which is the similarity between each phrase ph in sentence 1 and its corresponding phrase $ph^{'}$ in sentence 2. The sentence-level entropy and phrase-level entropy are mutual exclusive so there should always be an optimal division for an arbitrary pair of sentences.

Calculation Mechanisms

Based on the features we described above, first we express the similarity of a sentence pair(s_1, s_2) in the way of

$$r(s_1, s_2) = \frac{p(s_1, s_2)}{H(s_1) + H(s_2)} \quad (3)$$

where $r(s_1, s_2)$ is the similarity between sentences s_1 and s_2. $p(s_1, s_2)$ denotes the product score of corresponding phrases in the two sentences while $H(s_1)$ and $H(s_2)$ are the sentence entropies of s_1 and s_2, respectively.

Since we have already incorporated contextual features in word vectors, here we ignore the dependencies between phrases. Thus for each pair of phrases p_i^1 in sentence 1 and p_i^2 in sentence2, we can expand the equation to be

$$r(s_1, s_2) = \frac{\prod^{p_i^1 \in s_1, p_j^2 \in s_2} p(p_i^1, p_j^2)}{\sum_{i=1}^{s_1.length} H(p_i^1) + \sum_{i=1}^{s_2.length} H(p_i^2)} \quad (4)$$

in which $p(p_i^1, p_i^2)$ represents the similarity score of a pair of phrases p_i^1 in sentence 1 and p_j^2 in sentence 2.

When calculating phrase similarity scores we also take into account the possibilities that the words fit well in each string. We quantify this possibility using the language model. Since each word in the SkipGram model is represented by a vector, which denotes the context of the word in training set, we can safely say that a word fits for its context in the evaluated phrase if and only if the word vector of this word is close to the aggregated vector of all its context words. So combining the boundary validity conditions and word reorder costs, we write the function for calculating phrase similarity scores (sequence numbers i and j removed) as

$$\begin{aligned} p(p_1, p_2) = [sim^-(begin_1, begin_2) + sim^+(end_1, end_2)] \\ * [p(p_1) + p(p_2)]/[2 * reorder(p_1, p_2)] \end{aligned} \quad (5)$$

where $p(p_1, p_2)$ is the similarity score while $p(p_1)$ and $p(p_2)$ denote the possibility that the phrases p_1 and p_2 are valid. The variables $begin$ and end are the first and last words respectively in each phrase while their subscripts reveal the phrase the words belong to. For the functions, sim^- calculates word similarity considering only the preceding half while sim^+ acts the opposite. Reorder calculates the reordering penalty costs for the phrases p_1 and p_2.

Taking the phrase-level features into consideration, we further expand the validity of phrase to be

$$\begin{aligned} & p(p) \\ & = \frac{\prod_{i=1}^{n} p(w_i|p) * \sum_{i=1}^{n-1} adj(w_i, w_{i+1}, w_i^{'}, w_{i+1}^{'}) * len(p)}{\sum_{i=1}^{n} dist(w_i, w_{i+1}, w_i^{'}, w_{i+1}^{'})} \\ & + exist(p) \end{aligned} \quad (6)$$

where p being the phrase on which to calculate the score and n being its size. $p(w|p)$ calculates the degree that a word w fits in the phrase using the similarity between the word vector of w and the aggregated vector from all the words in its context. The function $len(p)$ plots the length of p onto an increasing function (in our experiment, a square function), and $exist(p)$ checks the history and normalizes the frequency to scale [0,1] as rewards to existing phrases. Functions adj and $dist$ evaluates that given a pair of words in phrase 1 and their corresponding pair of words in phrase 2, whether both pairs are adjacent words in the original sentence, and what the sum distances are, respectively.

After specifying all the terms needed to calculate the sentence similarity, we expand Equation (1) to be Equation (7), which is calculable since every element is observed in this formulation.

