Exploiting Symmetry in Relational Similarity for Ranking Relational Search Results

Tomokazu Goto, Nguyen Tuan Duc, Danushka Bollegala, and Mitsuru Ishizuka

The University of Tokyo, Japan
{goto, duc}@mi.ci.i.u-tokyo.ac.jp,
danushka@iba.t.u-tokyo.ac.jp, ishizuka@i.u-tokyo.ac.jp

Abstract. Relational search is a novel paradigm of search which focuses on the similarity between semantic relations. Given three words *(A, B, C)* as the query, a relational search engine retrieves a ranked list of words $\mathfrak{D}$, where a word $D \in \mathfrak{D}$ is assigned a high rank if the relation between *A* and *B* is highly similar to that between *C* and *D*. However, if *C* and *D* has numerous co-occurrences, then *D* is retrieved by existing relational search engines irrespective of the relation between *A* and *B*. To overcome this problem, we exploit the symmetry in relational similarity to rank the result set $\mathfrak{D}$. To evaluate the proposed ranking method, we use a benchmark dataset of Scholastic Aptitude Test (SAT) word analogy questions. Our experiments show that the proposed ranking method improves the accuracy in answering SAT word analogy questions, thereby demonstrating its usefulness in practical applications.

Key words: relational search, relational similarity, symmetry

1 Introduction

Relational search is a novel search paradigm based on relational similarity of word pairs. For the query {*(A,B),(C,?)*}, in which *A*, *B*, and *C* are input words, a relational search engine finds the words *D* such that the relation between *A* and *B* is also held between *C* and *D*. A candidate answer *D* is assigned a high rank when the word pair *(C, D)* has a high degree of relational similarity with the word pair *(A, B)*. In previous methods for relational search [3] and relational similarity measure [1], the relation between two words in a word pair is represented by lexico-syntactic patterns that frequently co-occur with those words. However, this approach imposes a bias towards the frequency of a word – a high frequency word *D* has a higher probability of being assigned a top rank, irrespective of the semantic relation shared between *(A, B)* and *(C, D)*. We propose a ranking method which uses the symmetry in relational similarity to alleviate this phenomenon.

To demonstrate the proposed ranking method, let us consider the query {*(Google, Eric Schmidt), (Microsoft, ?)*}. Here, ”?” denotes an entity. *Steve Ballmer* is expected to be ranked at the top of the result list for this query because *Steve Ballmer* is the CEO of *Microsoft*, whereas *Eric Schmidt* is the CEO of *Google*. Moreover, when we use the inverse query {*(Eric Schmidt, Google), (?, Microsoft)*}, *Steve Ballmer* is also expected to be ranked as the first result. This is because relational similarity is invariant if both word pairs are inverted [4]. The invariance of relational similarity under a symmetric transformation of word pairs provides us with a practical method to rank candidates in a relational search engine: we can obtain a better ranking if we take into account the ranking in the inverse query's result list.

In addition, we propose "*complementary rank*" for improving the precision in ranking the result set of a relational search query. When *D* is assigned a high rank (i.e., top rank) in the query $\{(A, B), (C, ?)\}$, we can expect that *C* is also assigned a high rank in the query $\{(A, B), (?, D)\}$. Therefore, we can consider the rank of *C* in the query $\{(A, B), (?, D)\}$ as an additional criterion for ranking *D* in the query $\{(A, B), (C, ?)\}$. We call this additional criterion as the "complementary rank of *D*". In the proposal method, we combine the symmetric property and complementary rank to improve the initial ranking.

2 Related Work

The idea of relational search has been introduced in Veale [6] and Bollegala, et al. [1]. Kato, et al. first implemented **relational search** [3] by issuing queries to a keyword-based Web search engine. To extract candidate answers, they first query a Web search engine for terms or lexico-syntactic patterns that are likely to appear only in documents which contain both *A* and *B*. The extracted term or pattern set T is supposed to contain terms or lexical patterns that express the relations between *A* and *B*. Then, they use *C* and a term $t \in T$ to find documents that contain both *C* and *t*. The candidate answer set $\mathfrak{D}$ is then defined as the set of terms that are likely to appear only in those documents. Then, they rank the candidate set using the likelihood of co-occurrence of the term *D* with the pair *(C,t)*. Our method also uses lexico-syntactic pattern to express the relations between *A* and *B*. However, the pattern generation algorithm and the scoring scheme are different. In particular, they use only the words in the mid-fix between A and B for extracting lexical patterns that might represent relations between *A* and *B*. On the other hand, we use wildcards and an n-gram model which can precisely capture the relation between *A* and *B* [1].

