A Study on Efficient Similar Sentence Matching

Yanhui GU†, Zhenglu YANG†, Miyuki NAKANO†, 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,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 study the efficiency and effectiveness 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. 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 on real datasets.

Key words rank aggregation, threshold algorithm, semantic, Top-k

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 [26], [37]. From a given sentence collection, this kind of queries asks for those sentences which are most semantically similar to a given one.

The problem can be solved as follows: we firstly measure the semantic similarity score between the query and each sentence in the data collection using the state-of-the-art techniques [15], [21], [25], [31], [33], 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 (i.e., precision) of the problem 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 [7], [8], [12], [13], [16], [27], [29], [40]. 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 strategy [15], [21], [33], that a threshold is predefined to filter out those dissimilar sequences. 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 [25], [33]; (2) corpus-based strategy [15], [17], [18], [34]; (3) syntax based strategy [15], [21]; and (4) hybrid strategy [15], [21], [34].

Naively testing every candidate sentence is time consuming, especially when the size of the sentence collection is huge. 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 [10]. As such, these issues are the challenges of the paper, which have not been studied before.

We propsed a propose to tackle the efficiency issue for searching top-k semantic similar sentences [41], which is different from previous works that focus on the effectiveness aspect. Based on the most comprehensive work [15], 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. In this paper, we study the most important issues of similar sentence mathcing, i.e., effectiveness and efficiency. We also study the trade-off between effectiveness and efficiency.

2. Efficient Similar Sentence matching

2.1 Preliminaries

To measure the similarity $sim(Q, P)$ between two sentences Q and P, we apply state-of-the art strategies by as-

sembling multiple similarity metric features [15], [21]. Given that we cannot evaluate all the similarity measurement strategies in this paper, we select several representative features based on the framework presented in [15]. 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 [15] 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 [11]). String similarity measurement strategies, including edit-distance, hamming distance and so on. We focus on three representative string similarity measurement strategies introduced in [15], namely, NLCS, NMCLCS_1 and NMCLCS_n (注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 PMI (SOC-PMI) [14], [15] 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.

2.2.3 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 [15] 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 [15], [21] for detail.

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, [15] 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 [41] as the evaluation base which is an optimization strategies on the framework which is proposed in [15]. The original are composed with the following: String (NLCS,NMCLCS_1 and NMCLCS_n), Semantic (corpus-based strategy) and Common word order (注2). Actually, they apply string and semantic strategies in the framework. Accordingly, [41] proposed efficient similar sentence matching strategy on string and semantic.

3. Experimental Evaluation

To evaluate effectiveness and efficiency, we conducted extensive experiments by using 16-core Intel(R) Xeon(R) E5530 server running Debian 2.6.26-2. All the algorithms were written in C language and compiled by GNU gcc. The baseline algorithm is implemented according to the state-of-the-art work [15].

In the whole experimental evaluation, we use three different datasets, i.e., the *benchmark* dataset which was used in [15], [21], BNC(注3) dataset and MSC(注4) dataset.

3.1 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 on two labeled datasets, i.e., the *benchmark* dataset and MCS dataset.

3.1.1 Evaluation on Precisely Labeled Dataset

The *benchmark* dataset which was used in [15], [21] and it has been labeled by concrete value, i.e., we can easily understand the closeness between measured result and labeled data. Here, we denote **Distance** as the metric which measures the closeness between two sets of results, e.g., baseline and our proposal, where x_i and y_i are two sets of results respectively. Smaller value indicates that the two results are

(注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 [15] for detail.

(注2)：We conducted experiments on *benchmark* dataset and found that common word order similarity has low importance in sentence similarity measurement.

(注3)：http://www.natcorp.ox.ac.uk/

(注4)：Microsoft Research Paraphrase Corpus. It contains 5801 pairs of sentences.

closer to each other. $\textbf{Distance} = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - y_i)^2$

The experimental result conducted on the *benchmark* dataset is illustrated in Fig. 1. 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.

