社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

A Study on Efficient Searching Top-*k* Semantic Similar Sentences

Yanhui GU† Zhenglu YANG† and Masaru KITSUREGAWA†

† Institute of Industrial Science, The University of Tokyo 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan

E-mail: † {guyanhui,yangzl,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract Semantic similarity measure between sentences is an important issue in many applications, such as, text mining, Web page retrieval, dialogue systems, and so forth. Although it has been explored for several decades, most of these studies focus on how to improve the effectiveness of the problem. In this paper, we address the efficiency issue, i.e., for a given sentence collection, how to efficiently discover the top-*k* most semantic similar sentences to the query. It is a very important issue for real applications while existing state-of-the-art strategies cannot satisfy the performance requirement of the users. We introduce a general framework to tackle the issue, in which several efficient strategies are proposed. Extensive experimental evaluations demonstrate that our approach outperforms the state-of-the-art methods.

Keyword semantic similarity, query aggregation, Top-*k*

1. Introduction

In many applications, searching semantic similar sentences is an important issue, such as text mining, Web page retrieval, and dialogue systems, etc. The framework for such application is: given a collection of sentences, the system gives the most (i.e., top-k) semantically similar sentences to a query.

The problem can be solved by firstly measuring the semantic similarity score between the query and each sentence in the data collection using the state-ofthe-art techniques [6, 9, 11, 13, 15], and then sorting them with regard to such similarity score and finally returning the top-k ones. However, when the size of the data collection increases, the scale of the problem has dramatically increased.

Note that almost all the previous studies focus on improving the effectiveness of the problem while in this paper we firstly propose the strategy which addresses the efficiency issue in the literature. Secondly, most of the previous approaches are threshold-based, i.e., the similarity threshold is predefined. But it is the case that such threshold is difficult for user to predefine because the returned results are sensitive with the threshold value, i.e., if the threshold is too small, few results will be returned while too many results will be returned when the threshold is set to be too large. Therefore, searching top-k similar sentences seems to be very challenging.

Traditionally, techniques for measuring similarity between long texts (e.g., documents) have centered on analyzing co-occurred words [14]. Such methods are usually effective when dealing with long texts because similar long texts usually contain a degree of sharing words. However, in short texts (e.g., sentences), word co-occurrence may be rare or even null. This problem poses a difficult computational challenge that we cannot apply the document similarity measurement strategies in sentences directly.

To remedy such problem, extensive studies have been explored based on the feature of sentence and can be classified into the following main groups: (1) knowledge-based strategies [15, 11]; (2) corpus-based strategies [6]; (3) common word order based strategies [9, 6]; (4) hybrid strategies [6, 9]. Since words are the components of sentences, word similarity is a non neglectable feature when we measure the sentence similarity [6, 9, 11].

There are two questions we are facing when search top-k semantic similar sentences. (1) Scalability. Traditional approaches [9, 6] will compute all the similarity between the query and candidates in the data collection. However, it is time consuming when searching top-k results in rather large data collection. If more features are taken into account, it will become more worse. (2) Efficiency. For searching top-k, we need not know all the similarity score between query and the candidates in the data collection, i.e., we only need know their orders. We regard that the candidates in the data collection will following some orders which are based on the similarity strategies. Therefore, if we preprocess the candidates, the first result will be returned to the user almost instantly and the more results will be output following the execution time. In this paper, we aim to address such challenges. Precomputing all the similarity scores of candidates may tackle the problem we mentioned [12]. However, data is frequently changed, e.g.,Web. Re-computing and processing all the similarity scores is time and space consuming. In addition, evaluating all the similarity scores also seems unpractical because we cannot predict a user's query. Therefore, computing the similarity on-the-fly seems a practical solution.

It should be noted that, we do not discuss the other optimization strategies, e.g., caching strategies, cloud computing which are out of the scope of our paper.

The contribution of this paper are as follows:

(1) We propose a strategy to tackle the efficiency of searching top-k semantic similar sentences, which is different from previous works which focus on the effectiveness aspect. We take two representative kings of measurement strategies, i.e., string similarity strategies and semantic similarity strategies. More similarity measurement strategies can be incorporated into the proposed framework which will be discussed later in this paper.