Bunescu and Mooney proposed an approach for overcoming the problem of bias due to high frequency words as mentioned in previous section [2].However, their method needs a large amount of texts from Web documents for compute word frequencies. This can not be accomplished by using only snippets from a keyword-based Web search engine's results.

3 Method

To answer the query $\{(A, B), (C, ?)\}$, the proposed method first extracts lexical patterns that represent relations between *A* and *B*. The lexical patterns are n-grams of the context surrounding the pair *(A, B)* in a sentence. It then uses the keyword *C* along with these patterns to query a Web search engine for the answer *D*, similar to [3]. To improve the ranking of the results that are returned by the above procedure, we use the symmetry of relational similarity and complementary rank.

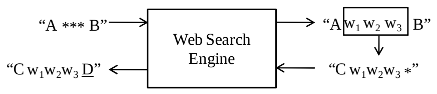

Fig. 1. Relational Search on the Web

Stem pair	ostrich	bird
1	**lion**	**cat**
2	goose	flock
3	ewe	sheep
4	cub	bear
5	primate	monkey

Fig. 2. An example SAT analogy question

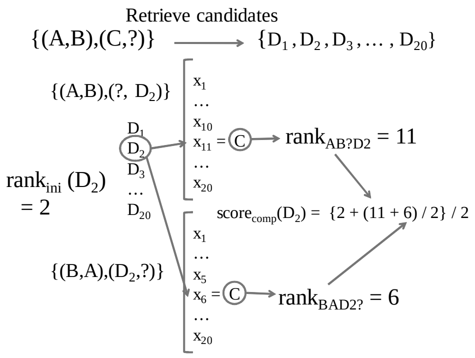

Fig. 3. Scoring candidates $\mathfrak{D}$ retrieved for the query $\{$*(A,B),(C,?)*$\}$

3.1 Relational Search on the Web

Fig. 1 shows the process to find the answer for the query $\{$*(A, B), (C, ?)*$\}$. First, we extract the semantic relation between A and B by issuing queries of type "A * * * B" to a Web search engine[1] to obtain some text snippets that include *A* and *B* separated by up to three words. Here, "*" denotes a wildcard for any word. To increase the similarity between two pairs that have similar contexts, we generate all n-grams ($n \leq 5$) which contain both two words in a word pair as lexical patterns for the pair. For instance, in the sentence *"big A such as B is considered to be ..."*, we generate sequences such as *"big A such as B"*, *"A such as B"* and *"A such as B is"*. We obtain the lexical patterns by replacing *A* with the variable α and *B* with β in the original sub-sequences: *"big α such as β"*, *"α such as β'* and *"α such as β is"*. To avoid noisy patterns, we ignore all patterns whose frequencies are smaller than a frequency threshold ξ. We denote the set of these patterns by *P*.

To get candidate answers, for each pattern $p \in P$ we input the query *"p[C/α, */β]"* (including the double qoutes) to the search engine. The formula p[C/α] represents the substitution of α by *C* in the pattern p. For this query, the search engine returns snippets which include *C* and other words in the pattern *p* and some extra words in this order. For example, for the query "lion is a large *", the search engine returns snippets such as "lion is a large cat ..." or "lion is a large four-legged animal ...". Because we want to get the word at the position of the wildcard * in the query, we add the those extra words into the candidate answer set $\mathfrak{D}$. We then rank the a candidate $D \in \mathfrak{D}$ using the following ranking score:

$$\text{score}_{\text{init}}(D) = \frac{\sum_{p \subseteq P_D}(\text{freq}(\text{“}p[C/\alpha, D/\beta]\text{”}))}{\text{freq}(\text{“}C***D\text{”})}. \quad (1)$$

In Formula 1, P_D are the patterns that appeared with *D*, freq("p[C/α, D/β]") is the frequency of co-occurrences of the word *D* with the word *C* and other words in the

[1] Yahoo Boss API http://developer.yahoo.com/search/boss/

patterns. Because the number of words between C and D is less than three, we normalize the sum by dividing the sum of freq("p[C/α, D/β]") by the hit count of the query *"C *** D"*. Finally, we assign a rank to each $D \in \mathfrak{D}$ using the score in Fomula 1. We call this ranking as the **initial ranking**. The ranking score $\mathrm{score}_{\mathrm{init}}(D)$ is called the initial ranking score.

