# Measuring Similarity for Short Texts on Social Media

Phuc H. Duong, Hien T. Nguyen(✉), and Ngoc-Tu Huynh

Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam
{duonghuuphuc,hien,huynhngoctu}@tdt.edu.vn

**Abstract.** In this paper, we present a method for measuring semantic similarity between short texts by combining two different kinds of features: (1) distributed representation of word, (2) knowledge-based and corpus-based metrics. Then, we present experiments to evaluate our method on two popular datasets - Microsoft Research Paraphrase Corpus and SemEval-2015. The experimental results show that our method achieves state-of-the-art performance.

## 1 Introduction

Measuring semantic similarity between two short texts, e.g., news headlines, tweets or comments in public forums, plays an important role in social network analysis, sentiment and opinion analysis, summarization of posts/replies, information retrieval, etc. Since a short text is usually limited in the number of characters, context-poor, irregular or noisy, techniques in natural language processing proposed for short texts are not tailored to perform well on those tasks. Most of the proposed methods in literature exploit corpus-based or knowledge-based to compute the degree of similarity between given texts by measuring the word-to-word similarity [1]. Other approaches take the advantage of machine translation metrics [2], discourse information [3]. In [2], the authors implement a heuristic alignment algorithm to identify pairs of plagiarism sentences, then, pass them to a learning algorithm for training a classifier. The approach proposed in [3] divides sentences into elementary discourse units (EDUs), aligns EDUs, and computes the overall similarity between sentences based on aligned EDUs. Although some previous work focuses on the preprocessing phase, it still does not consider many factors, for example, the number of tokens constructs a meaning word. Hence, in this paper, we elaborately consider many aspects, as presented below, in measuring similarity between short texts and apply those to our preprocessing phase.

- One of the most challenge task in determining the similarity of words or concepts is that they usually do not share actual terms in common. Consider an example, in analyzing a text, the concepts *"Artificial Intelligence"* and *"AI"* are similar to each other in the context of computer science. In other example, *"The Pentagon"* and *"United States Department of Defense"*, the two terms are different, but similar in meaning. Therefore, our method performs named entity recognition and named entity co-reference resolution to isolate them from the texts for other steps.
- Beside named entities, the number of tokens constructing a meaning word is also importance. Much previous work considers each token as a meaning word; however,


H.T. Nguyen and V. Snasel (Eds.): CSoNet 2016, LNCS 9795, pp. 249–259, 2016.
DOI: 10.1007/978-3-319-42345-6_22

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 order to overcome this drawback, our proposed method includes a step, namely *tokenizer*, that preserves phrasal words like the case of *"pull out"*.

Furthermore, in order to make our proposed method becomes flexible, we design a model which is suitable for measuring similarity for both formal and informal texts. We also investigate three different kinds of features and show that our proposed method achieves state-of-the-art performance.

In summary, the contribution of this paper is two-fold as follow: First, we preserve phrasal words, take named entities and their co-reference relations among them into account, which were not exploited in literature; Second, we exploit two different similarity measures as features: (1) Word-embedding-based similarity, (2) Knowledge-based and corpus-based similarity; Finally, we conduct experiments to evaluate our method and show that word-embedding-based similarity superior contribution to the performance.

The rest of this paper is organized as follows. First, we present related work in Sect. 2. Section 3 presents our method and the two features for measuring similarity. Then, experimenting our method on the two popular datasets are described in Sect. 4. Finally, Sect. 5 concludes the paper.

## 2 Related Work

There have been many studies on scoring the similarity degree between two short texts. In [4], the authors propose a method which combines semantic and syntactic information in the given texts. For semantic information, this approach exploits knowledge-based and corpus-based to reflect both the meanings and the actual usage of words. For syntactic information, the method represents the given texts as word-order vector to measure a number of different words and word pairs in a different order. In [5], the authors use pointwise mutual information, latent semantic analysis and six knowledge-based methods [1] for measuring word-to-word similarity, then, conclude the degree of similarity between two texts. In [6], the authors present the discriminative term-weighting metric, known as TF-KLD, which is an improvement of traditional TF-IDF and WTMF [7]. Then, they form a feature vector from the latent representations of each text segment pair and input to SVM classification. In [8], the authors combine the longest common subsequence and skip $n$-gram with WordNet[1] similarity.