Our system treats the sentence similarity problem using a top-down method. The input sentence is first regarded as a single phrase when the system starts. At each step a breakpoint is added either to a phrase in sentence 1, or to a phrase in sentence 2, or both. The added point(s) must have positive effect to the similarity score of the two sentences. If no such breakpoint exists, the system quits. Otherwise it updates the score matrix as well as the sentence division then repeats the action of adding breakpoints.

Word Representation

The Word2Vec model is a commonly used word representation model based on SkipGram model(Mikolov et al. 2013). For each word appearing not too few times in the training data, the Word2Vec model represents it as a vector. Though Word2Vec styled word vectors can satisfy most of our needs especially for the convenient word similarity calculation metrics and the context based prediction mechanism, there exists some drawbacks if we apply the Word2Vec model directly. First since the word vectors are one for each word, different meanings of a word may be screwed up and hence the accuracy of word similarity calculation will be influenced. Also Word2Vec word vectors never apparently distinguish the preceding and following context, which makes it hard to discover phrase borders. Due to these drawbacks we designed a new language model in which every word is represented by a set of vectors, each standing for one meaning of the word. Also each vector is separated into two directions, for the use of phrase boundary deciding.

Experiments and Results

In this section we design experiments to evaluate the validity of our mechanism. Since it follows our definition to sentence similarity, the results are highly emotion-directed. Given the fact that there exists neither a public clean Twitter corpus nor a test corpus tagged with emotions or feelings, it is hard for us to test our system directly. Thus we divide the experiments into two halves, namely a similarity ranking problem and a sentiment analysis problem. The details and results of these experiments are shown as follows.

Sentence Similarity Ranking Experiment

This experiment makes our system tag the similarity between each pair of sentence in a corpus and rank them. We evaluate the ranking of each sentence to calculate the accuracy.

In the experiment we chose 100 sentences in total, manually divided them into 20 groups, each with 5 sentences, according to relevancies. We compared the performance of our system with those of N-Gram and CNNs methods. A sample group of sentences is as shown in Figure 4, while we show the scores the systems got in Table 1. Numbers in parenthesis denote the value of N of the N-Gram these methods use.

Figure 4: Sentence Similarity Ranking Example

System	Average	Weighed
Bag-of-Words (5)	66.0%	47.8%
CNN (5)	70.0%	54.8%
Bag-of-Words (3)	68.0%	52.0%
CNN (3)	67.5%	52.8%
Structural	71.5%	**65.2%**
Structural+LSPS	**78.0%**	54.5%

Table 1: Performance on 20 groups of data (5/group)

Here the Structural method evaluates the similarity of every pair of corresponding phrases in the sentence pair while the Structural+LSPS method eliminates all ill-ordered phrases. Moreover, by Average score we treat each sentence as equal but for Weighted score we give the sentences with lower average correct rate higher weights, so as to emphasize the ability of treating more informal sentences.

From the results we observe that our system is too sensitive to wrong divisions. This is to say, once we add an improper breakpoint to either sentence, the mistake accumulates at each further step and causes significant trouble at the final reasoning phase. This problem gets better when we provide the system with some prior knowledge like valid phrases. But we still want to keep our mechanism from adding too much human interaction. We will in our future work try to eliminate the cold start problem.

Sentiment Analysis Experiment

In this experiment we evaluate our sentence similarity estimation mechanism over a clustering-classification task. The corpus we used was Sentiment 140, a tagged Twitter sentiment analysis corpus(Go, Bhayani, and Huang 2009). To save training time we sampled 1000 tweets from the training set of Sentiment 140 to cluster using NLTK clusterer, while we check the performance of the classifier on its test set of size 500. We compare the results by our method with those by pure N-Gram and N-Gram with POS tagging methods as baseline. Moreover we put the state-of-the-art results (Stanford) here. The results are shown in Table 2.