(2) We propose efficient strategies to extract the top-k results. For each similarity measure, we introduce a corresponding strategy to minimize the number of candidates to be evaluated. A rank aggregation method is presented to optimally obtain the top-k results when assembling the features.

(3) We conduct comprehensive experiments and evaluate the query efficiency of the proposed strategies. The results show that the proposed strategies outperform the state-of-the-art methods.

2. Problem Statement

Formally, for a query sentence Q, finding a set of k sentences P in a given sentence collection S which are most similar to Q, i.e., $\forall p \in P$ and $\forall r \in (S - P)$ will yield sim(Q, p) ≥ sim(Q, r). To measure the similarity sim(Q, P) between two sentences, we apply the state-of-the-art strategies by assembling multiple similarity metric features together [9, 6]. Because we focus on tackling the efficiency issue in this paper, we select several representative features from the main categories based on the framework which has been proposed in [6].

2.1. Similarity Measurement Strategies

String-based Similarity

String similarity measures the difference of syntax between strings. An intuitive idea is that two strings are similar to each other if they have enough common subsequences (i.e., LCS [4]). We focus on three representative string similarity measurement strategies, i.e., NLCS, NMCLCS1 and NMCLCSn1 which are denoted as SimNLCS, SimMCLCS1 and SimMCLCSn in the following.

NLCS LCS is a common string similarity measurement strategy and it measures the longest common subsequence of two strings. The similarity score is the length of LCS normalized by the product of the length of two strings. For two strings w_i, w_j , Formula 1 tells us how to evaluate their NLCS similarity.

We take two strings *abacd* and *acadb* as an example. These two strings have common subsequence *a, aa, ad* or *aad* while *aad* is the longest. So LCS(*abacd, acadb*)=3 and SimNLCS(*abacd, acadb*)= 9/25 .

NMCLCS1 NMCLCS1 measures the similarity between two strings where they have the maximal consecutive LCS from the first character which tells us whether these two strings have the maximal consecutive prefix substring. Different from NLCS, NMCLCS1 has two properties: (1) The longest common subsequence in NMCLCS1 should be consecutive; (2) The two strings should have the same first character. So the NMCCLS1 similarity between w_i and w_j is indicated as the following formula.

We take examples to illustrate how NMCLCS1 works. (1) Two strings *abcd* and *abed* have the longest common subsequence *abd*, but not consecutive. Therefore MCLCS1(*abcd,abed*)=*ab* and SimNMCLCS1(*abcd,abed*)= 4/16 . (2) Although *abcd* and *bbcd* have the longest common subsequence bcd and also be consecutive, they are different in the first character. So MCLCS1(*abcd,bbcd*)=0 and SimNMCLCS1(*abcd,bbcd*)=0;

NMCLCSn Similar to NMCLCS1, NMCLCSn measures the similarity between two strings where they have the maximal consecutive common subsequence. The only difference here is that it starts at any position. We show the NMCLCSn similarity measurement strategy of two strings wi and wj as follows:

For better under how NMCLCSn works, we take two strings *abcd* and *bbcd* as an example. Because NMCLCSn does not take the first character into account, so MCLCS(*abcd*,*bbcd*)=*bcd* and SimNMCLCSn(*abcd*,*bbcd*)= 9/16.

Corpus-based Similarity

Corpus-based similarity measurement strategy is to recognize the degree of similarity between words using large corpora [8]. There are several kinds of strategies: PMI (PointwiseMutual Information) [16] applies the search engine to gather the existence information from the Web; LSA (Latent Semantic Analysis) [7, 8] analyzes a large corpus of natural language text and generates a representation that captures the similarity of words and text passages; HAL (Hyperspace Analogues to Language) [1] uses lexical co-occurrence to produce a high-dimensional semantic space to capture the semantic information. There also exist some other strategies, e.g., chi-square, log-likelihood, and so forth. In this paper, we use the SOC-PMI (Second Order Co-occurrence PMI) word similarity method [5, 7] that uses PMI to sort lists of important neighbor words in a context window of the two target words from a large corpus. The underlying idea is that the neighbors of the two target words have the abundant semantic context with each other and aggregate the more important information. PMI for two words w_i,w_j is defined as follows:

