Question Answering Passage Retrieval Using Dependency Relations

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

Department of Computer Science
School of Computing
National University of Singapore

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

ABSTRACT

State-of-the-art question answering (QA) systems employ term-density ranking to retrieve answer passages. Such methods often retrieve incorrect passages as relationships among question terms are not considered. Previous studies attempted to address this problem by matching dependency relations between questions and answers. They used strict matching, which fails when semantically equivalent relationships are phrased differently. We propose fuzzy relation matching based on statistical models. We present two methods for learning relation mapping scores from past QA pairs: one based on mutual information and the other on expectation maximization. Experimental results show that our method significantly outperforms state-of-the-art density-based passage retrieval methods by up to 78% in mean reciprocal rank. Relation matching also brings about a 50% improvement in a system enhanced by query expansion.

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; I.7.1 [**Document and Text Processing**]

General Terms

Algorithms, Measurement, Experimentation

Keywords

Question Answering, Dependency Parsing, Passage Retrieval

1. INTRODUCTION

Passage retrieval has long been studied in information retrieval [11]. It aims to search for more precise and compact text excerpts in response to users' queries, rather than providing whole documents. Recently, passage retrieval has become a crucial component in question answering (QA) systems. Most current QA systems employ a pipeline structure that consists of several modules to get short and precise answers to users' questions. A typical QA system searches for answers at increasingly finer-grained units: (1) locating the relevant documents, (2) retrieving passages that may contain the answer, and (3) pinpointing the exact answer from candidate passages.

Passage retrieval (Step 2) greatly affects the performance of a QA system. If a passage retrieval module returns too many irrelevant passages, the answer extraction module is likely to fail to pinpoint the correct answer due to too much noise. Also, a passage can sufficiently answer a question. Lin *et al.* [16] showed that users prefer passages to phrase-long answers because passages provide sufficient context for them to understand the answer.

Tellex *et al.* [19] conducted a thorough quantitative component evaluation for passage retrieval algorithms employed by state-of-the-art QA systems. The authors concluded that neglecting crucial relations between words is a major source of false positives for current lexical matching based retrieval techniques. The reason is that many irrelevant passages share the same question terms with correct ones, but the relations between these terms are different from those in the question. We illustrate this by a sample question and some candidate sentences in Figure 1, where only sentence S1 contains the correct answer. The other three sentences share many question terms (in italics) but are incorrect.

<Question> What percent of the nation's cheese does Wisconsin produce?

Correct: <S1> In *Wisconsin*, where farmers *produce* roughly 28 *percent* of the *nation*'s *cheese*, the outrage is palpable.

Incorrect: <S2> … the number of consumers who mention California when asked about *cheese* has risen by 14 *percent*, while the number specifying *Wisconsin* has dropped 16 *percent*.

Incorrect: <S3> The wry "It's the *Cheese*" ads, which attribute California's allure to its *cheese* _ and indulge in an occasional dig at the *Wisconsin* stuff" … sales of *cheese* in California grew three times as fast as sales in the *nation* as a whole 3.7 *percent* compared to 1.2 *percent*, …

Incorrect: <S4> Awareness of the Real California *Cheese* logo, which appears on about 95 *percent* of California *cheeses*, has also made strides.

Figure 1. Sample question and candidate passages illustrating that lexical matching does not lead to the correct answer.

To address this problem, we propose a novel fuzzy relation matching method which examines grammatical dependency relations between question terms to improve current passage retrieval techniques for question answering. We employ Minipar [15], a fast and robust dependency parser, to accomplish dependency parsing. While previous work [2, 12] attempted to match dependency relations to extract answers, we present a statistical technique for measuring the degree of match of pertinent relations in candidate sentences with their corresponding relations in the question. Sentences that have similar relations between question terms are preferred. We perform fuzzy matching instead of strict matching because the same relationship is often phrased differently in the parse trees of the question and the answer. For instance, appositive relations can be rephrased using other dependency relations – such as the *whn* (nominal wh-phrase) relation – in the question. As such, strict matching of relations may fare poorly in recall, which is an important consideration in passage retrieval. Specifically, for non-trivial question terms, we represent all single relations between any two terms (or nodes) in the parse tree as a *relation path*. The overall likelihood of a candidate sentence in terms of dependency relations is the combination of the matching scores of all relation paths between matched question terms. We employ a variation of the statistical translation model to calculate the matching score of a relation path given another. In order to learn the mapping scores between relations in questions and potential answer sentences, we collect past question-answer pairs and train the mapping scores using two training methods: one based on mutual information and the other on expectation maximization.