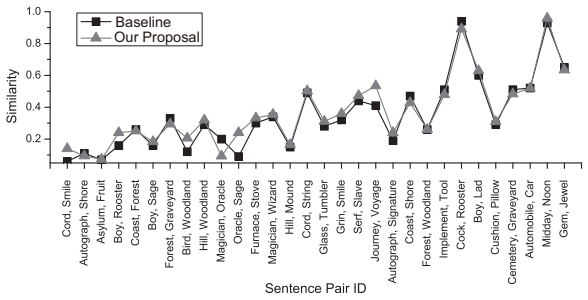


Fig. 1 Evaluation on effectiveness under benchmark dataset

3. 1. 2 Evaluation on Rawly Labeled Dataset

MSC is another labeled dataset which has been extracted from news sources on the web, along with human annotations indicating whether each pair captures a paraphrase-semantic equivalence relationship. They apply the value 1 as similar while the value 0 as dissimilar. However, such label data is rather raw and we cannot apply them to judge our measured results directly.

a) Raw data preprocessing

Since such dataset has no concrete value while only "1" or "0", we should distinct each value beforehand. Consider an example, there are two cases. Case (a): the similarity score between two sentences is 0.30 while the labeled data is 1. Case (b): the similarity score between two sentences is 0.60 while the labeled data is 0. From this example we can see that we cannot judge 0.30 or 0.60, which one represents "closeness". So, how to measure our experimental results by using such raw data. To answer this question, we apply a threshold τ as the bound, i.e., if any measured value is greater than or equal to τ and the labeled data is "1" , we regard it as "hit". Similarly, if one measured value is less than τ and the labeled data is "0", we also regard it as "hit", otherwise "miss". Table. 1 illustrates the strategy.

Labeled value	$\geqq \tau$	$< \tau$
1	hit	miss
0	miss	hit

Table 1 The selection of τ

Randomly selecting 20 sentence pairs from MSC and set different τ value, we can obtain the result which is presented in Table. 2. Here we set hit ratio be the number of hits divided by total results. With these different threshold τ, we select the optimal one that obtains the maximal hit ratio, i.e., τ=0.3 in this example.

Threshold	Hit	Miss	Hit Ratio
0.7	0	20	0%
0.6	1	19	5%
0.5	9	11	45%
0.4	12	8	60%
0.3	17	3	85%
0.2	15	5	75%
0.1	14	6	70%

Table 2 Different hit ratio under different selection of τ

Accordingly, we conducted extensive experiments to search the best τ value. Since we know that the similarity score comes from two different parts, i.e., string similarity and semantic similarity, and the weight α we apply before is 0.5. To evaluate the precise hit ratio under different τ, we evaluate the whole hit ratio by using MSC dataset under different τ and α value. Table. 3 shows the experimental results.

From the table, we can see, when τ=0.3 under α=0.6, we obtain the best hit ratio as 80.52%. However, the evaluation of each τ is under a fixed α, i.e., we cannot determine the combination of τ and α which can obtain the best hit ratio. Therefore, we should extensively train the value of α under different τ.

b) Training α under different τ

We apply a k-fold cross-validation(注5) strategy to check α under different τ. Firstly, we divide MSC dataset into k parts and apply the part from 1 to $k-1$ as the training data. Then we get α and τ under the best hit ratio and apply these two parameters to compute the hit ratio on the k^{th} part. In each step i, we can get the best hit ratio H_i and the hit ratio (test) Ht_i. For example, in the 1^{st} step, the best hit ratio occurs when $\alpha = 0.6$ and $\tau = 0.3$. Then we get the 81.24% hit ratio in testing part which means there is no distinguish difference between training and testing. After k steps finish, we get a list of values of α, τ and hit ratios. The finial α and τ will be the average value. From Table. 4, we can see that τ = 0.56 and α = 0.31 are the best parameters. We show the evaluation on the effectiveness for the MSC dataset in Fig. 2 under these two parameters.

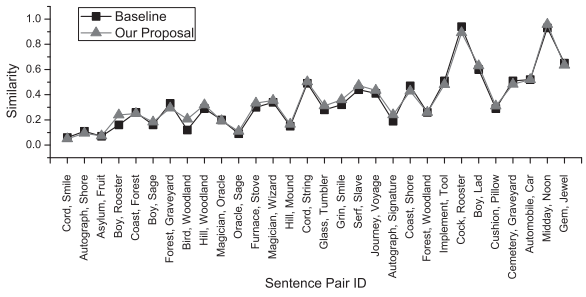


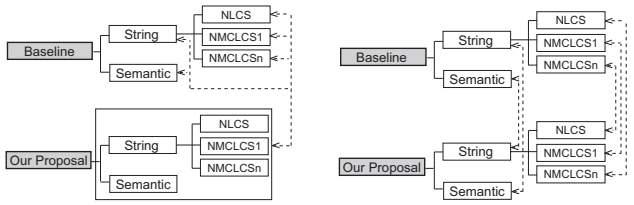
Fig. 2 Evaluation on effectiveness under MSC

