Towards Efficient Similar Sentences Extraction

Yanhui Gu, Zhenglu Yang, Miyuki Nakano, and Masaru Kitsuregawa

Institute of Industrial Science, The University of Tokyo, Japan
{guyanhui,yangzl,miyuki,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. Similar sentences extraction is an essential issue for many applications, such as natural language processing, Web page retrieval, question-answer model, and so forth. Although there are many studies exploring on this issue, most of them focus on how to improve the effectiveness aspect. In this paper, we address the efficiency issue, i.e., for a given sentence collection, how to efficiently discover the top-k semantic similar sentences to a query. The issue is very important for real applications because the data becomes huge and the existing state-of-the-art strategies cannot satisfy the users' performance requirement. We propose efficient strategies to tackle the problem based on a general framework. Extensive experimental evaluations demonstrate that the efficiency of our proposal outperforms the state-of-the-art approach.

Keywords: semantic similarity, top-k, rank aggregation.

1 Introduction

Searching semantic similar sentences is an essential issue because it is the basis of many applications, such as snippet extraction, image retrieval, question-answer model, document retrieval, and so forth [18,3,11,2,13]. From a given sentence collection, this kind of queries ask for those sentences which are most semantically similar to a given one.

The problem can be solved as follows: we firstly measure the similarity score between the query and each sentence in the data collection using the state-of-the-art techniques [7,10,12,14,15], then sort them with regard to the score and finally return the top-k ones. Almost all the previous studies focus on improving the effectiveness aspect (i.e., precision) and the datasets conducted are small. However, when the size of the data collection increases, the scale of the problem will dramatically increase and the state-of-the-art techniques will be impractical. As far as we know, this paper is the first study that aims to address the efficiency issue in the literature. Moreover, most of the previous strategies applied the threshold-based method [7,10,15] that a threshold is set to filter out those dissimilar sentences. However, this threshold is difficult for users to determine. For real applications (e.g., Google), users may prefer the top-k results.

There are mainly four kinds of techniques to measure the similarity between sentences: (1) knowledge-based strategy [15,12]; (2) corpus-based strategy [7]; (3) syntax based strategy [10,7]; and (4) hybrid strategy [7,10]. All of these

H. Yin et al. (Eds.): IDEAL 2012, LNCS 7435, pp. 270–277, 2012.

works, however, is time consuming when testing the candidates for top-k similar sentences extraction. To tackle this issue, we introduce efficient strategies to evaluate as few candidates as possible. Moreover, we aim to progressively output the top-k results, i.e., the top-1 result should be output almost instantly, then the top-2 and more results will be obtained as the execution time becomes longer. This satisfies the requirement of the real applications [5]. As such, these issues are the challenges of the paper, which have not been studied before. Our contributions of this paper are listed as follows:

We propose to tackle the efficiency issue for searching top-k semantic similar sentences, which is different from previous works that focus on the effectiveness aspect. Based on the most comprehensive work [7], we introduce the optimization techniques and improve the efficiency. For each similarity measurement, we introduce a corresponding strategy to minimize the number of candidates to be evaluated. A rank aggregation method is introduced to progressively obtain the top-k results when assembling the features. We conduct experiments and evaluate the performance of the proposed strategies. The results show that the proposed strategies outperform the state-of-the-art method.

2 Problem Statement

The issue we aim to tackle is to extract the top-k similar sentences to a query. 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 $sim(Q, p) \geq sim(Q, r)$.

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 [10,7]. Because we focus on tackling the efficiency issue in this paper, several representative features are selected based on the framework in [7]. Note that a sentence is composed of a set of words and therefore, the similarity score between two sentences is the overall scores of all the word pairs whose components belong to each sentence, respectively [7]. As such, we introduce the representative word similarity in the next section.

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 [6]). We focus on three representative string similarity measurement strategies, i.e., NLCS, NMCLCS_1 and NMCLCSn[1] which are denoted as Sim_{NLCS}, Sim_{NMCLCS_1} and $Sim_{NMCLCSn}$ [7]. The following are the formulas:

$$Sim_{StringStrategy}(w_i, w_j) = \frac{length(StringStrategy)^2}{length(w_i)length(w_j)} \quad (1)$$

[1] NLCS: Normalized Longest Common Substring, NMCLCS_1: Normalized Maximal Consecutive LCS starting at character 1, NMCLCSn: Normalized Maximal Consecutive LCS starting at any character n [7].