We conduct a series of extrinsic experiments to demonstrate the effectiveness of fuzzy relation matching for passage retrieval on the TREC-12 QA task data. When applied on top of standard density-based lexical matching systems, our relation matching method significantly improves these systems by 50 to 78 percent in mean reciprocal rank (MRR). We also examine how two other QA parameters interact with relation matching in passage retrieval: query length and query expansion. A key finding is that longer queries benefit more from utilizing relations. To show state-of-the-art performance, we apply fuzzy relation matching to a QA system that is reinforced by query expansion and obtain a further 50% enhancement. We also show that a full QA system employing relationship matching reaches the top performance in TREC, without parameter tuning.

This paper is organized as follows. In the next section, we review related work. Section 3 presents the details of our fuzzy relation matching method. We show our detailed experimental results next, and conclude the paper with directions for future work.

2. RELATED WORK

Research in question answering (QA) has been catalyzed by the Text Retrieval Conference (TREC) series since 1999. Almost all QA systems fielded at TREC employ some passage retrieval technique to reduce the size of the relevant document set to a manageable number of passages. The simplest passage retrieval method, employed by MITRE [14], counts the number of matched question terms in a passage. Other passage retrieval systems, such as those employed in SiteQ [13] and IBM [10], are density-based as they take into account the distances between question terms in the candidate passages.

The goal of passage retrieval is to identify passages similar to the question in semantic content. While differing in specifics, all existing passage retrieval algorithms rely only on lexical level matching to rank passages. The underlying assumption is that each question term is considered an independent token. However, this simplification does not hold in many cases because dependency relations exist between words. Some work has been done to address this problem. To extract precise answers, Harabagiu *et al.* [8] applied a theorem prover that conducts abductive reasoning over WordNet to derive semantic relationship between words. However, their method may not be applicable to information retrieval due to its high computational cost. Other techniques attempt to approximate such relations between words statistically. For instance, some language modeling approaches capture simple dependency relations by using bigrams (e.g., [18]). However, these models only capture dependency relations between adjacent words. Recently, Gao *et al.* [7] proposed a language model that captures dependency relations that are learned from training data. They proposed a statistical parsing model that captures dependency relations between words based on co-occurrences of words in the training data.

While existing methods model dependency relations statistically at the surface level, we adopt Minipar to extract dependency relations. The reason is three-fold: (1) Different from information retrieval, we do not have a large amount of QA data for training. Using relation matching based entirely on statistics could be problematic due to the sparse data. (2) QA questions are sentences, which enable us to adopt a dependency parser to extract various types of dependency relations. Such typed relations, which have more accurate meanings in expressing dependency relationships, tend to be of higher differentiating capability in filtering out irrelevant relations. (3) Unlike Gao *et al.*, we seek to build a system with an off-the-shelf parser so that the system and its results are easier to reproduce. Minipar is a free research dependency parser that fulfills this requirement.

Minipar has been used in question answering in the past. PiQASso [2] employed Minipar as its dependency parser and extracts the answer from a candidate sentence if the relations reflected in the question are matched in that sentence. However, that system does not perform well due to low recall resulting from matching relations in only the top ranked sentences. To overcome the recall problem, Katz and Lin [12] indexed and matched specific relations (*e.g.,* subject-verb-object) over an entire QA corpus. However, they performed their evaluation on only a handful of manually constructed questions instead of community-standard TREC data.

Both the above systems select answers based on strict matching of dependency relations. Strict matching is problematic when conducted on a large corpus because relations between the same pair of words often differ between question and answer sentences. To overcome this problem, Cui *et al.* [5] recently proposed a statistical method of measuring the similarity of the relations to rank exact answer candidates.