In [2], the authors re-examine 8 machine translation metrics for identifying paraphrase in two datasets, and method proposed in [9] gains the best performance. This study shows that a system only employs machine translation metrics can achieve promising results. The approach in [3] takes advantage of the elementary discourse units (EDUs) to identify paraphrase. Method in [10] presents a probabilistic model which

[1] http://wordnet.princeton.edu.

combine semantic and syntactic using quasi-synchronous dependency grammars. In [11], the authors present an unsupervised recursive auto-encoders to learn the feature vectors which contain the similarity of single word and multi-word extracted from parse trees of two text segments. In [12], the authors present two components in modular functional architecture. For the sentence modeling component, they use convolutional neural network, for the similarity measurement component, they compare pairs of regions of the sentence representations by combining distance metrics. In [13], the authors propose a kernel function which takes the advantage of search engine (e.g., Google) and TF-IDF for computing query expansion, then applies kernel function to multiply the two query expansion of two given texts to conclude the degree of similarity.

Because the basic element in constructing a text is words (tokens), the degree of similarity between two text snippets depends on the similarity between pairs of word of two given texts. There have been many approaches [1], but in this paper, we roughly classify word-to-word similarity metrics into three major types: (1) knowledge-based, (2) corpus-based, and (3) vector space representation. Methods in knowledge-based approach [14] exploit the semantic information from structure knowledge sources (e.g., WordNet, Wikipedia[2]) to measure the similarity of two given concepts. Other field of study which rely on the statistical information of concepts in large corpus, well-known methods in this approach are information content, latent semantic analysis, hyperspace analogue to language, latent dirichlet allocation. In [15], the authors present a word embedding approach using two-layer neural network with continuous bag-of-words (CBOW) or continuous skip-gram architecture. In [16], the authors consider both the proximity and disambiguation problems on word embedding method, and then they propose Proximity-Ambiguity Sensitive model to tackle them.

## 3 Proposed Method

In this section, we present our method for computing the semantic similarity between two given snippets of texts. Figure 1 presents our model for measuring of similarity between two short texts. We explain in detail our proposed method in Sects. 3.1 and 3.2.

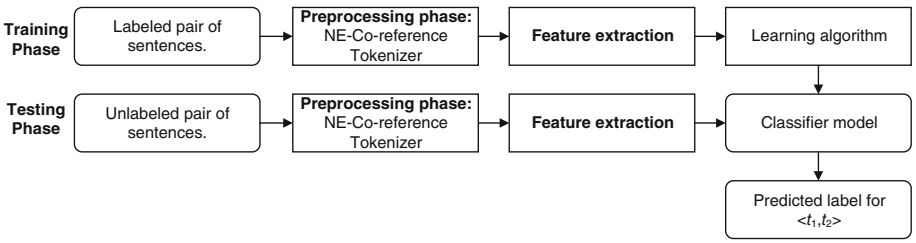


**Fig. 1.** Our proposed model of measuring similarity between short texts

[2] https://en.wikipedia.org/.

### 3.1 Preprocessing

Short texts (e.g., news title, message, tweet) often contain some special characters (e.g., dollar sign, colon, emoticon), but they do not contribute much semantic information for measurement. Therefore, we suggest to ignore those special characters in given texts but still preserve their structures.

In order to gain the best performance in computing similarity, we recognize named entities and then perform named entity co-reference resolution. A named entity often contains more than one word, e.g., *"United States"* is semantically different from *"United"* and *"States"*. To recognize named entities, we take the advantage of Wikipedia, which is an open encyclopedia contributed by a large community of users. Since Wikipedia contains named entities and common concepts (e.g., tree, data structures, algorithm), we treat those common concepts in Wikipedia as "named" entities. In reality, an entity may have more than one alias and an alias may corresponding to many entities in different context. For example, in Wikipedia, *"United States"* has up to four difference aliases {*United States of America*, *America*, *U.S.*, *USA*}, that means, all of them are similar to each other. By practice, we found out that named entity often has four tokens, thus, we propose to set a sliding window of four to get a set of all candidate named entities from given text. Next, we detect the orthographic co-reference between those recognized named entities by using rules proposed in [17]. After perform co-reference resolution step, named entities which referent to each other are grouped in co-reference chains. Finally, with the co-reference entities, we assign them a unique identifier (*"ID#"*) to make them become similar entities. Let's consider the example below:

- Obama calls on tech industry at SXSW to help solve nation's problems.[3]
- Obama, at South by Southwest, calls for law enforcement access in encryption fight.[4]

By using *exact match* and *equivalent* rules to perform named entity co-reference resolution, there are two pairs of co-reference named entities, which are {*"Obama"*$_1$, *"Obama"*$_2$} and {*"SXSW"*$_1$, *"South by Southwest"*$_2$}. Therefore, we replace them to *"ID#"* format, the input sentences become:

- *ID*1 calls on tech industry at *ID*2 to help solve nation's problems.
- *ID*1, at *ID*2, calls for law enforcement access in encryption fight.

