Exploration on Effectiveness and Efficiency of Similar Sentence Matching

Yanhui Gu, Zhenglu Yang, Miyuki Nakano, Masaru Kitsuregawa
Institute of Industrial Science, University of Tokyo
Komaba 4-6-1, Meguro, Tokyo, Japan 153-8505
Email: {guyanhui,yangzl,miyuki,kitsure}@tkl.iis.u-tokyo.ac.jp

***Abstract*—Similar sentence matching is an essential issue for many applications, such as text summarization, image extraction, social media retrieval, question-answer model, and so on. A number of studies have investigated this issue in recent years. Most of such techniques focus on effectiveness issues but only a few focus on efficiency issues. In this paper, we address both effectiveness and efficiency in the sentence similarity matching. For a given sentence collection, we determine how to effectively and efficiently identify the top-k semantically similar sentences to a query. To achieve this goal, we first study several representative sentence similarity measurement strategies, based on which we deliberately choose the optimal ones through cross validation and dynamically weight tuning. The experimental evaluation demonstrates the effectiveness of our strategy. Moreover, from the efficiency aspect, we introduce several optimization techniques to improve the performance of the similarity computation. The trade-off between the effectiveness and efficiency is further explored by conducting extensive experiments.**

I. Introduction

Similar sentence matching is an essential issue because it is the basis of many applications, such as as text summarization, image extraction, social media retrieval, question-answer model, and so on.

Traditional techniques for measuring the similarity between documents (long texts), e.g., TF-IDF, have been introduced based on an intuitive assumption that a large number of common words exit in similar documents. However, these methods are inappropriate for measuring similarities between sentences because in short texts common words are few or even nonexistent [14], [15], [19]. To address this issue, numerous strategies have been proposed to measure the similarity between sentences. These strategies can be classified into four categories: (1) knowledge-based [15], [19]; (2) string similarity based [2], [16]; (3) corpus-based [10], [11]; and (4) hybrid strategies [10], [14].

As far as we know, the most comprehensive framework for sentence similarity calculation is introduced in [10]. The authors integrate several representative string-based and corpus-based (i.e., BNC) similarities. It is well known that WordNet and Wiki are important semantic resources and have been extensively studied on the measurement of semantic similarities [4], [14], [15]. An intuitive idea is to incorporate these semantic resources (i.e., WordNet and Wiki) into the general framework (i.e., [10]) to improve the effectiveness. In the first part of this paper, we thoroughly explore the idea, that evaluates the effect of different measurements on calculating sentence similarities. We believe that this is the first work which comprehensively studies the sentence similarity measurement by using most semantic resources.

In addition to the effectiveness aspect, efficiently searching similar sentences from a large number of data has become an important issue [6], [17] in the literature. From a given sentence collection, such queries aim to identify sentences that are most semantically similar to a given one. A naive approach can employ the following procedure: we first measure the semantic similarity score between the query and each sentence in the data collection using state-of-the-art techniques. The sentences are then sorted based on the score. Finally, the top-k sentences are identified and returned to the user. However, as the data collection size increases, the scale of the problem likewise increases, thus rendering state-of-the-art techniques impractical [5], [17], which highlights the importance of the efficiency issue. Several works explored optimization strategies for similarity measurement. In [21], the author addressed the efficiency issue by optimizing the string similarity, WordNet similarity and semantic similarity of words. An efficient method for the extraction of similar sentences was proposed in [6], where different strategies were combined by applying the threshold algorithm. In this paper, taking into account the new similarities (i.e., WordNet and Wiki), we introduce the corresponding optimization strategies to improve the efficiency. The trade-off between effectiveness and efficiency is also studied in this paper.

The contributions of this paper are as follows:

- We introduce several representative similarity measurement strategies and evaluate the effectiveness of each strategy individually as well as that of different combinations.
- We propose a dynamic weight tuning strategy to improve the effectiveness of the similarity measure. In addition, we also study the weight setting of the combination of different similarity strategies.
- We introduce optimization strategies for the new semantic resources (i.e., WordNet and Wiki) to improve the efficiency of sentence similarity matching.
- We conduct comprehensive experiments to evaluate the performance of the proposed strategies. The results show that the proposed strategies outperform the state-of-the-art method. We also illustrate the trade-off between effectiveness and efficiency.