The above work focuses on utilizing relations in the answer extraction task by filtering out unsuitable answer candidates. Such a task requires stricter matching because only relations related to the question target should be examined. In contrast, utilizing relations in passage retrieval adds another criterion for ranking, and benefits from examining relations between all question terms. As such, we feel that relation matching can achieve more effective results in passage retrieval than in answer extraction. Our hypothesis is that relation matching can boost precision while maintaining high recall in passage retrieval.

3. FUZZY RELATION MATCHING FOR PASSAGE RETRIEVAL

In this section, we discuss how fuzzy relation matching is performed in detail. We first present how relation paths are extracted and paired from parse trees. We then adopt a variation of IBM translation model 1 [4] to calculate the matching score of a relation path given another, which combines the mapping scores of single relations in both paths.

We present two methods to learn a pairwise relation mapping model from training data: one is based on a variation of mutual information (MI) that captures the bipartite co-occurrences of two relations in the training data, and the other is based on the iterative training process presented in [4] using expectation maximization (EM).

3.1 Extracting and Pairing Relation Paths

We first extract relation paths between words from dependency trees for sentences generated by Minipar. In Figure 2, we illustrate the dependency trees for the sample question and the answer sentence S1 presented in Figure 1.

Figure 2. Dependency trees for the sample question and sentence S1 in Figure 1 generated by Minipar. Some nodes are omitted due to lack of space.

Question:			
Path_ID	Node1	Path	Node2
<P_{Q1}>	Wisconsin	<*subj*>	produce
<P_{Q2}>	produce	<*head, whn, prep, pcomp-n*>	cheese
<P_{Q3}>	nation	<gen>	cheese
S1:			
<P_{S1}>	Wisconsin	<*pcomp-n, mod, i*>	produce
<P_{S2}>	produce	<*obj, mod, pcomp-n*>	cheese
<P_{S3}>	nation	<*gen*>	cheese

In a dependency tree, each node represents a word or a chunked phrase, and is attached with a link representing the relation pointing from this node (the governor) to its modifier node. Although dependency relations are directed links, we ignore the directions of relations. This is because the roles of terms as governor and modifier often change in questions and answers. The label associated with the link is the type of dependency relation between two nodes. Examples of relation labels (or relations for short) are *subj* (subjective), *mod* (modifying) and *pcomp-n* (nominal complement of a preposition). There are 42 such relation labels defined in Minipar.

We further define a relationship path (or simply *path*) between nodes n_1 and n_2 as the series of edges that traverse from n_1 to n_2, as in [17]. In this way, our system is able to capture long dependency relations. For simplicity, we consider a path a vector $\mathbf{P}$ <Rel_i>, where Rel_i denotes single relations. In Figure 3, we illustrate several paths extracted from two parse trees.

We impose two constraints when extracting paths:

(1) The path length cannot exceed a pre-defined threshold. The length of a path is defined as the number of relations in the path. In our configuration, the threshold is set to 7 based on our experiments on a small validation dataset. The purpose is to exclude exceptionally long paths as Minipar only resolves nearby dependencies reliably.

(2) We ignore relation paths between two words if they belong to the same chunk (which is usually a noun phrase or a verb phrase), as determined by Minipar. For instance, we ignore the relation between "28" and "percent" in "28 percent" because they belong to the same NP chunk as parsed by Minipar. A similar example is "New" and "York" in "New York".

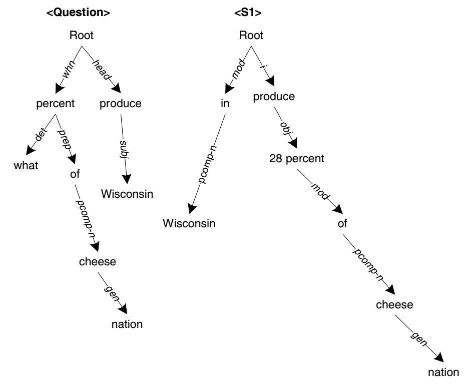

Figure 3. Illustration of relation paths extracted from the dependency trees in Figure 2.