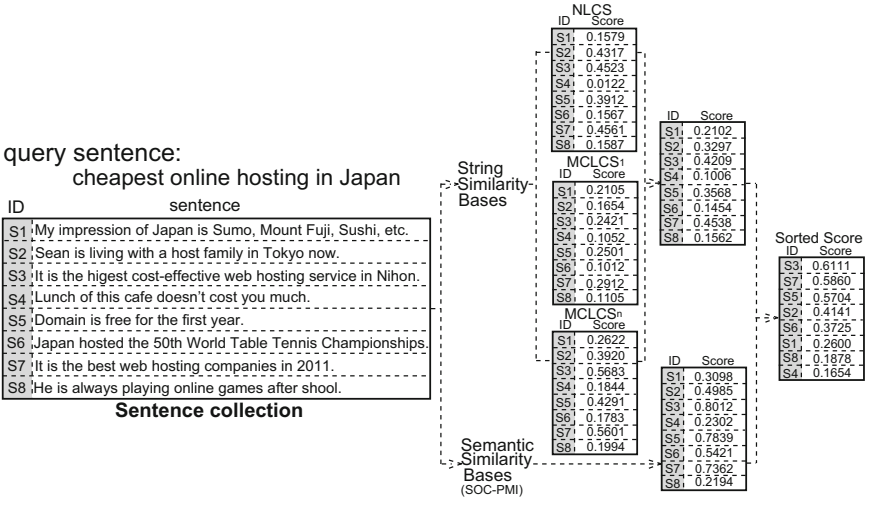


Fig. 1. The implementation of the top-k similar sentences searching framework

Here StringStrategy are NLCS, MCLCS_1 and MCLCS_n.

Corpus-Based Similarity. Corpus-based similarity measurement is to recognize the degree of similarity between words using large corpora [9]. There are several kinds of strategies: PMI [16], LSA [8], HAL [1], chi-square, and so forth. In this paper, we use SOC-PMI (Second Order Co-occurrence PMI) [7] which employs PMI while taking into account important neighbors in the context windows of the two words. The intuitive idea is that the neighbors have abundant semantic context and should be considered. PMI [16] is defined as follows:

$$f^{pmi}(w_i, w_j) = log_2 \frac{f(w_i, w_j) \times m}{f(w_i) f(w_j)} \tag{2}$$

where $f(w_i)$ is the frequency of the word w_i in the corpus, and m is the size of the corpus. Eq. 2 is used to calculate the similarities between words, and then high PMI scores are aggregated to obtain the final SOC-PMI score [7].

2.2 A General Framework

To measure the overall similarity between two sentences, a general framework is presented by incorporating the main similarity features. As far as we know, [7] is the most comprehensive approach which incorporates several representative similarity metrics (i.e., string similarity and semantic similarity[2]). The string similarity is further composed of three different measurement, i.e., NLCS, NMCLCS_1 and NMCLCSn. Fig. 1 illustrates the general framework for searching the top-k similar sentences. Due to lack of large labeled data, the existing state-of-the-art assembling techniques commonly set the weight values arbitrarily

[2] The common word order similarity is neglected here because [7] has demonstrated that it has no influence on the overall score.

[7,10,12,14]. We apply the same strategy (i.e., $weight_{string}$=$weight_{semantic}$=1/2, $weight_{NLCS}$=$weight_{NMCLCS_1}$=$weight_{NMCLCS_n}$=1/3). While obtaining optimal values of these weights is certainly an interesting issue, it is out of the scope of this paper. It is very time consuming to test every candidate especially when the data collection is large. Therefore, efficient strategies on searching top-k semantic similar sentences are necessary.

3 Proposed Approaches

We propose efficient strategies for extracting the top-k similar sentences. The key idea is that by building appropriate index in the preprocessing, we only need to test a small part of candidates in the whole data collection.

3.1 Optimization on String-Based Similarity

We employ NLCS, NMCLCS_1, NMCLCSn as our string-based similarity features [7].

• NLCS

The basic idea for improving the efficiency of similar word extraction, is to test as few candidates as possible. To address this issue, in the preprocessing we need to built an effective index which facilitates the candidate test process. A well known technique is n-gram (or q-gram [17]) model, which is utilized in our framework. Moreover, because our purpose is to search the top-k similar words, we can further improve the efficiency based on the property of the similarity.

