A Study on Efficient Semantic Similar Words Searching

Zhenglu YANG and Masaru KITSUREGAWA
Institute of Industrial Science, the University of Tokyo
Tokyo, Japan
{yangzl, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract

Abstract Measuring the semantic meaning between words is an important issue because it is the basis for many applications, such as word sense disambiguation, document summarization, and so forth. Although it has been explored for several decades, most of the studies focus on improving the effectiveness of the problem, i.e., precision and recall. In this paper, we study the efficiency issue, that given a collection of words, how to efficiently discover the most semantic similar words to the query. An extensive comparative experimental evaluation has been conducted to illustrate the effect of different strategies on the issue.

Key words Top-k, Similar Word Search, Efficiency

1 Introduction

Searching semantic similar words is an important issue because it involves many applications, such as query suggestion, word disambiguation, machine translation, and so forth. From a given collection of words, such queries ask for those words that are most (i.e., top-k) semantically similar to a given one.

Intuitively, the problem can be solved by firstly measuring the semantic similarity score between the query and each word in the collection using the state-of-the-art techniques [1, 6, 11, 14, 8, 3, 2, 13], and then sorting them with regard to the score, and finally extracting those top-k ones. However, the scale of the problem has dramatically increased and prevented existing strategies from conducting on large volume of data (i.e., the Web). Note that almost all the previous work focus on improving the effectiveness of the problem (i.e., precision and recall) yet this paper is to study the efficiency issue, which is rather challenging especially when conducting on large datasets. Another issue is that most of the previous works are rooted in a threshold-based framework and the similarity threshold is difficult to predefine by common users because many answers may be returned if the value is too small and there may be no answers if the value is too large. Therefore, searching for the top-k most similar words is an interesting problem.

The issue of finding similar words can be traced back to the 1940s [12] in the theory field where n-gram was first introduced. Two words are considered similar to each other if they have enough common subsequences (i.e., n-grams). While there are quite a few works on studying and extending n-gram based strategies [15] and they have been successfully applied in some applications such as spell checking, this line of work only takes into account the syntax of words and ignore the semantic meaning.

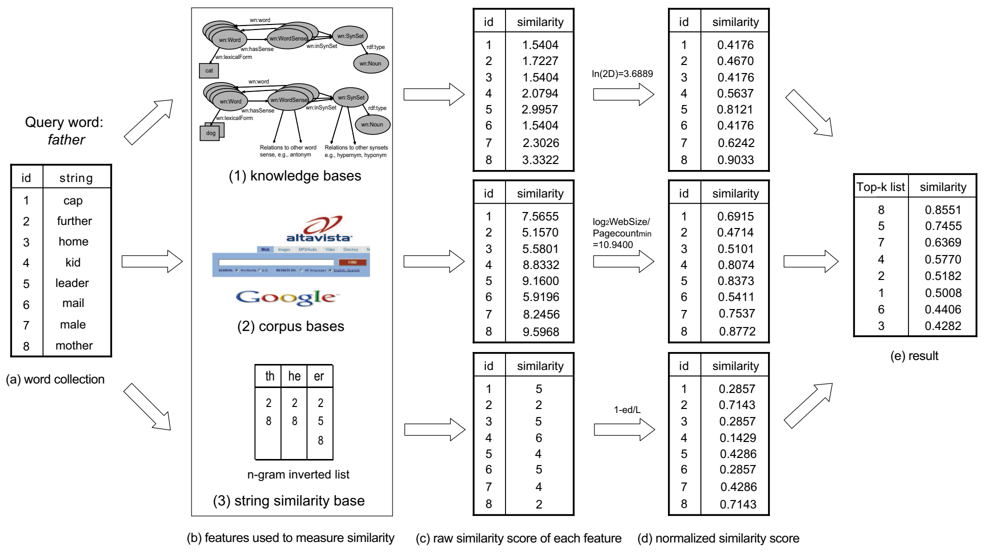
To remedy this problem, extensive studies have been explored and they can be classified into three main groups: (1) knowledge-based[1] strategies [10, 11]; (2) corpus-based strategies [14]; and (3) hybrid methods [2].