Though with bias, from the sentence similarity ranking experiment we can see that our structural sentence similarity estimation mechanism works well at deciding the feelings or purposes of tweet publishers. However for the weighted score the structural similarity with LSPS dropped below the original version. It should be due to the effect of random positioning of adverbials. This implies problem of our not giving any tolerance to ill-ordered phrases. In the future work we will try to solve this problem by add sentence-level re-

System	F-score
Structural	65.05%
Structural+POS	66.01 %
Stanford	**69.02%**
N-Gram	54.07%
N-Gram+POS	57.72%

Table 2: Sentiment Analysis Evaluation

ordering features. In the second experiment we tested our system on a sentiment analysis corpus due to the lack of tagged purpose mining corpora. The fact that syntactic parsing did not provide much advancement in scores proved our claim that syntactic parsing was not important in predicting feelings or attitudes. Moreover, though our system did not beat the state-of-the-art results by Stanford University, the close scores showed us the possibility for unsupervised or less supervised methods to beat supervised ones, towards which direction we will keep researching.

Conclusion

In this paper, we presented a new definition of sentence similarity estimation, which we call structural similarity. Using this new type of sentence similarity we hope to tackle the problem of purpose prediction on social-network corpora. Also we proposed a calculation mechanism for structural similarity. Our mechanism differs from the classical sentence similarity estimation mechanisms in the sense that it relies completely on contextual information, and that it is an attempt to solve word sense disambiguation problem without syntactic parsing or POS tagging. Though our system did not beat the state-of-the-art system in sentiment analysis problem, it showed the possibility of getting rid of grammatical regulations in analysis of human emotions. We will try to improve the performance of our system by refining our formulations and by adjusting parameters, probably using Recurrent Neural Network.

References

Chen, X.; Li, L.; Xu, G.; Yang, Z.; and Kitsuregawa, M. 2012. Recommending related microblogs: A comparison between topic and wordnet based approaches. In *AAAI*.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Go, A.; Bhayani, R.; and Huang, L. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1:12.

He, H.; Gimpel, K.; and Lin, J. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1576–1586.

Hill, F.; Cho, K.; Korhonen, A.; and Bengio, Y. 2015. Learning to understand phrases by embedding the dictionary. *arXiv preprint arXiv:1504.00548*.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, 2042–2050.

Islam, A., and Inkpen, D. 2009. Semantic similarity of short texts. *Recent Advances in Natural Language Processing V* 309:227–236.

Kiros, R.; Zhu, Y.; Salakhutdinov, R. R.; Zemel, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, 3276–3284.

Li, Y.; McLean, D.; Bandar, Z.; O'shea, J. D.; Crockett, K.; et al. 2006. Sentence similarity based on semantic nets and corpus statistics. *Knowledge and Data Engineering, IEEE Transactions on* 18(8):1138–1150.

Metzler, D.; Dumais, S.; and Meek, C. 2007. *Similarity measures for short segments of text*. Springer.

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

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2014. word2vec.

Pang, B., and Lee, L. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval* 2(1-2):1–135.

Ritter, A.; Etzioni, O.; Clark, S.; et al. 2012. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1104–1112. ACM.

Rosenthal, S.; Nakov, P.; Kiritchenko, S.; Mohammad, S. M.; Ritter, A.; and Stoyanov, V. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval*.

Tang, J.; Wang, X.; Gao, H.; Hu, X.; and Liu, H. 2012. Enriching short text representation in microblog for clustering. *Frontiers of Computer Science* 6(1):88–101.

Tu, Z.; Hu, B.; Lu, Z.; and Li, H. 2015. Context-dependent translation selection using convolutional neural network. *arXiv preprint arXiv:1503.02357*.

Wang, Z., and Ittycheriah, A. 2015. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*.

Xu, W.; Ritter, A.; Callison-Burch, C.; Dolan, W. B.; and Ji, Y. 2014. Extracting lexically divergent paraphrases from twitter. *Transactions of the Association for Computational Linguistics* 2:435–448.

Yan, S., and Tian, W. 2015. A kind of intelligent question-answering system based on sentence similarity calculation model. *Journal of Chemical & Pharmaceutical Research* 7(3).

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 649–657.

Zhou, D.; Chen, L.; and He, Y. 2015. An unsupervised framework of exploring events on twitter: Filtering, extraction and categorization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.