From Eq. 1 we know that NLCS is based on two factors: (1) length of LCS; (2) length of the candidate w_i (length of the query is not important because we cannot know it in the preprocessing). The similarity increases when $length(LCS)$ increases or $length(w_i)$ decreases. Therefore we have the following lemma.

Lemma 1 (Lower Bound of Gram Length). *Let P be the top-k similar word to the query Q so far. Q and P have the NLCS score as τ_{top-k}. We denote the set R be the untested words and the gram set G be the untested grams. If $\forall g \in G$ and $\forall r \in R$, $|g_r| < |Q| \cdot \tau_{top-k}$, then the top-k similar words w.r.t. string similarity score have been found.*

Proof (Sketch of Proof). (Proof by Contradiction) Assume there is one candidate word r with gram $|g_r| < |Q| \cdot \tau_{top-k}$ in R and it is ranked in the top-k list. Then we have $Sim(Q, r) \geq \tau_{top-k}$, $\frac{|g_r|^2}{|Q||r|} \geq \tau_{top-k}$, $|g_r|^2 \geq |Q| \cdot |r| \cdot \tau_{top-k}$, $|g_r| \geq |Q| \cdot \tau_{top-k}$, which contradicts the assumption. □

This lemma tells us that we can sort the grams in descending order of their lengths in the index. Moreover, for each *gram*, we sort all the corresponding words (which include this gram) in ascending order of their lengths. To search the top-k similar words, we start from the word list which has the longest gram. In the list, we first test shorter words and then the longer ones.

• **NMCLCS$_1$**
This similarity measures the maximal common consecutive prefix substrings of two strings. Similar to NLCS, from Eq. 1, we can see that NMCLCS$_1$ is determined by two factors: length of NMCLCS$_1$ and length of candidate word. NMCLCS$_1$ increases when the length of NMCLCS$_1$ increases or the length of the candidate decreases. Therefore, we can build a gram index that all the grams are in descending order of their lengths. For each *gram*, the related words are in ascending order of their lengths. The difference for indices between NLCS and NMCLCS$_1$ is that the grams for the latter are only a part of the former (i.e., start at the beginning of each string). The lower bound used to terminate is $|Q| \cdot \tau_{top-k}$. The computation of τ_{top-k} is based on Eq. 1. The algorithm of searching top-k similar words for NMCLCS$_1$ is very similar to that for NLCS. We need to modify the index strategy in the algorithm.

• **NMCLCSn**
NMCLCSn measures the maximal common consecutive substring which can starts at any position n. The index for NMCLCSn is similar to that for NMCLCS$_1$. The only difference is the strategy for decomposing words into grams. NMCLCSn parses words into grams which can start at any positions. The lower bound used to terminate is $|Q| \cdot \tau_{top-k}$, where τ_{top-k} is calculated by Eq. 1. The algorithm for NMCLCSn is similar to that for the above mentioned string similarities, with modification according to the definition of NMCLCSn.

3.2 Optimization on Corpus-Based Similarity

We apply SOC-PMI [7] as the corpus-based similarity evaluator. SOC-PMI linearly aggregates the top large PMI values (i.e., Eq. 2) of each pair words with their neighbors. We introduce an efficient technique [19].

Lemma 2 (Upper Bound of Word Frequency). *Let Q be the query word and P be the top-k similar word so far. Q and P have the similarity score τ_{top-k}. If $\forall r \in R$, $f(r) > \frac{m}{2^{top-k}}$, the top-k similar words have been found. Here R is the remaining untested words and m is the size of the corpus.*

Based on this lemma, we sort all the candidates in ascending order of their frequencies in the preprocessing and measure the similarity while querying one by one. When we find the current frequency of candidate word is large than $\frac{m}{2^{top-k}}$, we can terminate the process and avoid to test the remaining candidates.