0	20.55%	21.20%	36.25%	41.29%	36.06%	34.18%	18.86%	16.89%	16.43%
0.1	17.74%	24.67%	41.29%	41.39%	33.24%	28.44%	18.74%	15.14%	12.27%
0.2	56.73%	55.25%	53.51%	51.44%	46.13%	34.80%	30.32%	26.62%	22.76%
0.3	69.28%	72.33%	68.70%	63.39%	55.34%	49.53%	34.67%	22.51%	18.69%
0.4	77.71%	78.57%	68.85%	57.32%	51.46%	47.66%	36.36%	28.44%	22.31%
0.5	77.81%	79.16%	77.59%	74.37%	63.37%	51.27%	40.41%	32.77%	32.77%
0.6	71.02%	76.87%	**80.52%**	71.70%	63.21%	53.28%	39.05%	33.86%	22.29%
0.7	66.30%	71.07%	72.32%	63.06%	61.42%	39.91%	36.05%	30.01%	18.82%
0.8	34.84%	67.63%	69.92%	57.30%	50.99%	36.29%	34.22%	30.10%	21.32%
0.9	29.62%	36.32%	61.92%	53.28%	46.18%	41.13%	36.29%	30.81%	17.95%
1.0	20.84%	22.44%	21.34%	18.86%	16.95%	15.14%	12.72%	11.58%	10.36%

Table 3 Hit ratio under different τ and α

	Hit RatioH_i	parameter α	threshold τ	Hit-Ratio(test)Ht_i
1^{st}	81.25%	0.6	0.3	81.24%
2^{nd}	84.48%	0.7	0.3	81.58%
3^{rd}	78.72%	0.7	0.4	82.79%
4^{th}	80.46%	0.5	0.3	80.03%
5^{th}	80.67%	0.4	0.3	84.51%
6^{th}	84.18%	0.4	0.4	71.26%
7^{th}	79.10%	0.5	0.3	79.35%
8^{th}	82.16%	0.6	0.3	77.28%
9^{th}	82.07%	0.6	0.2	75.39%
10^{th}	80.42%	0.6	0.3	79.00%
	Avg.	**0.56**	**0.31**	

Table 4 10-fold cross-validation on τ under different α



(a) Single vs. Combination (b) Single vs. Single

Fig. 3 Two different evaluation strategies on trade-off between efficiency and effectiveness

3.2 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 (illustrated in Fig. 3(a)) 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, as illustrated in Fig. 3(a). Then we explore the performance of single feature in the baseline strategy vs. single feature in our framework as illustrated in Fig. 3(b).

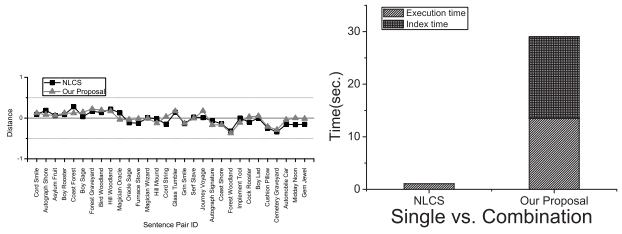
3.2.1 Single Strategy in the Baseline vs. Our Proposal

To evaluate the effectiveness, we apply single strategy in baseline and combination strategy(注6) 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 three different single string strategy in Fig. 4(a), 4(c) and 4(e). String strategy and semantic strategy results are listed in Fig. 4(g) and Fig. 4(i), respectively. 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), Fig. 4(d), Fig. 4(f), Fig. 4(h) and Fig. 4(j) present 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 NLCS vs. combination strategy pair as an example. Fig. 4(b) illustrates the execution time and the index time for NLCS (i.e., the left bar) and our proposal (i.e., the right

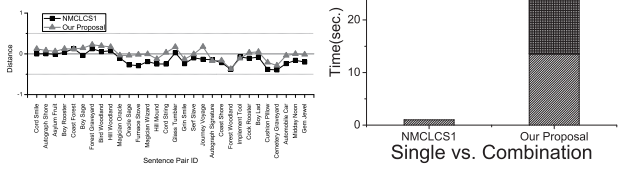
(注5)：We apply 10-fold cross-validation in this paper.

(注6)：Combination strategy means the whole framework strategies.