II. 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 [10], [14]. Given that we cannot evaluate all the similarity measurement strategies in this paper, we select several representative features based on the framework presented in [10]. 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 [10] for detail on computing sentence similarity based on word similarity.

A. Similarity Measurement Strategies

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 [8]). String similarity measurement strategies, including edit-distance, hamming distance and so on. We focus on three representative string similarity measurement strategies introduced in [10], namely, NLCS, NMCLCS_1 and NMCLCS_n[1].

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 PMI (SOC-PMI) [9], [10] 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.

3) Common Word Order Similarity:
Common word order similarity measures how similar the order of the common-words is between two sentences, as either the same order, almost the same order, or very different order. Although [20] indicates that syntactic information is less important during the semantic processing of sentences, we incorporate this similarity measurement strategy to test how much order similarity affects the whole sentence similarity. See [10], [14] for detail.

B. 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, [10] 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.

[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 [10] for detail.

III. Effectiveness Improvement with Additional Semantic Resources

Hybrid approaches incorporate different similarity strategies, such as string similarity, knowledge-based similarity, corpus-based similarity, etc. It is well known that WordNet and Wiki are two representative semantic resources and have been extensively studied on the measurement of semantic similarities [4], [14], [15]. Based on the general framework which is introduced in [10], in this paper we propose to take into account the additional important semantic resources (i.e., Wordnet and Wiki), to improve the effectiveness of the sentence similarity measurement. We explore the effect of different similarity metrics by using equal-weight setting (Section III), cross validation (Section IV), and dynamic weight tuning (Section V). Efficiency optimization on sentence similarity matching is introduced in Section VI.

A. Two Additional Semantic Resources

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 [12], 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.

2) Wiki-based Similarity Strategy: Unlike taxonomy-based methods, such as the WordNet-based strategy, Wiki-based similarity cannot employ a new entity. An representative strategy, ESA [4], which applies the Wiki encyclopedia as a knowledge base to map text into Wiki-based concepts. In this approach, each Wiki concept is represented as an attribute vector of words that occur in the corresponding article. Entries of these vectors are assigned weights by using the TF-IDF scheme which quantifies the strength of association between words and Wiki concepts. ESA measures similarity between sentences (arbitrary length) by aggregating each word distribution on concepts, i.e., sentence is a vector based on concepts with weight of each concept c_i calculated as: $\sum_{w_i \in T} v_i \cdot k_j$, where v_i is TF-IDF weight of w_i and k_j quantifies the strength of association of word w_i with Wiki concept c_j.

B. 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 two different strategies, such as $Sim_{WordNet}$ and Sim_{Wiki} into the framework with equal weight. We apply $benchmark$ dataset (Miller-Charles' dataset) which has also been used in [14] to evaluate the effectiveness in this and the following

sections. Figure 1 illustrates the results of the correlation coefficient with human ratings.

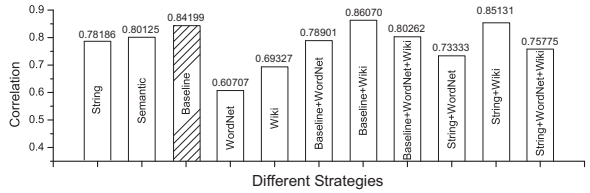


Fig. 1. Results of different strategies combination with equal weight

C. Result Explanation

The figure shows that all single similarity strategies are worse than the baseline strategy. Wiki is a good semantic resource because the combination of Wiki achieve better results than others but WordNet is not good on this dataset. However, the combination strategies are better than single similarity strategies but still falls short of the baseline strategy. Given their equal weights, all similarity strategies have the same proportion of similarity aggregation, that is, the weight is static and set arbitrarily.

IV. Effectiveness Improvement with Cross Validation

Considering that the weight in Section III is set to be equal, in this section, we test each possible weight of the combination strategy. Obtaining the optimal values of these weights is certainly an important issue. We argue that this work is orthogonal to the existing effectiveness oriented studies in a complementary manner. We try to achieve the best effectiveness by testing each possible weight.

A. Cross Validation

Cross validation strategy can test all the possible weight combination results with human ratings. In this paper, we apply a 10-fold cross validation strategy and study what weight can achieve the best effectiveness.