$$f^{pmi}(w_1,w_2) = log_2 \frac{f(w_1,w_2)\times m}{f(w_1)f(w_2)}$$

Common Word Order

Common word order measures the similarity between the common words of sentence pair. If two sentences have some words in common, we can measure how similar the order of the common words in these two sentences. For example, we have two sentences P and Q, and there are δ words appear in both sentences. We assign a unique index number for each common word in P from 1 to δ, that is X = x1, ..., xδ and them mark the index number to the common word Y in Q based on such unique index number. So the common word order similarity of two sentence is as follows:

$$S_o = 1 - \frac{|x_1 - y_1| + |x_2 - y_2| + ... + |x_\delta - y_\delta|}{|x_1 - x_\delta| + |x_2 - x_{\delta-1}| + ... + |x_\delta - x_1|}.$$

2.2. A General Framework

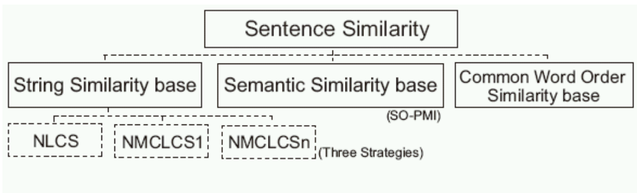

Fig. 1. The concept of the general framework.

To measure the overall similarity between two sentences, a general framework is needed to incorporate all the similarity measurement strategies. From the previous works, [6] is the most comprehensive approach which incorporates string similarity, semantic similarity and common word order similarity into its framework. Fig. 1 shows the concept of the framework. The string similarity base is composed of three different similarity measurement strategies, i.e., NLCS, NMCLCS1 and NMCLCSn. The final sentence pair similarity is the aggregation of string similarity, semantic similarity and common word order similarity. The common words plays syntax information in sentence pair. However the number of common word is rare and [6] demonstrates that common word order similarity plays a less important role in semantic processing of short text, e.g., sentence. Therefore, they ignore the such similarity in the implementation of their framework. Fig. 2 is the implementation of the top-k similar sentences searching framework.

If we want to obtain the top-k semantic similar sentences, we should calculate all the query sentence pairs and sort the overall sentence similarity score list and finally obtain top-k value. We demonstrate how [6] find the top-3 result in Fig. 2. However, accessing and computing all the sentences in the data collection is time consuming and it is inefficient when the data collection is large. Therefore, an efficient searching top-k semantic similar sentences strategy is needed.

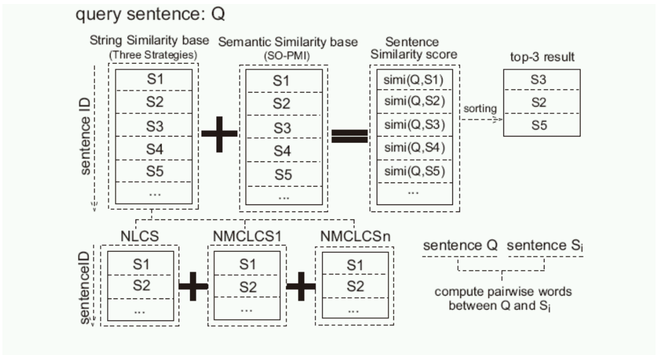

Fig. 2. The implementation of the top-*k* similar sentences searching framework.

3. Proposed Approaches

In this paper, we propose an efficient framework based on [6] which is the most effective in the literature of measuring sentence semantic similarity. Our key idea is by building appropriate index in the preprocessing, we only need access a small part but not the whole data collection. The query sentence and each candidate in the data collection are sent to two modules (i.e., String similarity evaluator, Corpus-based similarity evaluator) respectively, to obtain the corresponding similarity score. Then, the scores from different modules are assembled and ranked, resulting in the final ranked list. We introduce the best first search strategies of string similarity and semantic similarity evaluator, respectively. After that, the final assembled framework will be introduced. There are many studies on exploring how to efficiently search top-k similar words with respect to string similarity [19]. In this paper, we apply NLCS, NMCLCS1, NMCLCSn as string similarity features.