In this work we aim to study the efficiency and progressiveness issues of top-k semantic word searching.

2 Problem Statement and Analysis

We consider the top-k semantic similar word query on a given collection of words W. Formally, for a query word Q, finding a set of k words R in W most similar to Q, that is, $\forall r \in R$ and $\forall s \in (W - R)$ will yield $sim(Q, r) \geq sim(Q, s)$.

[1]It is also called dictionary-based or lexicon-based in the literature.


Query word:
father
(a) word collection
wn:word
wn:Word
wn:WordSense
wn:SynSet
wn:hasSense
wn:inSynSet
rdf:type
wn:lexicalForm
wn:Noun
cat
dog
Relations to other word sense, e.g., antonym
Relations to other synsets e.g., hypernym, hyponym
(1) knowledge bases
altavista
Google
(2) corpus bases
n-gram inverted list
(3) string similarity base
(b) features used to measure similarity
ln(2D)=3.6889
log2WebSize/ Pagecountmin =10.9400
1-ed/L
(c) raw similarity score of each feature
(d) normalized similarity score
(e) result

id	string
1	cap
2	further
3	home
4	kid
5	leader
6	mail
7	male
8	mother

th	he	er
2	2	2
8	8	5
		8

id	similarity
1	1.5404
2	1.7227
3	1.5404
4	2.0794
5	2.9957
6	1.5404
7	2.3026
8	3.3322

id	similarity
1	7.5655
2	5.1570
3	5.5801
4	8.8332
5	9.1600
6	5.9196
7	8.2456
8	9.5968

id	similarity
1	5
2	2
3	5
4	6
5	4
6	5
7	4
8	2

id	similarity
1	0.4176
2	0.4670
3	0.4176
4	0.5637
5	0.8121
6	0.4176
7	0.6242
8	0.9033

id	similarity
1	0.6915
2	0.4714
3	0.5101
4	0.8074
5	0.8373
6	0.5411
7	0.7537
8	0.8772

id	similarity
1	0.2857
2	0.7143
3	0.2857
4	0.1429
5	0.4286
6	0.2857
7	0.4286
8	0.7143

Top-k list	similarity
8	0.8551
5	0.7455
7	0.6369
4	0.5770
2	0.5182
1	0.5008
6	0.4406
3	0.4282

Figure 1: A general framework for searching top-k semantic strings

To measure the similarity $sim(Q, P)$ between two words Q and P, we apply the state-of-the-art strategy by assembling multiple similarity metric features [8, 2]. Because we focus on tackling the efficiency issue, for simplicity we select three representative features from the main categories in word similarity measuring literature.

- Knowledge-based Similarity: Word dictionaries, e.g., WordNet [9], have been acted as the knowledge base for text processing. In this study, we consider a representative metric, **Leacock & Chodorow** [5], defined as $Sim_{lch}(w_1, w_2) = -ln\frac{length(w_1,w_2)}{2*D}$, where $length$ is the length of the shortest path between two concepts (that the two words w_1 and w_2 belong to) using node-counting, and D is the maximum depth of the taxonomy.

- Corpus-based Similarity: Corpus-based measures of word semantic similarity try to recognize the degree of similarity between words using large corpora. The intuitive idea of corpus-based measures is that two words are similar to each other if they co-occur frequently in the large corpus. In this paper, we choose PMI-IR as one of the representative similarity measures between two words, w_1 and w_2, defined as, $Sim_{pmi-ir}(w_1, w_2){=}log_2\frac{pagecount(w_1 \cap w_2)*WebSize}{pagecount(w_1)*pagecount(w_2)}$, where $pagecount(w_i)$ is the number of documents retrieved when w_i is submitted to the search engine.

- String similarity: String similarity measures the difference of syntax between words. An intuitive idea is that two words are similar to each other if they have enough common subsequences (i.e., n-gram [12]). We focus on a representative string similarity measure, i.e., edit distance [7].