To determine the relevance of a sentence given another sentence in terms of dependency relations, we need to examine how similar all the *corresponding paths* embedded in these two sentences are. We determine such paired corresponding paths from both sentences by matching their nodes at both ends. For instance, P_{Q1} and P_{S1} are paired corresponding paths with the matched nodes "Wisconsin" and "produce". Note that we match only the root forms of open class words (or phrases), such as nouns, verbs and adjectives, when pairing corresponding paths.

3.2 Measuring Path Matching Score

After extracting and pairing relation paths from both a question and a candidate sentence, we need to measure the matching score of the paths extracted from the sentence according to those from the question. For instance, in Figure 3, we calculate and combine the matching scores of the paths <*pcomp-n, mod, i*>, <*obj, mod, pcomp-n*> and <*gen*> based on their corresponding counterparts from the question: <*subj*>, <*head, whn, prep, pcomp-n*> and <*gen*> respectively. This example also illustrates that in real corpora, the same relationship between two words is often represented by different combinations of relations. We conjecture that such variations in relations hinder existing techniques (*e.g.*, [2, 12]) that attempt to use strict matching to achieve significant improvements over lexical matching methods. In contrast, we approach this problem by employing a fuzzy method to achieve approximate relation matching.

We derive the matching score between paths by extending IBM statistical translation model 1. While statistical translation model has been applied in information retrieval [3] and answer extraction [6], our use of it for the task of matching dependency relation paths is new. We treat the matching score of a relation path from a candidate sentence as the probability of translating to it from its corresponding path in the question. Let us denote two paired corresponding paths from question Q and sentence S respectively as P_Q and P_S, whose lengths are represented as m and n. The translation probability $Prob(\mathbf{P}_S | \mathbf{P}_Q)$ is the sum over all possible alignments:

$$\Pr ob(P_S \mid P_Q) = \frac{\varepsilon}{m^n} \sum_{\alpha_1=1}^{m} \cdots \sum_{\alpha_n=1}^{m} \prod_{i=1}^{n} P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_{\alpha_i}^{(Q)}) \quad (1)$$

where $\mathrm{Re}l_i^{(S)}$ stands for the i^{th} relation in path $\mathbf{P}_S$ and $\mathrm{Re}l_{\alpha_i}^{(Q)}$ is the corresponding relation in path $\mathbf{P}_Q$. The alignments of relations are given by the values of α_i which indicates the corresponding relation in the question given relation $\mathrm{Re}l_i^{(S)}$. ε stands for a small constant. $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$ denotes the relation translation probability, *i.e.,* relation mapping scores, which are given by a translation model learned during training and will be described in the next subsection. Unlike in the original application of machine translation, we assume that every relation can be translated to another; thus, we do not include a NULL relation in position 0. Note that $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$ is 1 when Rel_i and Rel_j are identical because the translation probability is maximized when a relation is translated to itself.

While IBM model 1 considers all alignments equally likely, we consider only the most probable alignment. The reason is that, unlike text translation that works with long sentences, relation paths are short. Most often, the most probable alignment gives much higher probability than any other alignments. We calculate the alignment by finding the most probable mapped relation in the path from the question for each relation in the path from the sentence based on relation translation probability. As such, the path translation probability is simplified as:

$$\Pr ob(P_S \mid P_Q) = \frac{\varepsilon}{m^n} \prod_{i=1}^{n} P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_{A_i}^{(Q)}) \quad (2)$$

where A_i denotes the most probable alignment. Moreover, we can use only the length n of the path $\mathbf{P}_S$ in normalizing Equation (2). Since we rank all candidate sentences according to the same question, the length of each path extracted from the question is constant, and does not affect the calculation of the translation probability. We take the log-likelihood of Equation (2) and remove all constants. The matching score of $\mathbf{P}_S$ is as follows:

$$\begin{aligned} MatchScore(P_S) &= \Pr ob(P_S \mid P_Q) \\ &= \frac{\varepsilon'}{n} \sum_{i=1}^{n} \log P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_{A_i}^{(Q)}) \end{aligned} \quad (3)$$

where n is used as a normalization factor and ε' is a small constant.