B. Experimental Evaluation

We conduct experiments on the benchmark dataset. Table I shows that, by applying cross validation, we can obtain better results from combination strategies. From Table I, we observe that the WordNet strategy still has extremely low effectiveness. In addition, combination strategies that contain the WordNet strategy are also outperformed by other strategies.

TABLE I. Correlation coefficient on combination with cross validation

Strategy	Correlation	Weight			
		String	Semantic	WordNet	Wiki
Baseline+WordNet	0.82019	0.377	0.531	0.092	-
Baseline+Wiki	0.86073	0.470	0.210	-	0.320
Baseline+WordNet+Wiki	0.81002	0.406	0.301	0.072	0.221
String+WordNet	0.77815	0.699	-	0.301	-
String+Wiki	0.86132	0.579	-	-	0.421
String+WordNet+Wiki	0.80126	0.470	-	0.110	0.420

Note: Baseline strategy involving string and semantic strategy.

C. Result Explanation

The experiment results show that setting weight arbitrarily and equally is improper for similarity measurement, especially in combination strategies. We can achieve better results by using cross validation strategy. However, from the experiment results of Section III and Section IV, we can see that, the results of combination of WordNet are still low.

V. Effectiveness Improvement with Dynamic Weight Tuning

From Section IV we can see that, WordNet is not a good semantic resource when measuring the semantic similarity under the benchmark dataset. Because of the omission of two word pairs in WordNet, similarity score of these words are "0" which affect the whole similarities. In this section, we reduce the weight of these words which are not included in WordNet by dynamically weight tuning.

A. Dynamic Weight Tuning

To address this issue, one possible solution is to remove these words when calculating the similarity score. However, such a strategy may affect the similarity score of other strategies because of the reduction in the number of words. Another solution is by dynamically tuning the combination weight. We take the $String + WordNet$ strategy as an example. If the similarity score of some word pairs are "0", but this value holds a rather large weight, which can reduce the final similarity score. We propose a dynamic combination weight tuning strategy to address this issue. We denote two sentences Q and P, which have m and n words, respectively. A total of γ words are not included in WordNet of sentences P and Q. Based on the strategy in [10], at least $\gamma * min(m,n)$ or at most $\gamma * max(m,n)$ word pairs are "0" (we apply average value $\frac{(m+n)}{2} \cdot \gamma$). Therefore, we mitigate the effect of WordNet by tuning the weight to $\frac{1}{k} \cdot (1 - \frac{m+n}{2mn} \cdot \gamma)$, where k is the number of combination strategies. So, $Sim_{String+WordNet}$ = $\frac{1}{k} \cdot \frac{2mn(k-1)+(m+n)\cdot\gamma}{2mn} \cdot Sim_{String}$+ $\frac{1}{k} \cdot \frac{2mn-(m+n)\cdot\gamma}{2mn} \cdot \gamma \cdot Sim_{WordNet}$, where k = 2 in this case.

B. Experimental Evaluation

We apply this strategy and conduct experiments on the benchmark dataset. Table II shows the results by dynamically tuning the weight.

TABLE II. Correlation coefficient on dynamically weight tuning

Strategy	Correlation	Weight			
		String	Semantic	WordNet	Wiki
Baseline+WordNet	0.83033	0.408	0.375	0.217	-
Baseline+Wiki	0.86073	0.470	0.210	-	0.320
Baseline+WordNet+Wiki	0.84752	0.334	0.314	0.157	0.195
String+WordNet	0.79378	0.649	-	0.351	-
String+Wiki	0.86132	0.579	-	-	0.421
String+WordNet+Wiki	0.86201	0.420	-	0.210	0.370

Table II shows that, we obtain better correlation coefficient by dynamically tuning the weight. For $String + WordNet + Wiki$, we obtain a better result than $Baseline + Wiki$ and $String + Wiki$.

C. Result Explanation

From the experiment results, we can see that dynamically tuning the weight of each similarity strategy is a possible solution to improve the effectiveness. However, all the weight tuning is conducted on the benchmark dataset. The weight may be changed if we apply the strategy to another dataset. Supervised learning techniques could solve such an issue but are out of the scope of this paper.