As mentioned in Sect. 1, if we only consider special characters and named entities are not enough, because the assumption of word contains one token is weak. Example, consider the following words in the same context, *"cut a rug"* and *"dance"*, if we split the white space, the meaning of them is not similar. However, they are the same meaning, because *"cut the rug"* is a culturally understood meaning of *"dance"*, also known as *idiom*. We can see that not only phrasal verbs, but also idioms and many other cases, thus, in preprocessing phase, we need to recognize all of them, and this task is a subtask of tokenizer. To perform this task, we use Wiktionary[5], a free dictionary contributed

[3] http://usat.ly/1pla4oI.
[4] http://nyti.ms/1QS47Ga.
[5] http://en.wiktionary.org/.

by community members, contains 644,966[6] entries including 547,056 with gloss definitions. We apply longest matching algorithm, which find the first best matching between series of tokens and dictionary. Then, marking them with underscore symbol between tokens to group them together, for instance, *"look_after"* and *"take_care_of"*.

After perform named entity recognition and tokenizer, we have finished preprocessing phase, and two given texts are now ready for computing semantic similarity.

### 3.2 Feature Extraction

In this section, we systematically introduce two features in measuring semantic similarity for given short texts. They are (1) word-embedding-based similarity, (2) knowledge-based and corpus-based similarity.

**Word-Embedding-Based Similarity ($Sim_{word\text{-}embedding}$).** Before explain the method to score the similarity of given texts, we introduce an approach for measuring the degree of similarity between two words by learning distributed representation of words. The distributional hypothesis states that the words are similar meanings if they are in similar context. Therefore, we take advantage of the simplified neural network skip-gram model, which predicts surrounding words given the current word by sliding a context window along the text and uses back-propagation to train the network. Figure 2 shows the main idea of skip-gram model.

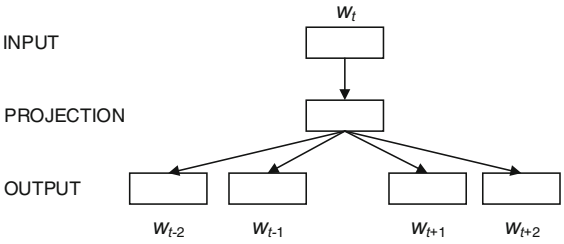


**Fig. 2.** Skip-gram model [18]

Given a sequence of words $\{w_1, w_2, \ldots, w_T\}$, the training objective of skip-gram model is maximizing the average log probability. In Eq. (1), $c$ is the size of the training context. The larger the context size is; the higher accuracy the model will be. However, it does expenses more training time. Therefore, in order to overcome the time consuming problem but maintain the accuracy, we use negative-sampling as softmax function. Unlike hierarchical softmax function, instead of considering all context of $w$ at each iteration, negative-sampling considers a few words by randomly chosen from context,

[6] This information is generated from the 03 March 2016 dump.

thus it can reduce training time. In experiment, we use the Google News dataset containing 100 billion words to train our skip-gram model.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0}\log p\left(w_{t+j}|w_t\right) \tag{1}$$

After calculating similarity between words using word embedding, we present a metric to compute the similarity of two given texts. We have three sub-tasks in this phase: (1) create a joint word set, (2) create semantic vectors, (3) normalize and compute the distance between vectors. Let's consider the example below:

- I am studying Artificial Intelligence.
- I learn AI with my friends.