2.1 A General Framework

A general framework of searching top-k similar words is illustrated in Fig. 1, with a concrete example presented to ground our discussion. The query word and each candidate in the collection are sent to the three modules (i.e. knowledge bases, web search engine, and string similarity evaluator), respectively, to obtain the corresponding similarity score. Then, the scores from different modules are normalized, assembled, and ranked, resulting in the final ranked lists [8, 2].

For the example as illustrated in Fig. 1, all the eight candidates need to be evaluated with the query $father$ and the top-3 semantic similar words are $mother$, $leader$, and $male$.

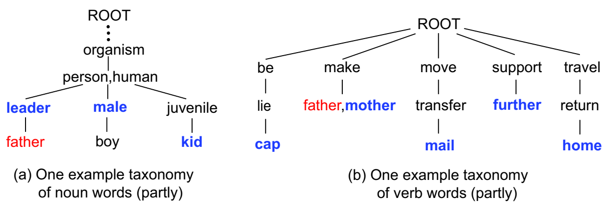


Figure 2: Top-k word searching on WordNet

3 Approaches

The assembling process on multiple similarity measures can be analog to the traditional rank aggregation issue [4], where each similarity measure in the former can be treated as a list in the latter. In this paper, we study several best first search strategies to facilitate accessing those top-k items (words) in each list (similarity measure feature).

3.1 Knowledge based Feature

WordNet is employed as the knowledge base in this paper. The traditional strategy is to measure the similarity of each candidate word and the query, by traversing the topology of WordNet. However, it is inefficient because every candidate needs to be tested. Fig. 2 illustrates the example aforementioned, where every pair of red word (query) and blue word (candidate) is evaluated.

We introduce a best first search strategy to efficiently obtain the top-k similar words. Because the query word could be polysemous, we address the issue when the query and the candidates exist in more than one taxonomy (i.e., Fig. 2 (a) and (b)). In this case, the ranking of a candidate not only depends on the shortest path in a taxonomy but also the maximum depth of the taxonomy (i.e., $\frac{D}{L}$, where D is the maximum depth and L is the shortest path). A best first search strategy can be constructed with regard to the value of $\frac{D}{L}$.

We illustrate the main process (i.e., order of nodes accessed) by using the aforementioned example. The sequentially accessed nodes should be: (1) $father$ in the noun taxonomy (i.e., $\frac{D}{L}$=20)[2]; (2) $father, mother$ in the verb taxonomy (i.e., $\frac{D}{L}$=14); (3) $leader$ in the noun taxonomy (i.e., $\frac{D}{L}$=10); and so forth.

[2]The maximum depths of the two taxonomies are 20 for noun and 14 for verb by querying WordNet in the preprocessing.

3.2 Corpus based Feature

To obtain the top-k similar words based on large corpus (i.e., the Web), the traditional strategy is to submit the query and each candidate to the search engine and compute their similarity (i.e., PMI-IR [14]). However, evaluating every candidate pair words may cause large computation cost (i.e., network delay time used to transfer words to search engine and return answer, computation time at server, and parsing time on the answer stream). We introduce a method to evaluate as few candidates as possible.

We sort all the candidates in ascending order of their $pagecount$ in the preprocessing, and measure the similarity while querying one by one. If we find that the $pagecount$ of the current candidate word is greater than $\frac{WebSize}{2^{\tau_{top-k}}}$, where τ_{top-k} is the top-k similar score so far, we can terminate the process and thus, avoid to evaluate the remaining candidates.

3.3 String Similarity Feature

There are not many studies on exploring how to efficiently search top-k similar words with respect to string similarity [16]. In this paper, we apply the existing efficient strategies. Specifically, they are (1) count filtering; (2) length filtering; and (3) divide-merge-skip.

3.4 Assembling Similarity Features

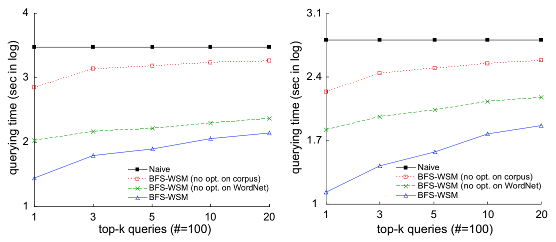
Given the progressively extracted words in each feature, i.e., the result obtained from the above sections, we apply an efficient approach to hasten the process of searching top-k similar words, based on the traditional rank aggregation algorithm [4].