Finally, we sum up the matching scores of each path from the sentence which has a corresponding path in the question to be the *relation matching score* of the candidate sentence given the question. This score reflects how well the candidate sentence's relations match those of the question: a high score indicates that the question terms are likely to be used with the same semantics as in the question, and that the sentence is more likely to contain a correct answer.

3.3 Model Training

We have described in the above section how to obtain a relation matching score between a sentence and the question, and that this process requires a relation mapping model as input, *i.e.,* $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$ in Equation (3). In this subsection, we show how the mapping model can be acquired by two statistical methods from training question-answer pairs: one based on mutual information (MI) and the other based on expectation maximization (EM).

The assumption is that paired corresponding paths extracted from training QA pairs are semantically equivalent. Thus, the relation mapping between such training answer sentences and questions can be used as a model for unseen questions and potential answers as well. We use Minipar to parse all the training questions and corresponding answer sentences. Relation paths extracted from the question are paired with those from answer sentences, as described in Section 3.1.

We first employ a variation of mutual information[1] to calculate relation mapping scores. The relatedness of two relations is measured by their bipartite co-occurrences in the training path pairs. Different from standard mutual information, we account for path length in our calculation. Specifically, we discount the co-occurrence of two relations in long paths. The mutual information based score of mapping relation $\mathrm{Re}l_j^{(Q)}$ to relation $\mathrm{Re}l_i^{(S)}$ is calculated as:

$$P_t^{(MI)}(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)}) = \log \frac{\sum \gamma \times \delta(\mathrm{Re}l_j^{(Q)}, \mathrm{Re}l_i^{(S)})}{|\mathrm{Re}l_j^{(Q)}| \times |\mathrm{Re}l_i^{(S)}|} \quad (4)$$

where $\delta(\mathrm{Re}l_j^{(Q)}, \mathrm{Re}l_i^{(S)})$ is an indicator function which returns 1 when $\mathrm{Re}l_j^{(Q)}$ and $\mathrm{Re}l_i^{(S)}$ appear together in a training path pair, and 0 otherwise. γ is the inverse proportion of the sum of the lengths of the two paths. $|Rel^{(Q)}|$ stands for the number of paths extracted from all questions in which relation *Rel* occurs. Likewise, $|Rel^{(S)}|$ gives the number of paths extracted from all answer sentences that contain relation *Rel.*

In the second configuration, we employ GIZA [1], a publicly available statistical translation package, to implement IBM translation model 1 training over the paired training paths. Each relation is considered a word and each corresponding path pair is treated as a translation sentence pair, in which the path from a question is the source sentence and the path from the answer sentence is the destination sentence. The resulting word translation probability table is used to define relation mapping score $P_t(\mathrm{Re}l_i^{(S)} \mid \mathrm{Re}l_j^{(Q)})$. GIZA performs an iterative training process using EM to learn pairwise translation probabilities. In every iteration, the model automatically improves the probabilities by aligning relations based on current parameters. We initialize the training process by setting translation probability between identical relations to 1 and a small uniform value for all other cases, and then run EM to convergence.

4. EVALUATIONS

In this section, we present empirical evaluation results to assess our relation matching technique for passage retrieval systems. We have two hypotheses to test in our experiments:

1) The relation matching technique improves the precision of current lexical matching methods. Moreover, the proposed

[1] We use frequencies instead of probabilities in Equation 4 to approximate mutual information and use the logarithm to scale the result.

fuzzy relation matching method outperforms the strict matching methods proposed in previous work.

2) Long questions are more suitable for relation matching. We hypothesize that the effectiveness of relation matching is affected by question length. Long questions, with more question terms, have more relation paths than short questions, and benefit more from relation matching.

3) Relation matching also brings further improvement to a system that is already enhanced with query expansion because of the high precision it allows. We test whether the fuzzy relation matching technique brings further improvement to a passage retrieval system that uses query expansion.

4.1 Experiment Setup

We use the factoid questions from the TREC-12 QA task [20] as test data and the AQUAINT corpus to search for answers. We use TREC-12 test data because the questions are long enough to obtain corresponding relation paths to perform relation matching. We accumulate 10,255 factoid question-answer pairs from the TREC-8 and 9 QA tasks for use as training data, which results in 3,026 unique corresponding path pairs for model construction using both MI and EM based training methods.