3.1. Assembling Similarity Features

Our task is to find the top-k similar semantic sentences.We introduce an efficient approach to hasten the process of searching top-k similar sentences, based on the rank aggregation algorithm [2]. For example, there are five sentence in the data collection. For query Q: My father likes his work at Toyota. Based on two different similarity score list,

we obtain two lists as shown in Fig. 4. Gray rectangles in each list denote the sentence ID and the similarity score behinds it. In [2], they tell us that if a candidate in the data collection whose similarity score Sim(Q, S_i)threshold, there is at least for one feature that Simfeature(Q, S_i) threshold. We use an example to illustrate the process which is figured in Fig. 4. In the first iteration, we cannot obtain the top-1 result for either similarity score is smaller than the threshold. We put the similarity score into the max queue. The top-1 result can be outputted because S3 : 0.91, threshold : 0.885. In addition, we can obtain top-3 results in the third iteration. However, the sentences are composed of words and we can also apply searching top sentences strategies in words which has been proposed in [18]. The remaining processes can be executed in s same way as Fig. 4 shows.

4. Related Work

Measuring the semantic similarity between sentences is not similar with the methods of measuring the similarity between documents [3] and words. Sentence is shorter than document but longer than individual words. There is less work related to the measurement the semantic similarity between sentences. The methods of measurement can be classified into the following major categories: word co-occurrence or vector-based document model methods [10], corpus-based methods [6–8], hybrid methods [16, 9]. However, the document model methods are not very effective when we measure the sentence level similarity. The corpus based methods uses the outside resource to measure the semantic similarity. The hybrid methods fuse two or more methods into a uniform model, e.g., corpus based and document model based, corpus-based and knowledge-based, etc. However, all the approaches above is not take the efficiency into account. When we search top-k semantic similar sentence, the methods above should access all the sentences in the data collection. To the best of our knowledge, we firstly propose a strategy on searching top-k semantic similar sentence in the literature. Our work is similar with [18], while [18] focus on word level, i.e., their approach in on searching top-k semantic similar words but not sentences.

5. Conclusion and Future Work

In this paper, we proposed an efficient searching top-k semantic similar sentences strategy based on a state-of-the-art framework which has been proposed in [6]. This is the first work in searching semantic similar in large data collection. Several efficient best search strategies are proposed to tackle the efficiency issue in the traditional similarity measurement approach. Our experimental evaluation demonstrate the efficiency on searching top-k semantic similar sentences. In the future, we will research on other similarity measurement strategies to further evaluate the efficiency and effectiveness.

References

1. C. Burgess, K. Livesay, and K. Lund. Explorations in context space: words, sentences, discourse. Discourse Processes, 1998.

2. Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In PODS, 2001.

3. Vaseleios Hatzivassiloglou, Judith L. Klavans, and Eleazar Eskin. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In EMNLP/VLC, 1999.

4. D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. Commun. ACM, 1975.

5. Aminul Islam and Diana Inkpen. Second order co-occurrence PMI for determining the semantic similarity of words. In LREC, 2006.

6. Aminul Islam and Diana Inkpen. Semantic text similarity using corpus-based word similarity and string similarity. ACM Transactions on Knowledge Discovery from Data, 2008.

7. T. Landauer and S. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychological Review, 1997.

8. T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. Discourse Processes, 1998.

9. Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett. Sentence similarity based on semantic nets and corpus statistics. IEEE Trans. on Knowl. and Data Eng., 2006.

10. Charles T. Meadow. Text information retrieval systems. Academic Press, 1992.

11. Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In AAAI, 2006.

12. Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In EMNLP, 2009.

13. Mehran Sahami and Timothy D. Heilman. A web-based kernel function for measuring the similarity of short text snippets. In WWW, 2006.

14. Gerald Salton, editor. Automatic text processing. Addison-Wesley Longman Publishing Co., Inc., 1988.

15. George Tsatsaronis, Iraklis Varlamis, and Michalis Vazirgiannis. Text relatedness based on a word thesaurus. J. Artif. Intell. Res. (JAIR), 2010.

16. Peter D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In EMCL, 2001.

17. P. Wiemer-Hastings. Adding syntactic information to lsa. In Proceedings of the 22nd Annual Conference of the Cognitive Science Society, 2000.

18. Zhenglu Yang and Masaru Kitsuregawa. Efficient searching top-k semantic similar words. In IJCAI, 2011.

19. Zhenglu Yang, Jianjun Yu, and Masaru Kitsuregawa. Fast algorithms for top-k approximate string matching. In AAAI, 2010.