4 Performance Analysis

We performed the experiments using a Intel(R) Core(TM) 2 Dual CPU PC (3GHz) with a 2G memory, running Redhat linux. We conducted experiments on two real life datasets: (1) $Dict$[3]; and (2) $Word$[4].

[3]http://www.aspell.net/
[4]http://wordnet.princeton.edu/

4.1 Progressiveness



(a) **Dict** dataset (b) **Word** dataset

Figure 3: Progressiveness of query performance

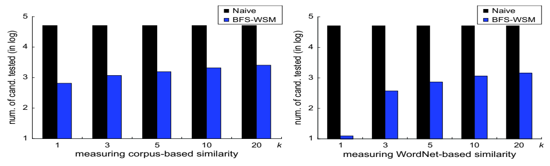
The progressiveness performance of the introduced algorithm was evaluated by varying k. We randomly selected 100 queries from the datasets and reported the average value. The result is shown in Fig. 3. We can see that $Naive$ outputs all the top-k results at the same time, and much slower than $BSF\text{-}WSM$ (i.e., more than two orders of magnitude slower for top-1 query on $Dict$ data, as illustrated in Fig. 3 (a)). $BSF\text{-}WSM$ can progressively extract more results the longer the execution time. For the effect of different optimization strategies, optimization on corpus-based similarity measuring seems to dominate the whole query processing (i.e., red line in the figure). This is not surprising because the cost of requesting word $pagecount$ to search engine is high. Moreover, we can see that the introduced two optimizations (i.e., measuring on WordNet and corpus similarities) account for most of the performance improvement.

4.2 Number of Candidate Words Accessed

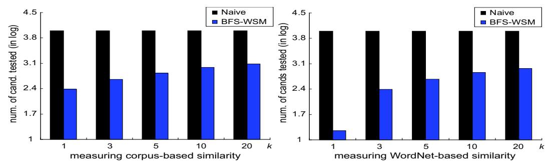
We evaluate the number of candidates tested in different algorithms when varying k. As shown in Fig. 4, we can see $BSF\text{-}WSM$ accesses much smaller sets of words compared with $Naive$. This is the intrinsic reason why it performed better on query processing (as shown in Fig. 3).

5 Conclusion

We have studied the top-k similar word searching problem. Several efficient strategies are introduced following the best first search manner. Several experiments are conducted and the results show that the introduced strategies can efficiently answer the top-k semantic similar word queries.



(a) #candidates vs k on **Dict** dataset



(b) #candidates vs k on on **Word** dataset

Figure 4: Number of words tested for different algorithms

References

[1] E. Agirre and G. Rigau. Word sense disambiguation using conceptual density. In *COLING*, 1996.

[2] D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW*, 2007.

[3] A. Budanitsky and G. Hirst. Evaluating wordnet-based measures of lexical semantic relatedness. *CL*, 2006.

[4] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *PODS*, 2001.

[5] C. Leacock and M. Chodorow. *Combining local context and WordNet similarity for word sense identification*. In C. Fellbaum (Ed.), 1998.

[6] C. Leacock, G. A. Miller, and M. Chodorow. Using corpus statistics and wordnet relations for sense identification. *CL*, 1998.

[7] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. Technical Report, 1966.

[8] Y. Li, Z. A. Bandar, and D. McLean. An approach for measuring semantic similarity between words using multiple information sources. *TKDE*, 2003.

[9] G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 1995.

[10] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *SMC*, 1989.

[11] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, 1995.

[12] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 1948.

[13] G. Tsatsaronis, I. Varlamis, and M. Vazirgiannis. Text relatedness based on a word thesaurus. *JAIR*, 2010.

[14] P. D. Turney. Mining the web for synonyms: Pmi-ir versus lsa on toefl. In *ECML*, 2001.

[15] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *TCS*, 1992.

[16] Z. Yang, J. Yu, and M. Kitsuregawa. Fast algorithms for top-k approximate string matching. In *AAAI*, 2010.