There are 413 factoid questions in the TREC-12 task, from which 30 NIL-answer questions are excluded because they do not have answers in the corpus. TREC-12 had a passage retrieval task which used the same factoid questions as the main task except it accepted longer answers (250 bytes). Since we intend to evaluate passage retrieval techniques, we create the gold standard based on the official judgment list for the passage retrieval task provided by TREC. For each question, we generate a list of passages that are judged to be correct and supported by the corpus in the judgment list as standard answer passages. We cannot create the gold standard for 59 of the questions because no correct passages for them were judged by TREC evaluators. This leaves us with a final test set of 324 QA pairs, on which all evaluations in this paper are based. While Tellex *et al.* [19] made use of TREC-supplied exact answer patterns to assess returned passages, we observe that common answer patterns can be matched in incorrect passages as answer patterns are usually very short. We therefore use a stricter criterion when judging whether a passage is correct: it must be matched by the exact answer pattern, and additionally, it must have a cosine similarity equal to or above 0.75 with any standard answer passage.

Similar to the configuration used by Tellex *et al.* [19], we use the top 200 documents for each question according to the relevant document list provided by TREC as the basis to construct the relevant document set for the questions. If the 200 documents do not contain the correct answer, we add the supporting documents that have the answer into the document set. We conduct different passage retrieval algorithms on the document set to return the top 20 ranked passages. Note that the optimal passage length varies across different retrieval algorithms. For instance, SiteQ is optimized to use a passage length of three sentences [19]. In our evaluations for relation matching techniques, we take one sentence as a passage, as Minipar can only resolve intrasentential dependency relations. But for SiteQ, we still use the three-sentence window to define a passage.

We use four systems for comparison:

1) MITRE (baseline): This approach simply matches stemmed words between question and answer.

2) Strict Matching of Relations: A system that uses strict matching of relations to rank sentences. It employs the same technique as fuzzy matching to extract and pair relation paths, but it counts the number of exact path matches as its ranking score.

3) SiteQ: One of the top performing density-based systems in previous work. We follow the adaptation described in [19] in our implementation.

4) NUS [5]: Another top-performing factoid question answering system. We utilize its passage retrieval module, which is similar to SiteQ except that it uses single sentences as passages and calculates sentence ranking scores by iteratively boosting a sentence's score with adjacent sentence scores.

We employ three performance metrics: mean reciprocal rank (MRR), percentage of questions that have no correct answers, and precision at the top one passage. The former two metrics are calculated on the returned 20 passages by each system.

4.2 Performance Evaluation

In the first experiment, we evaluate the overall performance of our relation matching technique compared to other passage retrieval systems.

We apply both strict and fuzzy matching of relations in our experiments. We perform relation matching on the MITRE and NUS systems but not on SiteQ as it retrieves multiple-sentence passages, in which cross-sentence dependencies cannot be modeled by our system. For simplicity, we linearly combine the normalized lexical matching score obtained by MITRE or NUS and the relation matching score to obtain the overall ranking score of a sentence. In calculating fuzzy relation matching scores, we utilize the two relation mapping score models generated by both the MI-based and EM-based training methods. We illustrate the evaluation results in Table 1. From the table, we draw the following observations:

1) Applying relation matching over lexical matching methods boosts system performance dramatically. Applied on top of the MITRE and NUS systems, both strict and fuzzy relation matchings augment performance in all metrics significantly. When integrating strict relation matching with the NUS system, MRR improves by 35% and 31% over the results obtained by the standard NUS and SiteQ systems respectively. Relation matching also yields better precision in the top one passage task. When fuzzy relation matching is applied on top of NUS, the system achieves even better results. Here, all improvements obtained by relation matching are statistically significant as judged by using paired t-test [9] ($p < 0.001$). We believe that the improvement stems from the ability of the relation matching technique to model dependency relationships between matched question terms. Thus, many false positive sentences that would be favored by normal bag-of-word approaches are subsequently eliminated as they often do not contain the correct relations between question terms.

