Performance Evaluation of Similar Sentences Extraction

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

Institute of Industrial Science, University of Tokyo, Japan
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan
{guyanhui,yangzl,miyuki,kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract. Similar sentence extraction is an important issue because it is the basis of many applications. In this paper, we conduct comprehensive experiments on evaluating the performance of similar sentence extraction in a general framework. The effectiveness and the efficiency issues are explored on three real datasets, with different factors considered, i.e., size of data, top-k value. Moreover, the WordNet is taken into account as an additional semantic resource and incorporated into the framework. We thoroughly explore the performance of the updated framework to study the similar sentence extraction.

1 Introduction

It is well known that extracting similar sentences is an important problem because it can be applied in a number of applications, such as snippet extraction, image retrieval, question-answer model, document retrieval, and so forth [11,15]. From a given sentence collection, this kind of queries asks for those sentences which are most semantically similar to a given one.

In this paper, we aim to study the issue of similar sentence extraction by comprehensively evaluating the performance in a general framework. There are several components considered, i.e., syntax-based similarity [9,5]; semantic-based similarity [13,10,5,14]; common order (structure) similarity [9,5]; and hybrid similarity [5,9,14]. Due to the comprehensive property of the framework in [5], it is considered as the baseline of this paper. We extended the baseline by deliberately designing several novel strategies to improve the efficiency [2].

The efficiency issue is modeled as the top-k similar sentence extraction. To deal with this problem, the baseline (and other previous works) naively tests every candidate sentence, which is very time consuming, especially when the size of the sentence collection is huge. To tackle this issue, we introduced efficient strategies to evaluate as few candidates as possible. Specifically, for each similarity measurement, we introduced 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 [2]. In the first part of this paper (i.e., Section 2 and Section 3), we focus on thoroughly evaluating the trade-off of the effectiveness and efficiency of the two works (i.e., [5] and [2]).

From the experimental evaluation, we can see that although the efficiency can be improved, the effectiveness (i.e., precision) may be not satisfied for users. To address this issue, in the second part of the paper (i.e., Section 4), we incorporate an additional semantic resource, i.e., WordNet, into the general framework. The trade-off between the effectiveness and efficiency of the updated framework is studied.

The rest of this paper is organized as follows. In Section 2, we introduce the general framework for similar sentence extraction, with optimization implementation. The comprehensive experimental evaluation is illustrated in Section 3. Incorporating WordNet as an additional semantic resource is introduced in Section 4 and the experimental evaluation is presented in this section. The related work is introduced in Section 5 and finally we conclude our paper in Section 6.

2 Efficient Top-*k* Similar Sentence Extraction

2.1 Preliminaries

To measure the similarity $sim(Q, P)$ between two sentences Q and P, we apply state-of-the art strategies by assembling multiple similarity metric features [5,9]. Given that we cannot evaluate all the similarity measurement strategies in this paper, we select several representative features based on the framework presented in [5]. Notably, considering that a sentence comprises a set of words, the similarity score between two sentences denotes the overall scores of all word pairs, the components of which belong to each sentence. See [5] for detail on computing sentence similarity based on word similarity.

2.2 Similarity Measurement Strategies

2.2.1 String Similarity

String similarity measures the difference in syntax between strings. An intuitive idea is that two strings are similar to each other if they have adequate common subsequences (e.g., LCS [4]). String similarity measurement strategies, including edit-distance, hamming distance and so on. We focus on three representative string similarity measurement strategies introduced in [5], namely, NLCS, NMCLCS$_1$ and NMCLCS$_n$[1].

2.2.2 Corpus-Based Similarity

The corpus-based similarity measurement strategy recognizes the degree of similarity between words using large corpora, e.g., BNC, Wikipedia, Web and so on. Corpus-based similarity measurement strategies are of several types: PMI-IR, LSA, HAL, and so on. In this paper, we apply the Second Order Co-occurrence

[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. See [5] for detail.

PMI (SOC-PMI) [5] which employs PMI-IR to consider important neighbor words in a context window of the two target words from a large corpus. They use PMI-IR to calculate the similarities between word pairs (including neighbor words). High PMI scores are then aggregated to obtain the final SOC-PMI score.

