Dependency Relation Matching for Answer Selection

Renxu Sun Hang Cui Keya Li Min-Yen Kan Tat-Seng Chua

Department of Computer Science
School of Computing
National University of Singapore

{sunrenxu, cuihang, likeya, kanmy, chuats}@comp.nus.edu.sg

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Retrieval Models; I.2.7 [**Artificial Intelligence**]: Natural Language Processing;

General Terms

Algorithms, Measurement, Experimentation

Keywords

Question Answering, Answer Selection, Dependency Relation Matching

1. INTRODUCTION

Open domain question answering (QA) has become a popular research area in recent years. Most current QA systems search for answers in three major stages: document retrieval, passage retrieval and answer selection. As QA requires exact answers, answer selection is a crucial module in a QA system.

Many answer selection techniques have been proposed. Density-based ranking, which considers the surface distance between question terms and the answer target, is employed by most state-of-the-art QA systems. While density-based ranking may be effective for questions with answer targets of certain named entity (NE) types, it may fare poorly with questions whose answer target type is unknown, for instance: "What does AARP stand for?" For such questions, there is a much larger answer candidate space to choose from, and the density-based method often does not have enough information to pinpoint the correct answer. In addition, even for those questions with known NE-type answer targets, lexical level matching often leads to false positives [1].

To overcome such problems, we use dependency relations between matched question terms and the answer target as additional evidence to pinpoint the correct answer. We have applied dependency relation matching to passage retrieval in our past work [1]. We now focus on answer selection and extend dependency relation matching by using WordNet to allow flexible relation alignment.

2. THE APPROACH

In this section, we briefly describe the modules in our factoid QA system, particularly the answer selection module.

2.1 Document and Passage Retrieval

We use Lucene[1] as our document retrieval system. We adopt a density-based passage ranking algorithm with query expansion as described in [2] to rank the relevance of passages. We use single sentence as passage since the dependency parsers such as Minipar [3] can only analyze a single sentence as input. We choose the top ranked 50 sentences as input for answer selection.

2.2 Answer Selection

We employ Minipar to generate dependency trees for each top ranked sentence from the passage retrieval module. A dependency tree depicts dependency relations between nodes, *i.e.,* tokens of a sentence (a token can be a single word, a noun phrase or a verb phrase). For each pair of adjacent nodes in the tree, an edge is labeled with the relation between them from modifier to head. For any two tokens in a sentence, a relation path exists between their nodes in the tree. The path consists of a series of relations attached to the intermediate nodes. As such, we define a relation triple in the form of (Start_Token, Relation_Path, End_Token). Our answer selection is conducted on relation triples. The node with the most similar relations with other question term nodes compared to the relations they have in the question is selected as the answer.

2.2.1. Relation Similarity Learning

Due to variations in natural language texts, the same relation is often phrased differently in questions and answer sentences. Instead of performing exact matching between relations, we adopt a statistical method to learn the relatedness of relations from training data.

We accumulate around 1,000 factoid question-answer pairs from TREC 2001 and 2002 QA tasks to build our statistical model. In order to align relations between question terms and answer terms, we substitute a general tag <ANS> for those question targets in questions and answer sentences. We then use Minipar to parse the questions and answer sentences. For each question and answer, relation paths from question triples are generated and aligned with those from answer triples if their starting and ending tokens are the same after stemming. This results in 2,557 relation path pairs for model construction. The relatedness of two relations is measured by a variation of mutual information:

$$MI(\text{Rel}_0, \text{Rel}_1) = \log \frac{\sum \alpha \times \delta(\text{Rel}_0, \text{Rel}_1)}{f_Q(\text{Rel}_0) \times f_A(\text{Rel}_1)}$$

where Rel_0 and Rel_1 are the relations extracted from the question paths and the answer paths respectively. $f_Q(Rel)$ and $f_A(Rel)$ represent the number of occurrences of *Rel* in question paths and answer paths. $\delta(\text{Rel}_0, \text{Rel}_1)$ is 1 when relations Rel_0 and Rel_1 occur in a question path and its corresponding answer path respectively, and 0 otherwise. α is inversely proportional to the number of relations appearing in the question and in the answer.

We compute pair-wise similarity for all dependency relations. These relation similarities form the basis for calculating relation path

1 http://jakarta.apache.org/lucene/docs/index.html

similarity at the answer selection stage. Other techniques for estimating these pair-wise relation similarities can be found in [1].

2.2.2 Answer Selection

We retrieve 50 top ranked sentences from the passage retrieval module. For each sentence, we perform NE tagging. For each question, we use two different approaches to select a group of answer candidates, depending on the question type. For questions that require a specific NE type as the answer target, we use all the tokens of that NE type from the top 50 sentences as our target set. For questions requiring no known NE types as answer targets, we use all verb phrases or noun phrases to form the answer candidate set.