   Interestingly, even strict matching of relations significantly improves the performance of a passage retrieval system while

Table 1. Overall performance comparison of MRR, percentage of incorrectly answered questions (% Incorrect) and precision at top one passage. Strict relation matching is denoted by Rel_Strict, with the base system in parentheses. Fuzzy relation matching is denoted by Rel_MI or Rel_EM for both training methods. All improvements obtained by relation matching techniques are statistically significant (*p<0.001*).

Passage retrieval systems	MITRE	SiteQ	NUS	Rel_Strict (MITRE)	Rel_Strict (NUS)	Rel_MI (MITRE)	Rel_EM (MITRE)	Rel_MI (NUS)	Rel_EM (NUS)
MRR	0.2000	0.2765	0.2677	0.2990	0.3625	0.4161	0.4218	0.4756	0.4761
% MRR improvement over									
MITRE	N/A	+38.26	+33.88	+49.50	+81.25	+108.09	+110.94	+137.85	+138.08
SiteQ	N/A	N/A	N/A	+8.14	+31.10	+50.50	+52.57	+72.03	+72.19
NUS	N/A	N/A	N/A	+11.69	+35.41	+55.43	+57.56	+77.66	+77.83
% Incorrect	45.68%	37.65%	33.02%	41.96%	32.41%	29.63%	29.32%	24.69%	24.07%
Precision at top one passage	0.1235	0.1975	0.1759	0.2253	0.2716	0.3364	0.3457	0.3889	0.3889

work in answer extraction (*e.g.,* [2]) seems to be hindered by strict matching. We conjecture that the passage retrieval task is less constraining than answer extraction as the latter has to match relations of the identified target for the question. As such, we feel passage retrieval is more likely to benefit from relation matching.

2) Fuzzy relation matching outperforms strict matching significantly. When integrated with the NUS system, it gains a statistically significant improvement of 31% in MRR and 43% in precision at top one passage when using fuzzy matching of relations over strict matching. Note that while strict matching does not bring large improvements in terms of percentage of incorrect questions compared to lexical matching methods, the fuzzy relation matching method decreases such errors by 34% in comparison to NUS and by 56% compared to MITRE. Strict matching often fails due to variations in representing the same relationship because of parsing inconsistency and the flexibility exhibited in natural language. Such interchangeability between relations is captured by fuzzy matching methods. In this way, our statistical model is able to accommodate the variation in natural language texts.

3) Using MI and iterative EM to train relation mapping scores does not make any obvious difference in our tests. However, we present both training methods because they differ in complexity and scalability. The MI method has lower complexity compared to the EM method because it does not perform any alignment of relations during training, as it uses relation co-occurrences as approximations to relation mapping. The EM training process does alignment by improving the probability of alignment iteratively. We conjecture that the EM training method could outperform the MI method if a larger amount of training data is available. MI-based mapping scores are likely to be more susceptible to noise when scaling up. The EM training method is unlikely to suffer due to its gradual improvement mechanism. However, we cannot show the scalability of the two training methods given our limited test and training data.

4.3 Performance Variation to Question Length

It seems intuitive that longer questions are likely to benefit more from relation matching than shorter questions. The rationale is that more relation paths in longer sentences lead to more reliable relation ranking scores. In this experiment, we examine the effect of varying the number of non-trivial question terms on MRR.

Among the 324 questions in our test set, the number of question terms varies from one to 13, after removing trivial stop words such as "what". In Figure 4, we plot the MRR values along with 95% error bars of the systems that apply fuzzy relation matching with EM training on top of the MITRE and NUS systems when question length is varied. We consider only questions with two to six non-trivial question terms because there are less than 10% of questions with fewer than two or more than six question terms in our test set.

From Figure 4, we can see that as indicated by little overlap of the error bars, MRR nearly monotonically increases when more terms are present in the question. This is evidence that longer questions are more likely to improve with relation matching. We surmise that with more paired corresponding paths, relation matching based ranking would be of higher precision.