3.3 Sentence Similarity Computation

We have introduced how to measure the similarity between words. Here we take illustrate how to obtain the similarity between sentences. Let P ="*Cheapest online hosting in Japan.*", Q="*Domain is free to use for the first year*". After removing all stop words and lemmatizing, we obtain P={*cheap online hosting Japan*} and Q={*domain free use year*}. Through similarity measuring on each combination of word pair for the sentences, we construct a similarity matrix

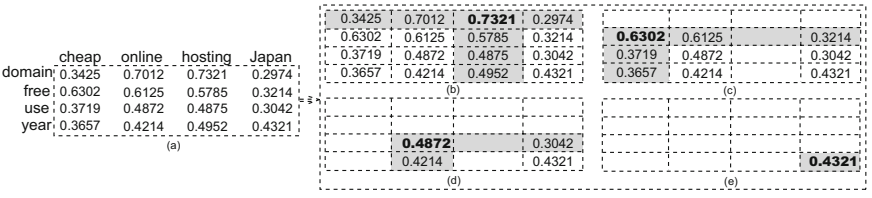


Fig. 2. Similarity measurement between two sentences

which is illustrated in Fig. 2 (a). Each element represents the overall similarity score of the word pair, i.e., aggregation of string-based and corpus-based similarities. To obtain the score between sentences, we first find the maximal-valued element (i.e., representative word). Then the related row and column which includes this element are removed. This process is recursively executed and finally all the similarities of the representative words have been extracted (i.e., Fig. 2 (b-e)). The similarity between sentences is computed as follows: $S(P,Q) = \frac{\Sigma_{i=1}^{|min(|P|,|Q|)|}\rho_i(|P|+|Q|)}{2|P|*|Q|}$, where ρ_i is the value of the representative word of round i. Therefore we have $S(P,Q)=\frac{(0.7321+0.6302+0.4872+0.4321)(4+4)}{2*4*4}=0.5704$.

3.4 Assembling Similarity Features

We introduce an efficient assembling approach to hasten the process of searching top-k similar sentences [4]. To illustrate the method, we use the concrete example (i.e., Fig. 1) to explain, as shown in Fig. 3. In the first iteration, we obtain the top-1 sentences with their scores in the features, i.e. S7: 0.4358 and S3: 0.8012. Next the threshold is computed, i.e., 0.6185 ($w_A \cdot Sim_{A_i} + w_B \cdot Sim_{B_i}$, where w is the weight value[3]). Because the overall scores of the two accessed sentences are smaller than the threshold, no result is output in this round. In the second iteration (i.e., testing on top-2 sentences in both lists), the threshold is computed as 0.6024. Because the score of S3 is 0.6111 which is larger than the threshold, S3 can be output immediately. We can see that for this example, the performance of obtaining the top-1 result is very efficient because we do not need to evaluate many candidates. The remaining processes are executed in a similar way.

In addition to the above mentioned feature aggregation, we apply the threshold-based strategy in assembling different string similarities (i.e., NLCS, NMCLCS$_1$, and NMCLCS$_n$), and assembling words into sentence to obtain the top elements.

4 Experimental Evaluation

We conducted experiments using 16-core Intel(R) Xeon(R) E5530 server which runs Debian 2.6.26-2. All the algorithms were written in C and compiled by GNU gcc. The baseline is implemented based on the state-of-the-art [7].

[3] The weights are set to 1/2, respectively, as explained in Section 2.

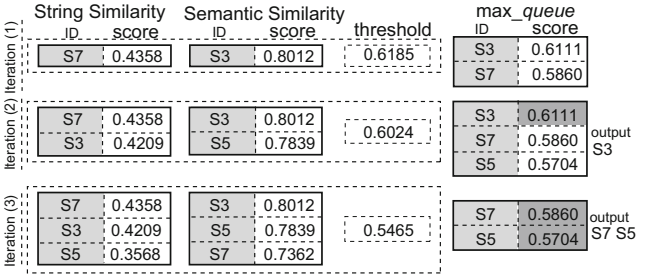
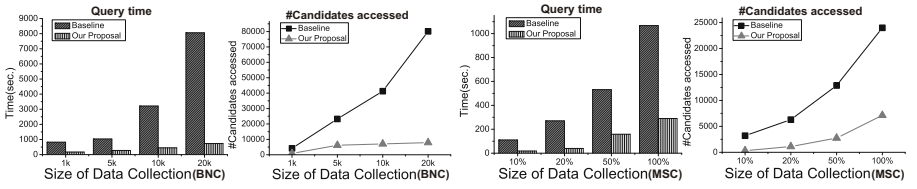
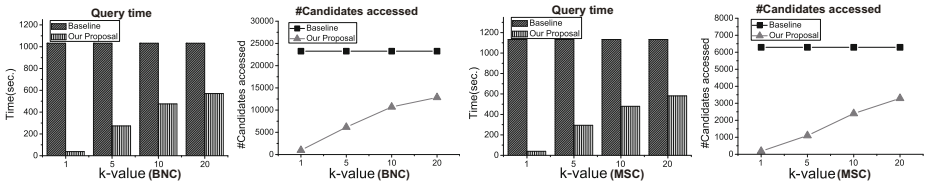