2.3 General Framework for Measuring Sentence Similarity

To measure the overall similarity between two sentences, a general framework is presented by incorporating all similarity measurement strategies. To the best of our knowledge, [5] presented the most comprehensive approach that incorporates representative similarity metrics. They construct a similarity matrix and recursively extract representative words (maximal-valued element) which are then aggregated to obtain the similarity between two sentence.

2.4 Optimization Strategies

We apply the framework of [2] as the evaluation base which is an optimization strategies on the framework which is proposed in [5]. The original are composed with the following: String (NLCS,NMCLCS$_1$ and NMCLCS$_n$), Semantic (corpus-based strategy) and Common word order [2]. Actually, they apply string and semantic strategies in the framework. Accordingly, [2] proposed efficient similar sentence matching strategy on string and semantic.

3 Experimental Evaluation

To evaluate effectiveness and efficiency, we apply the baseline algorithm which is implemented according to the state-of-the-art work [5]. In the whole experimental evaluation, we use three different datasets, i.e., the *benchmark* dataset which was used in [9,5], BNC[3] dataset and MSC[4] dataset. 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.

3.1 Evaluation on Efficiency

To evaluate the efficiency, we conduct the experiments on two real datasets, i.e., BNC and MCS. Fig. 1 shows the execution time based on different dataset size and different k value. We can see that our proposal is much faster than the baseline for both datasets because the proposal. We see the baseline needs to access all candidates and the query time is the same for all situations. Fig. 2 illustrates the number of candidates accessed. Our proposal largely reduces the number of

[2] We conducted experiments on *benchmark* dataset and found that common word order similarity has low importance in sentence similarity measurement.

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

[4] Microsoft Research Paraphrase Corpus. It contains 5801 pairs of sentences.

candidates tested. When the size of data collection increases, the query time of our proposal increases linearly and it scales well. For different k, baseline needs to access all the candidates while ours accesses a small parts of them.

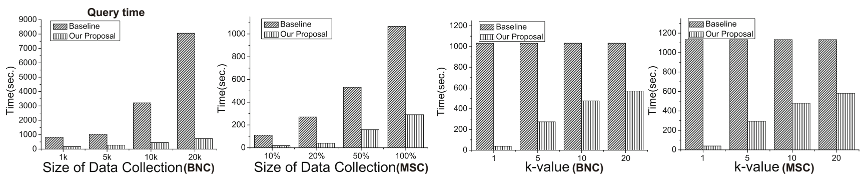


Fig. 1. Results on execution time

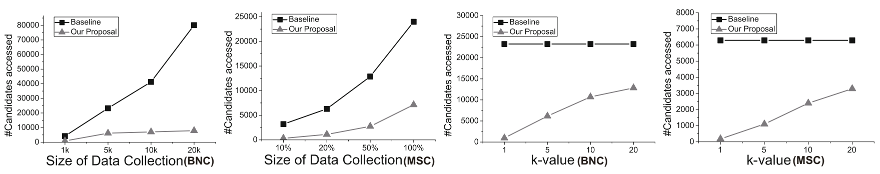


Fig. 2. Results on number of candidates accessed

3.2 Evaluation on Effectiveness

In the former experiments, we have demonstrated that our proposal outperforms the state-of-the-art technique with regard to the efficiency issue. In this section, we evaluate the effectiveness of our proposal. We conduct experiments a labeled datasets, i.e., the *benchmark* dataset. The experimental result conducted on the *benchmark* dataset is illustrated in Fig. 3. From this figure we can see that the results of both algorithms are close to each other, which indicates that our proposal can obtain the same high precision as the state-of-the-art technique.

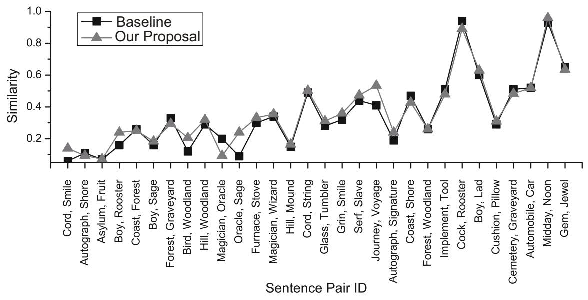
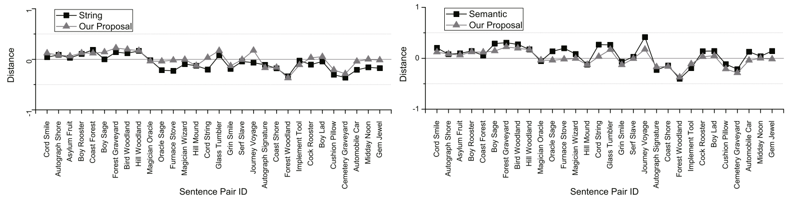