3.2 Symmetry in Relational Similarity

In the initial ranking, a candidate *D* might receive a top rank merely because it frequently occurs with *C* irrespective of the relation between *A* and *B*. To solve this problem, we propose a ranking score using the symmetry in relational similarity. Let us denote the relational similarity between *(A, B)* and *(C, D)* by $R((A,B),(C,D))$. Relational similarity will remain unchanged under certain permutations of the four words (e.g., $R((A,B),(C,D)) = R((B,A),(D,C))$). Therefore, the candidates that are ranked at the top by one form of the query (e.g., *(A,B),(C,?)*) must also be ranked at the top by the other (alternative) forms of the query (e.g., *(B,A),(?,C)*). In other words, if *D* is an incorrect candidate, then it will be ranked at the top only in a small number of alternative forms of the query and it will receive bad ranks in almost all alternative forms. To consider the symmetric property, we define the score of *D* as follows:

$$\mathrm{score}(D) = \frac{\mathrm{score}_{\mathrm{comp}}(D) + \mathrm{score}_{\mathrm{compR}}(D)}{2}. \quad (2)$$

In the above formula, $\mathrm{score}_{\mathrm{comp}}(D)$ is the score of *D* in the query $\{$*(A,B),(C,?)*$\}$ when we take into account the complementary rank (we will explain complementary rank in the next section). Similarly, $\mathrm{score}_{\mathrm{compR}}(D)$ is the score of *D* in the other forms of the query whose similarities are invariant to a symmetric transformation (e.g., $\{$*(B,A),(?,C)*$\}$).

In addition to symmetry, we use complementary rank of *C* or *D* to rank candidate answers in a relational search engine. The complementary rank of a candidate *D* in the query $\{$*(A, B), (C, ?)*$\}$ is the initial rank of *C* in the query $\{$*(A, B), (?, D)*$\}$ and vice versa. We define the score of *D* by using complementary rank as follows,

$$\mathrm{score}_{\mathrm{comp}}(D) = \frac{\mathrm{rank}_{\mathrm{ini}}(D) + \frac{\mathrm{rank}_{\mathrm{AB?D}}(C) + \mathrm{rank}_{\mathrm{BAD?}}(\mathrm{C})}{2}}{2}, \quad (3)$$

where $\mathrm{rank}_{\mathrm{ini}}(D)$ is the rank of *D* in the initial ranking (i.e., ranking by $\mathrm{score}_{\mathrm{init}}(D)$ as shown in Fomula 1), $\mathrm{rank}_{\mathrm{AB?D}}(C)$ is the initial rank of *C* in $\{$*(A, B),(?, D)*$\}$ and $\mathrm{rank}_{\mathrm{BAD?}}(C)$ is the initial rank of *C* in $\{$*(B, A),(D, ?)*$\}$. We denote the score of *D* in initial ranking of $\{$*(A, B),(C, ?)*$\}$ as $\mathrm{score}_{\mathrm{comp}}(D)$ and the score of *D* in initial ranking of $\{$*(B, A),(?, C)*$\}$ as $\mathrm{score}_{\mathrm{compR}}(D)$. By combining the Formula 2 and 3, we obtain the final score of D ($score(D)$) for ranking candidates $D \in \mathfrak{D}$.

We illustrate the process of calculating $\mathrm{score}_{\mathrm{comp}}(D)$ in Figure 3 in the query $\{$*(A, B),(C, ?)*$\}$. We assign *D* a high rank if *C* is assigned high ranks when we use the queries $\{$*(A, B),(?, D)*$\}$ and $\{$*(B, A),(D, ?)*$\}$.

4 Evaluation

4.1 Experiments

To evaluate the proposed ranking algorithm, we use the SAT dataset [1, 5]. The SAT dataset contains 374 word analogy questions selected from the Scholastic Aptitude Test.

Each questions has a question word pair (stem pair) and five choices for answer word pairs, in which the correct pair has the highest similarity with the stem pair as shown in Fig. 2. Therefore, we use the following method for solving SAT analogy questions.

Calculating the score of a word in the search result set:
Given a stem word pair *(A,B)* and a choice word pair *(C,D)* (e.g., *A* is *ostrich*, *B* is *bird*, *C* is *lion* and *D* is *cat*), we first perform the query $\{$*(A,B),(C,?)*$\}$ to obtain a candidate answer set $\mathfrak{D}$. Using the Formula 1, we rank the set $\mathfrak{D}$ to get the initial ranking. Suppose that the rank of D in this ranking is N^D. Next, we perform the query $\{$*(A,B),(?, D)*$\}$ to obtain a candidate set and record the rank (according to the score in Formula 1) N_1^C of *C* in this set. Similarly, we use the query $\{$*(B,A),(D, ?)*$\}$ to get the rank of *C* as N_2^C. Finally, we define the SAT candidate score of D using the following formula:

$$SATSubScore(D) = \frac{N^D + \frac{N_1^C + N_2^C}{2}}{2} \tag{4}$$

Score of a SAT candidate answer:
We calculate the score of a SAT candidate word pair $c = (C, D)$ as follow

$$SATScore(c) = \frac{SATSubScore(C) + SATSubScore(D)}{2} \tag{5}$$

After calculating SATScore for each candidate SAT answer, we select the choice whose score is minimal as the answer to the SAT question. To evaluate the performance, we compare the answer that our system outputs with the correct answer.