Common word order is an important strategy in similarity measurement. To evaluate the effect of common word order similarity, we incorporate such similarity strategy into the framework. We incorporate common word order similarity into baseline strategy. Experiments conducted on the benchmark dataset illustrated that only 19 of 30 pairs have common words. Of all these 19 pairs, 15 pairs have exactly the same order (including 10 pairs which have only "1" common word). Such similarity only affects 4 pairs. We set weight of common word order similarity from 0 to 0.5 with the granularity "0.01" and we obtain the best correlation coefficient "0.81972" at 0.01 which is less than baseline.

VI. Efficiency Improvement

Searching for similar sentences from a large amount of data has become an important issue [1], [6] in the literature. From a given sentence collection, such queries aim to identify sentences that are most semantically similar to a given one. A naive approach could be: we first measure the similarity score between the query and each sentence in the data collection using state-of-the-art techniques [10], [14], [15]. The sentences are then sorted based on a score. Finally, the top-k statements are identified and returned to the user. However, as the size of the collected data, testing each candidate sentence becomes time consuming. In [6], the author proposed a solution which optimized state-of-the-art techniques to retrieve top-k sentences. Techniques that optimize string similarity and semantic similarity is proposed. From the analysis in Section V, we select a best similarity combination strategy that has the best effectiveness. We select one representative strategy which achieves best performance, that is, $string$, $WordNet$ and $Wiki$, with the weight "0.420", "0.210" and "0.370", respectively.

A. Optimization on WordNet

We apply the Leacock and Chodorow strategy as a WordNet evaluator which is an efficient technique [21].

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., $\frac{D}{L} = 14$)[2]. 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.

B. Optimization on Wiki

ESA measures the similarity between sentences (arbitrary length) by aggregating each word distribution on concepts, that is, a sentence is a vector based on concepts with the weight of each concept c_i calculated as: $\sum_{w_i \in T} v_i \cdot k_j$, where v_i is TF-IDF weight of w_i and k_j quantifies the strength of association of word w_i with Wiki concept c_j. The traditional approach has to test each candidate in the data collection. In our optimized strategy, we first calculate all the similarity scores between each word in Wiki and between sentences in the data collection to obtain a set of lists during preprocessing which is illustrated in Figure 2. Then we build a weighted inverted list, here each list indicates a word with sorted corresponding sentences based on the similarity score. Given a query sentence Q, each word in Q corresponds a list of sentences. Therefore, we apply the threshold algorithm [3] with TF-IDF weight to retrieve the top-k sentences. This manner accesses a small number of components of the data collection without need to test every candidate sentence.

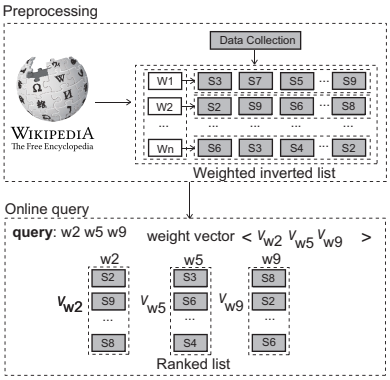


Fig. 2. Optimization on Wiki based strategy

C. Assembling Similarity Features

We introduce an efficient assembling approach to accelerate the process of searching for top-k similar sentences [3]. In [6], the author illustrated the method by using a concrete example. In this paper, we apply three different similarity measurement strategies, $String$, $WordNet$ and $Wiki$. We apply the threshold based strategy in assembling different similarities as well as in assembling words into a sentence to obtain the top elements. Given the page limitation, we do not include detailed explanations here.

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

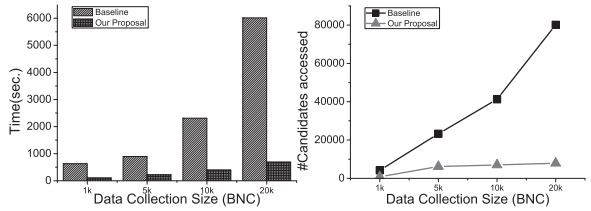
D. Experimental Evaluation on Efficiency

To evaluate the efficiency, we conduct extensive experiments on two large real datasets: BNC dataset (extracted from British National Corpus); MSC dataset (extracted from Microsoft Research Paraphrase Corpus).[3] Table III shows the statistics of these two datasets (Statistics after preprocessing is italicized.).