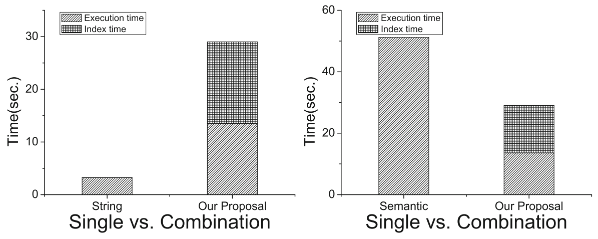
Fig. 3. Evaluation on effectiveness under benchmark dataset

3.3 Evaluation on Trade-Off between Efficiency and Effectiveness

Because the introduced framework is built based on the aggregation of different features, i.e., string similarity and semantic similarity, the execution time is related to the number of features used. Therefore, if we apply only one feature, the execution time is shorter yet such strategy may affect the whole effectiveness. So we conduct a set of experiments to study the trade-off between efficiency and effectiveness. In this set of experiments, we first evaluate the performance of single feature in the baseline strategy vs. our whole framework. Then we explore the performance of single feature in the baseline strategy vs. single feature in our framework as illustrated in Fig. 4 and Fig. 5.



(a) Precision comparison



(b) Execution time and index time comparison

Fig. 4. Effectiveness, efficiency and index cost evaluation between single strategy in baseline and combination strategy in our proposal

To evaluate the effectiveness, we apply single strategy in baseline and combination strategy[5] in our proposal. Firstly, we compare the effectiveness between each strategy in baseline and combination strategy in our proposal. We also evaluate the execution time and index time of each pair under such strategy. The experimental result is illustrated in Fig. 4. We report string strategy and semantic strategy results which are listed in Fig. 4. From the figure we can see that, single strategy beats our proposal in execution time while not in the effectiveness.

The former evaluation on effectiveness tells us that the combination strategy can obtain more precise results. Fig. 4(b) presents the experimental results of execution time of single strategy in baseline and execution time of our proposal. Since the strategies in baseline do not need to index, we report the index time of combination strategy in our proposal. Note that in all the evaluation, we show the performance of extracting the top-5 results with 10 randomly selected queries. Here we take string vs. combination strategy pair as an example. The execution time and the index time for string (i.e., the left bar) and our proposal (i.e., the right bar). We can easily see that the execution time of string is very fast while the combination strategy consumes more time. However, the execution time of single semantic strategy is longer than that of combination strategy. Such

[5] Combination strategy means the whole framework strategies.

result tells us that the optimization on semantic similarity is crucial among all the optimization strategies.

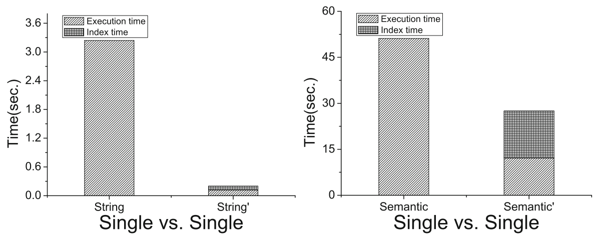


Fig. 5. Efficiency and index cost evaluation between single strategy in baseline and combination strategy in Our proposal

Evaluation on single strategy vs. combination strategy demonstrates the trade-off between effectiveness and efficiency. In this section, we study the performance of single feature in the baseline strategy vs. single feature in our framework (i.e., Fig. 5). Since we apply the same similarity strategy, they have the same precision. Therefore, we only compare the execution time and index time of each strategy. From the figure we can see that the execution time of each single strategy in baseline is longer than that of in our proposal (i.e., including the execution time and index time). These results demonstrate that the optimizations of our proposal are efficient and make effect on each feature.