For sentences in the answer candidate set, we extract relation paths by aligning the matched terms in the question and answer sentences. Note that the <ANS> node and the candidate answer node are considered to be a match. However, we notice that sentences containing potential answers may not use the exact term used in the question. For example, "Who did Capriati *beat* in the French Open final?" can be answered by "Capriati *defeat* Graf by 2-1 in French final last night". In this case, "beat" and "defeat" should be matched. We thus define two matching criteria, namely *strict matching* and *flexible matching*. With strict matching, two terms are matched if and only if they have the same stem form. With flexible matching, we use WordNet to find semantically related words. Specifically, for a particular term $\boldsymbol{Q_0}$, we use WordNet to extract terms that are in $\boldsymbol{Q_0}$'s gloss ($\boldsymbol{G_Q}$) and synset ($\boldsymbol{S_Q}$) in terms of its top three senses according to its POS tag. We define the matched term pair between the question term $\boldsymbol{Q_0}$ and any term in the answer candidate sentence ($\boldsymbol{T_0}$) by:

$$Match\{Q_0, T_0\} \Leftrightarrow (T_0 \in G_Q \vee T_0 \in S_Q) \vee (stem(Q_0) = stem(T_0))$$

These matched terms can be used to expand terms in the starting and ending nodes for a relation path, and thus increase the chance of a match. Given two relation paths P_Q and P_A, we first treat relations along the path as a sequence, and try all possible alignments of the relation sequence between the question triple and the answer triple. Among all possible alignments, we only choose the one that maximizes the sum of mutual information. The path similarity is calculated as:

$$Sim(P_Q, P_A) = \frac{\varepsilon}{1 + len(P_Q)^{len(P_A)}} \operatorname*{argmax}_{all\_possible\_alignments} \left(\sum MI(\mathrm{Rel}_i^Q, \mathrm{Rel}_j^A)\right)$$

where ε is a constant, and $len(P_Q)$ and $len(P_A)$ denote the length of the relation sequence in the question triple and the answer triple respectively. Finally, the score of the answer candidate is calculated as:

$$Score(Q, Ans) = \sum_{matched\_triples\_containing\_Ans} Sim(P_Q, P_A)$$

Based on this score, we select the highest ranked answer string to be the final answer.

3. EVALUATIONS

To evaluate our approach to selecting answers, we use 200 factoid questions from the TREC 2004 QA task and 380 factoid questions from TREC 2003 as our test set. Note that we exclude from the test set those questions that have no answers. We set up three systems for comparison: We use a state-of-the-art density-based answer selection module as the baseline system (BS) [4], and we implement two relation-based answer selection modules, S1 and S2. S1 uses strict token matching (Section 2.2.2 without WordNet extensions) while S2 uses flexible token matching (using WordNet extensions). We further divide the questions into two types according to whether their answer target types are known. We list the experimental results in Tables 1 and 2.

Table 1. Performance comparison on TREC 2004 questions
%Imp. denotes percentage improvement over the baseline system (BS)

	BS	S1 (%Imp.)	S2 (%Imp.)
Overall average accuracy	0.51	0.62 (22%)	0.65 (27%)
For questions with NE-type targets	0.68	0.78 (15%)	0.81 (19%)
For questions without NE-type targets	0.29	0.42 (45%)	0.44 (52%)

Table 2. Performance comparison on TREC 2003 questions

	BS	S1 (%Imp.)	S2 (%Imp.)
Overall average accuracy	0.44	0.53 (20%)	0.56 (27%)
For questions with NE-type targets	0.57	0.67 (18%)	0.70 (23%)
For questions without NE-type targets	0.22	0.29 (32%)	0.33 (50%)

Our main observations from the tables are: (a) Our approach using dependency relations produces significant improvement over the baseline system for both TREC 2003 and 2004 questions. The improvement is more significant for questions in which the answer type is unknown. In such cases, the baseline system will have a much larger candidate set to choose from and the density-based approach is reduced to blind search. (b) The result obtained by flexible matching is not significantly better than that obtained by strict matching. We conjecture that the main reason is WordNet does not capture the variation between nouns and verbs. For example, the answer to the question "*Who did XXX marry*?" may appear in the form "*XXX's wife YYY*". In this case, WordNet fails to capture the similarity between "marry" and "wife". Another reason is that although approximate matching criteria may increase matching recall, it also introduces noise in answer extraction, especially when the matched term has different senses and cannot be substituted simply by its synonyms from other senses.

4. REFERENCES

[1] H. Cui, R. Sun, K. Li, M.-Y. Kan and T.-S. Chua, *Question Answering Passage Retrieval Using Dependency Relations*, In Proc. of SIGIR 2005, Salvador, Brazil, 2005.

[2] H. Cui, K. Li, R. Sun, T.-S Chua and M.-Y. Kan, *National University of Singapore at the TREC-13 Question Answering Main Task*, Proc. of TREC-13, 2004.

[3] D. Lin, *Dependency-based Evaluation of MINIPAR,*. Workshop on the Evaluation of Parsing Systems, Granada, Spain, 1998.

[4] H. Yang, H. Cui, M.-Y. Kan, M. Maslennikov, L. Qiu and T.-S. Chua, *QUALIFER in TREC-12 QA Main Task,* In Proc. of TREC-12, 2003.