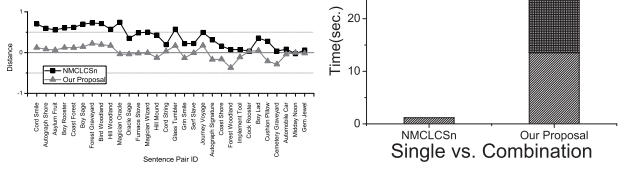
(a) NLCS vs. Our Proposal

(b) NLCS vs. Our Proposal



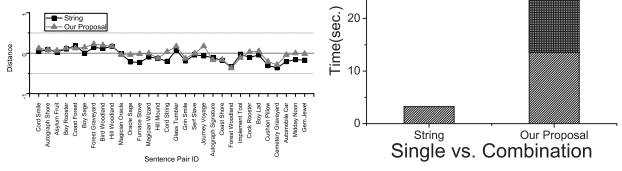
(c) NMCLCS_1 vs.OurProposal

(d) NMCLCS_1 vs.OurProposal



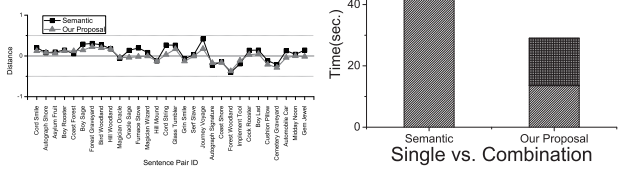
(e) NMCLCS_n vs.OurProposal

(f) NMCLCS_n vs. OurProposal



(g) String vs. Our Proposal

(h) String vs. Our Proposal



(i) Semantic vs. Our Proposal

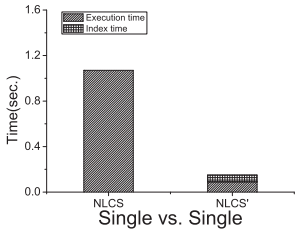
(j) Semantic vs. Our Proposal

Fig. 4 Evaluation on trade-off between efficiency and effectiveness

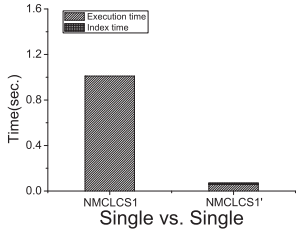
bar). We can easily see that the execution time of NLCS is very fast while the combination strategy consumes more time. The similar results can be seen in Fig. 4(d), Fig. 4(f) and Fig. 4(h). In Fig. 4(j), the execution time of single semantic strategy is longer than that of combination strategy. Such result tells us that the optimization on semantic similarity is crucial among all the optimization strategies.

3. 2. 2 Single Strategy vs. Single Strategy

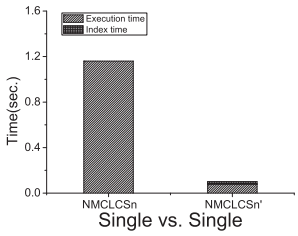
Evaluation on single strategy vs. combination strategy demonstrates the trade-off between effectiveness and efficiency. In this section, we study the performance of single feature in the baseline strategy vs. single feature in our framework (i.e., Fig. 3(b)). Firstly, we compare the effectiveness between the two situations, as illustrated in Fig. 5. 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.



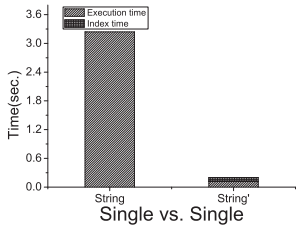
(a) NLCS vs. Our NLCS′



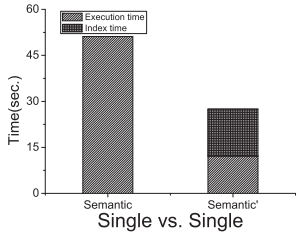
(b) NMCLCS_1 vs. NMCLCS'_1



(c) NMCLCS_n vs. NMCLCS'_n



(d) String vs. String′



(e) Semantic vs. Semantic′

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

4. Conclusion and Future Work

In this paper, we study the effectiveness and efficiency on similar sentence matching. Efficient searching top-k similar sentences is very important especially when the data is large. Several efficient strategies are introduced to test as few candidates as possible during the searching process. The comprehensive experimental evaluation demonstrates the efficiency of the proposed techniques while keeping the same high effectiveness as the state-of-the-art techniques. For further understand how optimization strategy works, we conducted extensive experiments on different types of datasets. To evaluate the effectiveness, we conducted on two different types of labeled datasets and trained two parameters for the efficiency evaluation. To understand the trade-off between effectiveness and efficiency, we evaluated different combination of features between the baseline and our proposal. In the future, we will incorporate more similarity strategies to

evaluate the efficiency and effectiveness of the framework.