4 Effectiveness Evaluation by Incorporating an Additional Semantic Resource

Hybrid approaches incorporate different similarity strategies, such as string similarity, knowledge-based similarity, corpus-based similarity, etc. In this section, we explore the effect of incorporate a knowledge-based similarity, i.e., WordNet similarity into the whole framework.

4.1 WordNet-Based Similarity Strategy

A word thesauri such as WordNet, constitutes the knowledge base for text-related research. An intuitive idea to determine whether two words are semantically similar to each other is by finding if the shortest path between them is small. This edge-counting approach has been extended by incorporating additional features in the knowledge base, such as depth, information content, or semantic density. We select one representative metric proposed in [7], that is, Leacock and Chodorow strategy. We take two words w_i,w_j, the similarity of which is determined as follows:

$$Sim_{lch}(w_i, w_j) = -ln\frac{length(w_i, w_j)}{2 * D}$$

where $length(w_i, w_j)$ is the length of the shortest path between two concepts (by applying node-counting strategy). D is the maximum depth of the taxonomy.

4.2 Optimization on WordNet

We apply the Leacock and Chodorow strategy as a WordNet evaluator which is an efficient technique [16].

Lemma 1 (Ordering in WordNet). *Let Q be the query. Let P and S be two candidates that exist in the same taxonomy of Q, that is, T_P and T_Q. The shortest path between Q and P (or S) is L_P in T_P (or L_S in T_S). The maximum depth of T_P is D_P (or D_S of T_S). P is more similar to Q compared with S. Thus, we have $\frac{D_P}{L_P} > \frac{D_S}{L_S}$.*

The lemma tells us that the similarity ordering between candidates in WordNet depends on the integration of the shortest path and the maximum depth of the taxonomy. For example, $father$ is in both a noun taxonomy (i.e., $\frac{D}{L} = 19$) and a verb taxonomy (i.e., $\frac{D}{L} = 14$)[6]. Thus, $father$ in a noun taxonomy should be accessed before that in a verb taxonomy. Sequentially we access the synonyms set between two taxonomies successively based on the value of $\frac{D}{L}$. Based on this lemma, we index all the candidates together with their neighbors and maximum taxonomy depth. We sequentially access nodes based on Lemma 1 and obtain the top-k results in a progressive manner.

4.3 Experimental Evaluation

In this section, we first evaluate the single strategy and then the different combination of similarity strategies. Besides $Sim_{Baseline}$ (baseline is the combination of string similarity and BNC-based semantic similarity), we incorporate a different strategy, i.e., $Sim_{WordNet}$ into the framework with equal weight. We apply *benchmark* dataset (Miller-Charles' dataset) which has also been used in [9] to evaluate the effectiveness in this and the following sections. Table 1 illustrates the results of the correlation coefficient with human ratings.

Table 1. Precision on different strategies

Strategy	WordNet	Baseline+WordNet	String+WordNet
Equal weight	0.60707	0.78901	0.73333
Cross validation	0.60707	0.82019	0.77815
Weight tuning	0.60707	0.83033	0.79378

The original framework of baseline applies the equal weight of each strategy, i.e., all the strategies have the same effect on the similarity score. However, this equal weight strategy cannot handle well because each strategy has its own property and has its own importance in the similarity measurement. Therefore, we apply cross validation strategy to tune the weight, which can obtain better results than equal weight strategy. However, there are some words which are

[6] The maximum depths of the two taxonomies are 19 for noun and 14 for verb by querying WordNet during preprocessing.

not included in it and these "missing words" may affect the similarity score. We dynamically tune the weight of these missing words which can reduce the affect of similarity measurement. The whole results shows in Table. 1.

5 Related Work

Measuring similarity between long texts has been extensively studied [3,6]. However, only a few of them can be directly applied to sentence similarity measurement [5,9,10]. Based on the different strategies applied, existing works on similarity measurement between sentences can be classified into several categories:

Syntax Based Strategy. Numerous strategies estimate the string similarity between two texts [8]. One representative q-gram based strategy calculates the edit distance between words. In [17] the authors proposed several strategies, including adaptive q-gram selection, for the efficient retrieval of the top-k results. In [12], the authors introduced deliberated techniques, e.g., divide-skip-merge, to extract similar strings. Common word order [5] evaluates the similarity of word position difference.