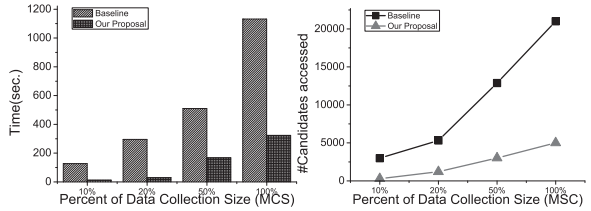
TABLE III. DATASET STATISTICS

	BNC		MSC	
Avg. sentence length	11.72	*7.19*	15.23	*9.31*
Min. sentence length	3	*2*	5	*3*
Max. sentence length	107	*38*	29	*17*
Max word length	17	*17*	13	*13*

1) Evaluation on Effect of Data Collection Size: Figure 3 shows the top-5 results after 10 randomly selected queries. We can see that our proposed optimized strategy is significantly faster than the baseline strategy for both datasets because our strategy substantially reduces the number of candidates tested. As the size of collected data increases, the query time of our proposed strategy also increases linearly and proportionately well.



(a) BNC dataset



(b) MSC dataset

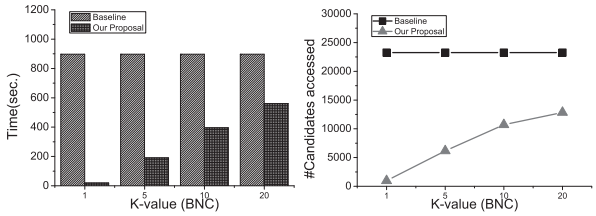
Fig. 3. Effect of data collection size

2) Evaluation on Effect of k value: In addition, we also verify the effect of k value. We randomly chose 10 queries from both datasets and fix the data size to be 5k for BNC and the whole size for MSC. Figure 4 shows that the baseline has to access all candidate sentences, such that, the query time is the same for all the situations. For our proposed method, the top-1 can be returned almost instantly. The query time increases when k increases because more candidates need to be accessed.

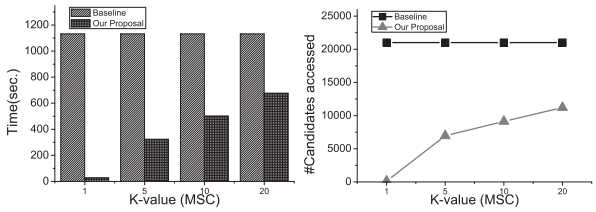
VII. TRADE-OFF BETWEEN EFFECTIVENESS AND EFFICIENCY

In Section VI-D, we prove that, our proposed optimization strategy can significantly reduce the execution time when retrieving top-k values. However, effectiveness and efficiency require a trade-off. We conducted experiments on the benchmark dataset by using $Baseline$ and $Baseline+WordNet+Wiki$ strategies to retrieve top-5 results. Table IV tells us combining several strategies can achieve high precision but may be time consuming. Therefore, designing an effective similar sentence matching framework with high efficiency remains a challenge.

[3]1k, 5k, 10k, 20k sentences are extracted from BNC and 10%, 20%, 50%, 100% of MSC are divided. After removing the duplicated sentences in MSC, 11,212 sentences are remained.



(a) BNC dataset



(b) MSC dataset

Fig. 4. Effect of k-value

TABLE IV. TRADE-OFF BETWEEN EFFECTIVENESS AND EFFICIENCY

Strategy	Effectiveness	Efficiency
	Correlation	Execution Time(Sec.)
Baseline(String+Semantic[BNC])	0.84019	2.90
Baseline+WordNet+Wiki	0.86201	3.32

VIII. RELATED WORK

Measuring similarity between long texts has been extensively studied [7], [11]. However, only a few of them can be directly applied to sentence similarity measurement [10], [14], [15]. Based on the different strategies applied, existing works on similarity measurement between sentences can be classified into several categories:

String similarity based strategy. Numerous strategies estimate the string similarity between two texts [13]. One representative q-gram based strategy calculates the edit distance between words. In [22] the authors proposed several strategies, including adaptive q-gram selection, for the efficient retrieval of the top-k results. In [18], the authors introduced deliberated techniques, e.g., divide-skip-merge, to extract similar strings.