First, we create a joint word set $W$ contains all distinct words in given texts, denoted by $T_1$ and $T_2$, as proposed in [4]. With the example above, after go through preprocessing phase, the $W$ would be $W = \{\text{I, am, study, } ID1\text{, learn, } ID1\text{, with, my, friend}\}$. Because $W$ is directly derived from given texts, we use it as standard semantic vector for comparing with $T_i$. Second, we represent $T_1$ and $T_2$ as semantic vectors, denoted by $V_i$. The $V_i$'s length is equal $W$, and each element of $V_i$ will be assigned as the following rules:

– **Rule 1:** if $w_i$ appears in $T$, assign 1 to $v_i$ position in $V$.
– **Rule 2:** unless, compute the similarity score $s$ between $w_i$ and each word in $T$. If $s$ exceeds a preset threshold $\tau$, then assign $s$ to the considering position in $V$, otherwise, assign 0. When $s$ is near to 0, it would better to assign 0 to $v_i$ because it does not contribute valuable information.

Depend on the length of given texts, we can keep or ignore function words. In case of short texts, we recommend to maintain function words, but we can assure that they do not affect the whole meaning of texts due to our preset threshold. Finally, after having two semantic vectors, the similarity of two texts is computed by cosine coefficient of those vectors. The output of Eq. (2) has already been normalized between 0 and 1. As the value nears 1, the given texts are more similar, and vice versa.

$$Sim_{word-embedding}(T_1, T_2) = \cos(\theta) = \frac{V_1 \cdot V_2}{\|V_1\|\|V_2\|} \tag{2}$$

**Knowledge-Based and Corpus-Based Similarity ($Sim_{knowledge\text{-}and\text{-}corpus}$).** In previous section, we have presented an approach using neural language model to represent word as semantic vector. In this section, we present a method which exploits knowledge base and corpus. With knowledge-based method, we use a semantic graph structure (e.g., WordNet), in which words (also known as *concepts*) are organized as a hierarchy, to measure the relatedness between words. The meaning of relatedness is more general than similarity, for example, *"car"* and *"wheel"* are not similar, but between them exists *part-of* relationship. In WordNet, concepts are grouped to *synsets*, which means *sets of synonyms*, and represented as graph structure, together with six types of relationship: (1) synonymy, (2) antonymy, (3) hyponymy, (4) meronymy, (5) troponomy and

(6) entailment. In order to identify the relatedness, we take into account the path between two concepts, its length reflects the degree of relationship. However, only considering the path length may lose the generalization, we also consider the lowest common subsumer (LCS) [19] concept, which is the nearest to the compared concepts. Although we have looked for the LCS of two concepts, it does not reflect the contribution of both LCS and two concepts. Therefore, we combine the statistical technique on large corpus, e.g., Brown corpus[7]. As proposed in [20], first, we form a set of LCSs that subsume two concepts, then, we compute the probability that each element in LCSs set appears in the corpus and get the maximum probability. This metric denoted by $Sim_{F2}$, as Eq. (3).

Though WordNet is a good choice in many semantic metrics, it does not cover all up-to-date concepts. For instance, with the growth of social networks, there are many new concepts created in every day, e.g., *"selfie"*, *"emoji"*. Therefore, to overcome this drawback, when a concept not found in WordNet, we will find it in Wiktionary. However, the structure of Wiktionary is not well for finding LCS, we use another metric, called gloss-based. Each concept in Wiktionary comes with descriptions, called as *gloss texts*. The method proposed in [21] is based on the assumption that the level of overlapping between gloss texts of concepts is proportional to the level of similarity. After calculating similarity between words based on knowledge and corpus, we represent given short texts as vectors and compute the similarity between them using Eq. (3).

$$Sim_{F2}(c_1, c_2) = \begin{cases} \max_{c \in LCS(c_1, c_2)} \left[ -\log p(c) \right], \{c_1, c_2\} \in \text{WordNet} \\ gloss(c_1) \cap gloss(c_2), \text{otherwise} \end{cases} \tag{3}$$

## 4 Experiments

### 4.1 Datasets

We conduct experiments on two datasets: (1) Microsoft research paraphrase corpus (MSRP) [22], and (2) SemEval-2015[8]. The MSRP is a well-known dataset for the problem of paraphrase identification, containing pairs of labeled sentences, if two sentences are paraphrase, the label will be 1 and vice versa. This dataset can be applied to supervised learning approaches, the training set contains 4,076 sentences (2,753 positive, ~67.5 %), and the test set contains 1,725 sentences (1,147 positive, ~66.5 %). The SemEval is series of evaluation of computational semantic analysis systems. The SemEval-2015 dataset also contains two parts, training and test set. Both sets are divided into five domains, in which, each pair of texts is manually semantic annotated by human, in range of [0,5]; the score is proportional to the similarity degree.