Semantic Based Strategy. Knowledge based and corpus based are two kinds of semantic based strategies. In [13], they firstly create semantic networks from word thesauri and then measure the relatedness between words based on these semantic networks. The hierarchy property of WordNet has been explored in [9]. Some well known methods in corpus-based similarity are LSA (Latent Semantic Analysis) and HAL (Hyperspace Analogues to Language), etc. One representative strategy ESA (Explicit Semantic Analysis) [1] which applies machine learning techniques to explicitly represent the meaning of any text as a weighted vector of Wiki-based concepts.

Hybrid Strategy. To tackle the drawback of single strategy, the hybrid strategy was proposed [9,5]. The combination of knowledge based strategy and word order based strategy was proposed in [9]. In [5], the author applies string based, common word order based, and corpus based strategies to measure the similarity between sentences.

Currently, several works [16,2] explore efficiency issue to optimize state-of-the-art similarity strategy. Efficient extraction on semantic similar words is presented in [16] by optimizing string-based, WordNet-based and corpus-based similarity strategies. In [2], the authors address efficiency issue to efficiently search for semantic similar sentences on three string similarity strategies and corpus-based strategy.

6 Conclusion and Future Work

In this paper, we study the performance on top-k similar sentence extraction. Extensive experiments have been explored on three real datasets with different factors considered. The trade-off between efficiency and effectiveness has also been introduced. Moreover, the WordNet is taken into account as an additional semantic resource and incorporated into the framework. We thoroughly explore the performance of the updated framework to study the similar sentence extraction.

References

1. Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the International Joint Conference on Artifical Intelligence, IJCAI 2007, pp. 1606–1611 (2007)
2. Gu, Y., Yang, Z., Nakano, M., Kitsuregawa, M.: Towards Efficient Similar Sentences Extraction. In: Yin, H., Costa, J.A.F., Barreto, G. (eds.) IDEAL 2012. LNCS, vol. 7435, pp. 270–277. Springer, Heidelberg (2012)
3. Hatzivassiloglou, V., Klavans, J.L., Eskin, E.: Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning. In: Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP/VLC 1999, pp. 203–212 (1999)
4. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Communications of ACM 18(6), 341–343 (1975)
5. Islam, A., Inkpen, D.: Semantic text similarity using corpus-based word similarity and string similarity. ACM Transactions on Knowledge Discovery from Data 2(2), 1–25 (2008)
6. Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. Psychological Review 104, 211–240 (1997)
7. Leacock, C., Chodorow, M.: Combining local context and wordnet similarity for word sense identification. In: Fellbaum, C. (ed.) WordNet: An Electronic Lexical Database, pp. 305–332. MIT Press (1998)
8. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10(8), 707–710 (1966)
9. Li, Y., McLean, D., Bandar, Z., O'Shea, J., Crockett, K.A.: Sentence similarity based on semantic nets and corpus statistics. IEEE Transactions on Knowledge and Data Engineering 18(8), 1138–1150 (2006)
10. Mihalcea, R., Corley, C., Strapparava, C.: Corpus-based and knowledge-based measures of text semantic similarity. In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2006, pp. 775–780 (2006)
11. Mihalcea, R., Tarau, P.: Textrank: Bringing order into text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2004, pp. 404–411 (2004)
12. Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2004, pp. 743–754 (2004)
13. Tsatsaronis, G., Varlamis, I., Vazirgiannis, M.: Text relatedness based on a word thesaurus. Journal of Artificial Intelligence Research 37, 1–39 (2010)
14. Turney, P.D.: Mining the Web for Synonyms: PMI-IR versus LSA on TOEFL. In: Flach, P.A., De Raedt, L. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 491–502. Springer, Heidelberg (2001)
15. Wang, K., Ming, Z.Y., Hu, X., Chua, T.S.: Segmentation of multi-sentence questions: towards effective question retrieval in cqa services. In: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, pp. 387–394 (2010)
16. Yang, Z., Kitsuregawa, M.: Efficient searching top-k semantic similar words. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 2011, pp. 2373–2378 (2011)
17. Yang, Z., Yu, J., Kitsuregawa, M.: Fast algorithms for top-k approximate string matching. In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2010, pp. 1467–1473 (2010)