References

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

[2] Diego Ceccarelli, Claudio Lucchese, Salvatore Orlando, Raffaele Perego, and Fabrizio Silvestri. Caching query-biased snippets for efficient retrieval. In Anastasia Ailamaki, Sihem Amer-Yahia, Jignesh M. Patel, Tore Risch, Pierre Senellart, and Julia Stoyanovich, editors, *EDBT*. ACM, 2011.

[3] Gobinda G. Chowdhury. *Introduction to modern information retrieval*. Facet, 3. ed. edition, 2010.

[4] Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR*. ACM, 2005.

[5] Ronald Fagin, Amnon Lotem, and Moni Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[6] P. W. Foltz, W. Kintsch, and T. K. Landauer. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 1998.

[7] Amit Goyal and Hal Daumè III. Approximate scalable bounded space sketch for large data nlp. In *EMNLP*, 2011.

[8] Amit Goyal, Hal Daumè III, and Suresh Venkatasubramanian. Streaming for large scale nlp: Language modeling. In *HLT-NAACL*, 2009.

[9] 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.

[10] Joseph M. Hellerstein, Ron Avnur, Andy Chou, Christian Hidber, Chris Olston, Vijayshankar Raman, Tali Roth, and Peter J. Haas. Interactive data analysis: The control project. *Computer*, 32, aug 1999.

[11] D. S. Hirschberg. A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, 1975.

[12] Ming Hua, Jian Pei, Ada Wai-Chee Fu, Xuemin Lin, and Ho fung Leung. Efficiently answering top-k typicality queries on large databases. In *VLDB*, 2007.

[13] Ming Hua, Jian Pei, Ada Wai-Chee Fu, Xuemin Lin, and Ho-Fung Leung. Top-k typicality queries and efficient query answering methods on large databases. *VLDB J.*, 18, 2009.

[14] Aminul Islam and Diana Inkpen. Second order co-occurrence pmi for determining the semantic similarity of words. In *LREC*, 2006.

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

[16] Jong Wook Kim, Ashwin Kashyap, Dekai Li, and Sandilya Bhamidipati. Efficient wikipedia-based semantic interpreter by exploiting top-k processing. In *CIKM*, 2010.

[17] 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.

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

[19] Thomas K Landauer and Susan T. Dutnais. A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 1997.

[20] C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identification*, pages 305–332. In C. Fellbaum (Ed.), MIT Press, 1998.

[21] 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.

[22] Ana Gabriela Maguitman, Filippo Menczer, Heather Roinestad, and Alessandro Vespignani. Algorithmic detection of semantic similarity. In *WWW*, 2005.

[23] Charles T. Meadow. *Text information retrieval systems.* Academic Press, 1992.

[24] Donald Metzler, Susan T. Dumais, and Christopher Meek. Similarity measures for short segments of text. In *ECIR*, 2007.

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

[26] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, 2004.

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

[28] Filip Radlinski, Andrei Broder, Peter Ciccolo, Evgeniy Gabrilovich, Vanja Josifovski, and Lance Riedel. Optimizing relevance and revenue in ad search: a query substitution approach. In *SIGIR*, 2008.

[29] Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

[30] Norvald H. Ryeng, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. Efficient distributed top-k query processing with caching. In *DASFAA*, 2011.

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

[32] Gerard M. Salton, Andrew K. C. Wong, and Chung-Shu Yang. A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 1975.

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

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

[35] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical Computer Science*, 92(1), 1992.

[36] Akrivi Vlachou, Christos Doulkeridis, Kjetil Nørvåg, and Michalis Vazirgiannis. On efficient top-k query processing in highly distributed environments. In *SIGMOD*, 2008.

[37] Kai Wang, Zhaoyan Ming, Xia Hu, and Tat-Seng Chua. Segmentation of multi-sentence questions: towards effective question retrieval in cqa services. In *SIGIR*, 2010.

[38] Furu Wei, Wenjie Li, Qin Lu, and Yanxiang He. Query-sensitive mutual reinforcement chain and its application in query-oriented multi-document summarization. In *SIGIR*, 2008.

[39] Zhenglu Yang and Masaru Kitsuregawa. Efficient searching top-k semantic similar words. In *IJCAI*, 2011.

[40] Xi Zhang and Jan Chomicki. Semantics and evaluation of top-k queries in probabilistic databases. *Distrib. Parallel Databases*, 2009.

[41] Yanhui Gu, Zhenglu Yang, Miyuki Nakano and Masaru Kitsuregawa. Towards efficient similar sentences extraction. *IDEAL*, 2012.