[7] https://en.wikipedia.org/wiki/Brown_Corpus.
[8] http://alt.qcri.org/semeval2015/task2/.

### 4.2 Experimental Results

In order to measure the performance of our proposed method, we train our model by using support vector machine learning algorithm on MSRP and SemEval-2015 training sets, and then, test the model on two datasets respectively. However, to show the contribution of the presented features, we perform independently two training and testing tasks: (1) only consider $Sim_{word\text{-}embedding}$, denoted by $F_1$; (2) combine $Sim_{word\text{-}embedding}$ with $Sim_{knowledge\text{-}and\text{-}corpus}$, denoted by $F_1 + F_2$.

In Table 1, we present the performance of our method by evaluating the contribution of the features on two testing sets, but with SemEval-2015 dataset, we only show the best result of all domains. Tables 2 and 3 present our experiment results on two datasets in comparison to other approaches. With MSRP dataset, we use the accuracy to present the performance of our system, with SemEval-2015 dataset, we use the Pearson correlation coefficient.

**Table 1.** Evaluate the combination of the presented features

| Features | Datasets | |
|---|---|---|
| | MSRP *(accuracy)* | SemEval-2015 *(ρ)* |
| $F_1$ | 0.83 | 0.89 |
| $F_1 + F_2$ | 0.82 | 0.87 |

**Table 2.** Experiment results on MSRP dataset

| Method | Accuracy |
|---|---|
| Madnani*et al.* [2] | 77.4 % |
| Ji and Eisenstein [6] | 80.4 % |
| Milajevs *et al.* [25] | 73.0 % |
| Nguyen *et al.* [23] | 80.7 % |
| **This paper** | **83.6 %** |

**Table 3.** Experiment results on SemEval-2015

| Domain | Sultan *et al.* [24] | This paper | Feature |
|---|---|---|---|
| Answer-forums | 0.73 | **0.75** | $F_1$ |
| Answers-students | 0.78 | **0.79** | $F_1 + F_2$ |
| Belief | 0.77 | **0.76** | $F_1$ |
| Headlines | 0.84 | **0.89** | $F_1$ |
| Image captions | 0.86 | **0.87** | $F_1 + F_2$ |

With the experiment results in Table 1, we can see that the contribution of $F_1$ does yield the best performance on two datasets. When we combine $F_1$ with $F_2$, the results are not quite good, because WordNet does not contain all up-to-date concepts, thus we combine with gloss-based method on Wiktionary. By this combination, it may increase

the noise in our model, as gloss-based method does not perform well when the gloss texts are short, and the part-of-speech of words may also affect the selection of appropriate gloss texts.

In Table 2, the experiment results on MSRP dataset shows that our method yields a better result than our proposed method in [23] when using $Sim\text{-}_{word\text{-}embedding}$ feature. The main difference between this method and the previous method is how to measure word-to-word similarity. In [23], Nguyen *et al.* use WordNet as knowledge base with information content metric, but WordNet can cover about 64.5 % words on MSRP dataset. On the other hand, in this study, we use the word embedding model to exploit the context surrounding words and combine with tokenizer in preprocessing phase to conclude the level of similarity, and this overcomes the previous drawback. In Table 3, with the results on SemEval-2015, our performance is slightly better than the method proposed in [24]. In [24], the authors gained the best experiment results when using $S_1$ method, which is quite similar to our method, but differs from the training set for word-similarity metric.

## 5 Conclusion

We have presented our method for measuring the semantic similarity between short texts on social media by independently evaluating and combining the two different kinds of features: (1) distributed representation of word, (2) knowledge-based and corpus-based metrics. The main contribution of our work can be summarized as follow:

- First, by performing the named entity co-reference resolution, we have increased the system performance because of removing the influence of them. Besides that, we have showed the assumption "each token is a meaning word" is weak, thus, we do tokenizer in our preprocessing phase.
- Second, using skip-gram model to represent word as semantic vector to measure the semantic similarity between words, instead of only relying on semantic graph structure (WordNet) and corpus (Brown Corpus).
- Third, by evaluating the contribution when combines the two features on MSRP and SemEval-2015 datasets, we realize that word embedding feature performs better than another feature, and also significantly improves the performance of our method.
- Finally, our proposed method is quite easy for re-implementing and evaluating other datasets, and can also apply to many applications of natural language processing with an acceptable performance.