Knowledge-based strategy. Knowledge base (sometimes called word thesauri), e.g., WordNet, contains the labeled (or semi-labeled) data for text related research tasks. In [19], 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 [14]. The word pair similarity is estimated from the hierarchy based on a node counting strategy, i.e., calculating the number of nodes between the target words.

Corpus-based strategy. Statistics information of large corpus can be used to calculate the similarity between two words or texts. 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) [4] 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 [10], [14]. The combination of knowledge based strategy and word order based strategy was proposed in [14]. In [10], the author applies string based, common word order based, and corpus based strategies to measure the similarity between sentences.

Currently, several works [6], [21] explore efficiency issue to optimize state-of-the-art similarity strategy. Efficient extraction on semantic similar words is presented in [21] by optimizing string-based, WordNet-based and corpus-based similarity strategies. In [6], the authors address efficiency issue to efficiently search for semantic similar sentences on three string similarity strategies and corpus-based strategy.

IX. Conclusion

In this paper, we have studied both effectiveness and efficiency aspect in the sentence similarity matching. The optimal strategies have been proposed through cross validation and dynamically weight tuning. We also introduced several efficient techniques to improve the performance of the similarity computation. The trade-off between effectiveness and efficiency is also explored by conducting extensive experiments.

References

[1] Blake, M.B., Cabral, L., König-Ries, B., Küster, U., Martin, D.: Semantic Web Services: Advancement through Evaluation. Springer (2012)

[2] Cohen, W.W.: Integration of heterogeneous databases without common domains using queries based on textual similarity. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '98, pp. 201–212 (1998)

[3] Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: Proceedings of the ACM SIGMOD symposium on Principles of Database Systems, PODS '01, pp. 102–113 (2001)

[4] Gabrilovich, E., Markovitch, S.: Computing semantic relatedness using wikipedia-based explicit semantic analysis. In: Proceedings of the International Joint Conference on Artifical Intelligence, IJCAI'07, pp. 1606–1611 (2007)

[5] Goyal, A., Daumé III, H.: Approximate scalable bounded space sketch for large data nlp. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, pp. 250–261 (2011)

[6] Gu, Y., Yang, Z., Nakano, M., Kitsuregawa, M.: Towards efficient similar sentences extraction. In: Proceedings of Intelligent Data Engineering and Automated Learning, IDEAL'12, pp. 270–277 (2012)

[7] 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 '99, pp. 203–212 (1999)

[8] Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Communications of ACM **18**(6), 341–343 (1975)

[9] Islam, A., Inkpen, D.: Second order co-occurrence pmi for determining the semantic similarity of words. In: Proceedings of the International Conference on Language Resources and Evaluation, LREC '06, pp. 1033–1038 (2006)

[10] 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)

[11] 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)

[12] Leacock, C., Chodorow, M.: Combining local context and wordnet similarity for word sense identification. In: WordNet: An Electronic Lexical Database, pp. 305–332. In C. Fellbaum (Ed.), MIT Press (1998)

[13] Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady **10**(8), 707–710 (1966)

[14] 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)

[15] 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'06, pp. 775–780 (2006)

[16] Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys **33**(1), 31–88 (2001)

[17] Pantel, P., Crestan, E., Borkovsky, A., Popescu, A.M., Vyas, V.: Web-scale distributional similarity and entity set expansion. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP'09, pp. 938–947 (2009)

[18] Sarawagi, S., Kirpal, A.: Efficient set joins on similarity predicates. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD '04, pp. 743–754 (2004)

[19] Tsatsaronis, G., Varlamis, I., Vazirgiannis, M.: Text relatedness based on a word thesaurus. Journal of Artificial Intelligence Research **37**, 1–39 (2010)

[20] Wiemer-Hastings, P.: Adding syntactic information to lsa. In: Proceedings of the Annual Conference of the Cognitive Science Society, COGSCI'00, pp. 989–993 (2000)

[21] Yang, Z., Kitsuregawa, M.: Efficient searching top-k semantic similar words. In: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'11, pp. 2373–2378 (2011)

[22] Yang, Z., Yu, J., Kitsuregawa, M.: Fast algorithms for top-k approximate string matching. In: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI'10, pp. 1467–1473 (2010)