Fig. 3. Efficient searching top-k semantic sentences framework



(a) BNC dataset (b) MSC dataset

Fig. 4. Effect on size of data collection



(a) BNC dataset (b) MSC dataset

Fig. 5. Effect of k-value

Effect of Size of Data Collection. The evaluated datasets come from BNC[4] and MSC[5]. We randomly extracted 1k, 5k, 10k, 20k sentences from BNC and divided MSC into different size, i.e.,10%, 20%, 50%, 100%, as our datasets. Fig. 4 shows the top-5 results under 10 randomly selected queries. We can see that our proposal is much faster than the baseline for both datasets because the proposal largely reduces the number of candidates tested. When the size of data collection increases, the query time of our proposal increases linearly and it scales well.

Effect of k. We randomly chose 10 queries from the collections. The size of BNC was fixed to 5k and the whole MSC was used. From Fig. 5 we see the baseline needs to access all candidates and the query time is the same for all situations. For our proposal, the top-1 can be returned almost instantly. When k increases, the query time increases because more candidates need to be evaluated.

[4] `http://www.natcorp.ox.ac.uk/`

[5] Microsoft Research Paraphrase Corpus. It contains 5800 pairs of sentences.

5 Conclusion

In this paper, we proposed to tackle the efficiency issue of searching top-k similar sentences which has not been studied before. Several efficient strategies are introduced to test as few candidates as possible in the process. The comprehensive experiments demonstrates the efficiency of the proposed techniques.

References

1. Burgess, C., Livesay, K., Lund, K.: Explorations in context space: words, sentences, discourse. Discourse Processes (1998)
2. Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., Silvestri, F.: Caching query-biased snippets for efficient retrieval. In: EDBT (2011)
3. Cui, H., Sun, R., Li, K., Kan, M.-Y., Chua, T.-S.: Question answering passage retrieval using dependency relations. In: SIGIR (2005)
4. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: PODS (2001)
5. Hellerstein, J.M., Avnur, R., Chou, A., Hidber, C., Olston, C., Raman, V., Roth, T., Haas, P.J.: Interactive data analysis: The control project. Computer 32 (1999)
6. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Commun. ACM (1975)
7. Islam, A., Inkpen, D.: Semantic text similarity using corpus-based word similarity and string similarity. ACM Transactions on Knowledge Discovery from Data (2008)
8. Landauer, T., Dumais, S.: A solution to plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychological Review (1997)
9. Landauer, T.K., Foltz, P.W., Laham, D.: An introduction to latent semantic analysis. Discourse Processes (1998)
10. Li, Y., McLean, D., Bandar, Z.A., O'Shea, J.D., Crockett, K.: Sentence similarity based on semantic nets and corpus statistics. IEEE Transaction on Knowledge and Data Engineering (2006)
11. Metzler, D., Dumais, S.T., Meek, C.: Similarity Measures for Short Segments of Text. In: Amati, G., Carpineto, C., Romano, G. (eds.) ECiR 2007. LNCS, vol. 4425, pp. 16–27. Springer, Heidelberg (2007)
12. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: AAAI (2006)
13. Radlinski, F., Broder, A., Ciccolo, P., Gabrilovich, E., Josifovski, V., Riedel, L.: Optimizing relevance and revenue in ad search: a query substitution approach. In: SIGIR (2008)
14. Sahami, M., Heilman, T.D.: A web-based kernel function for measuring the similarity of short text snippets. In: WWW (2006)
15. Tsatsaronis, G., Varlamis, I., Vazirgiannis, M.: Text relatedness based on a word thesaurus. Journal of Artificial Intelligence Research (2010)
16. Turney, P.D.: Mining the web for synonyms: Pmi-ir versus lsa on toefl. In: EMCL (2001)
17. Ukkonen, E.: Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science 92(1) (1992)
18. Wei, F., Li, W., Lu, Q., He, Y.: Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In: SIGIR (2008)
19. Yang, Z., Kitsuregawa, M.: Efficient searching top-k semantic similar words. In: IJCAI (2011)