## References

1. Duong, P., Nguyen, H., Nguyen, V.: Evaluating semantic relatedness between concepts. In: IMCOM, pp. 20:1–20:8. ACM (2016)
2. Madnani, N., Tetreault, J., Chodorow, M.: Re-examining machine translation metrics for paraphrase identification. In: HLT-NAACL, pp. 182–190 (2012)
3. Bach, N., Nguyen, M., Shimazu, A.: Exploiting discourse information to identify paraphrases. Expert Syst. Appl. **41**(6), 2832–2841 (2014)

4. Li, Y., McLean, D., Bandar, Z., O'Shea, J., Crockett, K.: Sentence similarity based on semantic nets and corpus statistics. IEEE Trans. Knowl. Data Eng. **18**(8), 1138–1150 (2006)
5. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI, pp. 775–780 (2006)
6. Ji, Y., Eisenstein, J.: Discriminative improvements to distributional sentence similarity. In: EMNLP, pp. 891–896 (2013)
7. Guo, W., Diab, M.: Modeling sentences in the latent space. ACL **1**, 864–872 (2012)
8. Kozareva, Z., Montoyo, A.: Paraphrase identification on the basis of supervised machine learning techniques. In: Salakoski, T., Ginter, F., Pyysalo, S., Pahikkala, T. (eds.) FinTAL 2006. LNCS (LNAI), vol. 4139, pp. 524–533. Springer, Heidelberg (2006)
9. Snover, M., Madnani, N., Dorr, B., Schwartz, R.: TER-Plus: paraphrase, semantic, and alignment. Mach. Transl. **23**(2–3), 117–127 (2009)
10. Das, D., Smith, N.: Paraphrase identification as probabilistic quasi-synchronous recognition. In: Su, K.-Y., Su, J., Wiebe, J. (eds.) ACL/IJCNLP, pp. 468–476 (2009)
11. Socher, R., Huang, E., Pennington, J., Ng, A., Manning, C.: Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In: Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K. (eds.) NIPS, pp. 801–809 (2011)
12. He, H., Gimpel, K., Lin, J.: Multi-perspective sentence similarity modeling with convolutional neural networks. In: Lluís, M., Callison-Burch, C., Pighin, D., Marton, Y. (eds.) EMNLP, pp. 1576–1586 (2015)
13. Sahami, M., Heilman, T.: A web-based kernel function for measuring the similarity of short text snippets. In: Carr, L., Roure, D., Iyengar, A., Dahlin, M. (eds.) WWW, pp. 377–386 (2006)
14. Witten, I., Milne, D.: An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In: Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: An Evolving Synergy, pp. 25–30. AAAI Press, Chicago (2008)
15. Mikolov, T., Chen, K., Corrado, G.: Efficient estimation of word representations in vector. In: Proceedings of International Conference of Learning Representations (2013)
16. Qiu, L., Cao, Y., Nie, Z., Yu, Y.: Learning word representation considering proximity and ambiguity. In: Brodley, C., Stone, P. (eds.) AAAI, pp. 1572–1578 (2014)
17. Bontcheva, K., Dimitrov, M., Maynard, D., Tablan, V., Cunningham, H.: Shallow methods for named entity coreference resolution. In: Chaınes de références et résolveurs d'anaphores, Workshop TALN (2002)
18. Mikolov, T., Sutskever, I., Chen, K., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
19. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics, pp. 133–138 (1994)
20. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI, pp. 448–453 (1995)
21. Lesk, M.: Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In: Proceedings of the 5th Annual International Conference on Systems Documentation, pp. 24–26 (1986)
22. Dolan, W., Brockett, C.: Automatically constructing a corpus of sentential paraphrases. In: Proceedings of IWP (2005)
23. Nguyen, H.T., Duong, P.H., Le, T.Q.: A multifaceted approach to sentence similarity. In: Huynh, V.-N., Inuiguchi, M., Demoeux, T. (eds.) IUKM 2015. LNCS, vol. 9376, pp. 303–314. Springer, Heidelberg (2015). doi:10.1007/978-3-319-25135-6_29

24. Sultan, M., Bethard, S., Sumner, T.: DLS@ CU: Sentence similarity from word alignment and semantic vector composition. In: Proceedings of the 9th International Workshop on Semantic Evaluation, pp. 148–153 (2015)
25. Milajevs, D., Kartsaklis, D., Sadrzadeh, M., Purver, M.: Evaluating neural word representations in tensor-based. In: EMNLP, pp. 708–719 (2014)