4.2 Results

We obtain 105 correct answers before using the symmetry and complementary rank. After using symmetry and complementary rank, we get 114 correct answers. Table 1 shows the experimental results. When we do not retrieve the word *C* or *D* for all five choices, we can not use the queries $\{$*(A,B),(C,?)*$\}$ or $\{$*(A,B),(?,D)*$\}$ respectively. In such cases, we can not estimate our method's effect, so we also measure the performance when we ignore those cases. After eliminating such cases, only 243 questions remain. For those questions, the proposed method achieved an accuracy of 46.9% when use the symmetry, whereas in initial ranking it is only 43.0%.
To measure our method's effect, we consider questions including correct answers and two or more answer candidates which include *C* or *D*. This results in 216 questions in which we made 78 correct answers (36.1%) before utilizing symmetry and complementary rank and 87 correct answers (40.3%) after. Therefore, by using symmetry and complementary rank, we could obtain 4.2% improvement in the SAT result.

Table 1. Comparison of correct rates

Criterion	Initial ranking	Using symmetry and complementary rank
# correct / # questions (recall)	28.1%	30.5%
# correct / # questions that we can get *C* or *D* (precision)	43.0%	46.9%
# correct / # questions that we can retrieve the correct choice and at least one other choice	36.1%	40.3%

5 Discussion

We observe that the use of symmetry and complementary rank improves the initial ranking. This shows that the proposed ranking method can be effectively applied to rank relational search results. Especially, the proposed method of exploiting symmetry of relations can be combined with advanced lexical pattern extraction techniques (e.g., PrefixSpan algorithm, etc.) to drastically improve the precision of relational search. Furthermore, one can improve the precision by combining existing relational search scoring algorithm such as [3] with the proposed scoring algorithm. Therefore, the proposed method can be smoothly integrated with other existing methods for ranking relational search results. The integration can be done easily because the proposed method exploits a special aspect of relations (i.e., the symmetry of relations) that is not utilized in existing approaches. It is worth noting that relational search is the first task concerning relational similarity in which complementary rank can be exploited and therefore be invented. In other tasks such as similarity measuring [1, 5], complementary rank does not appear because in those tasks, the four words in the two pairs (A, B) and (C, D) are all given. On the other hand, in relational search or tasks in which one or more words are not given, we can define complementary rank to represent the strength of the relation between the candidate word and the input query word.
It is worth noting that the evaluation using SAT benchmark gives an interesting criterion for evaluating performance of a relational search engine, which can not be easily evaluated using normal criteria such as F-score or MRR (mean reciprocal rank).

6 Conclusion

We implemented relational search by using web search engine and proposed a ranking method for relational search. There are some noisy candidate words in the initial ranking of relational search results. To eliminate noisy candidate words from the initial ranking, we used a symmetric property and complementary rank. By using these features, we could improve 4.2% of precision. This shows that our proposed method of using symmetric property is effective for improving correct rate on SAT dataset and ranking relational search results.

References

1. Bollegala, D., Matsuo, Y., Ishizuka, M.: Measuring the similarity between implicit semantic relations from the web. In: Proc. of WWW'09. pp. 651–660 (2009)
2. Bunescu, R.C., Mooney, R.: Learning to extract relations from the web using minimal supervision. In: Proc. of ACL'07. pp. 576–583 (2007)
3. Kato, M.P., Ohshima, H., Oyama, S., Tanaka, K.: Query by analogical example: relational search using web search engine indices. In: Proc. of CIKM'09. pp. 27–36 (2009)
4. Medin, D., Goldstone, R., Gentner, D.: Respects for similarity. Psychological Review 6(1), 1–28 (1991)
5. Turney, P., Littman, M., Bigham, J., Shnayder, V.: Combining independent modules to solve multiple-choice synonym and analogy problems. In: Proc. of RANLP'03. pp. 482–486 (2003)
6. Veale, T.: The analogical thesaurus. In: Proc. of 15th Innovative Applications of Artificial Intelligence Conference (IAAI'03). pp. 137–142 (2003)