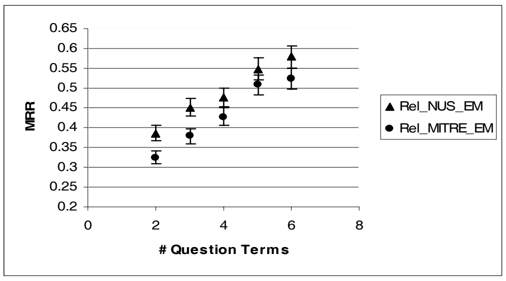

Figure 4. Illustration of MRR variation to change in number of question terms.

Note that number of question terms is only an approximation of number of actual paired corresponding relation paths. However, as the number of relation paths extracted for each question varies more than the number of question terms does, our small test data prevents us from conducting thorough experiments to examine the effect of number of relation paths on matching. Future work on a larger dataset can be done to reinforce the results shown here.

4.4 Error Analysis for Relation Matching

Although we have shown that relation matching greatly improves passage retrieval, there is still plenty of room for improvement. A key question is whether we can further characterize the types of questions that are adversely affected by relationship matching. Based on the above two experiments, we perform micro-level error analysis on those questions for which relation matching degrades performance. We find that fuzzy relation matching sometimes fails with incorrectly paired relation paths mainly for the following two reasons:

1) Mismatch of question terms: In some cases, the paths are incorrectly paired due to the mismatch of question terms. For instance, given the question #1912 "*In which city is the River Seine?*", the correct answer should be "*Paris*". Without question analysis and typing, the relation matching algorithm mistakenly takes "*city*" as a question term, instead of recognizing it as the question target. Thus, sentences containing all three question terms, *i.e.,* "*city*", "*river*" and "*Seine*", are ranked high while the correct answer does not contain "*city*". To overcome this problem, we need to incorporate question analysis in the passage retrieval system such that the question target and the answer candidate of the expected type can be matched when corresponding relation paths are paired.
2) Paraphrasing between question and answer sentences: Some correct sentences are paraphrases of the given question. In this case, both lexical matching and relation matching are likely to fail. Consider the question: "*What company manufactures X?*" The correct sentence is: "*… C, the manufacturer of X …*". The system needs to resolve such a paraphrase as "*C is the manufacturer of X → C manufactures X*" to answer this kind of questions. Lin and Pantel [17] attempted to find paraphrases (also by examining paths in Minipar's output parse trees) by looking at common content between the two nodes at both ends of relations. However, their method is limited as it relies on abundant training data to find inference rules between specific relations.

4.5 Performance with Query Expansion

As discussed above, short questions and paraphrases are obstacles in enhancing performance using relation matching. State-of-the-art QA systems adopt query expansion (QE) to alleviate such problems [5, 10, 8]. Here, we show how performance varies when the relation matching technique is reinforced by query expansion.

We conduct simple query expansion as described in [5], which submits the question to Google and selects expansion terms based on their co-occurrences with question terms in result snippets. We use the same method as described in the first two experiments to linearly combine the lexical matching score with query expansion and the relation matching score. We list the evaluation results in Table 2.

With query expansion, the performance of NUS (the lexical matching based system) again improves greatly. Specifically, query expansion reduces the percentage of incorrect answers from 33% to 28.4%. This is close to the figures obtained by relation matching methods without query expansion as listed in Table 1. This shows that query expansion boosts recall using expansion terms, allowing the system to answer more questions correctly.

When relation matching is incorporated into the NUS system along with query expansion, MRR values are boosted by 49%, which is statistically significant. This demonstrates that our relation matching technique can help re-rank passages to allow higher precision when the system is equipped with query expansion.

Table 2. Comparison of performance with query expansion. All showed improvements are statistically significant (*p-value<0.001*).

Passage Retrieval Systems	NUS (baseline)	NUS+QE	Rel_MI (NUS+QE)	Rel_EM (NUS+QE)
MRR (% improvement over baseline)	0.2677	0.3293 (+23.00%)	0.4924 (+83.94%)	0.4935 (+84.35%)
% MRR improvement over NUS+QE	N/A	N/A	+49.54%	+49.86%
% Incorrect	33.02%	28.40%	22.22%	22.22%
Precision at top one passage	0.1759	0.2315	0.4074	0.4074

However, query expansion does not boost the performance of systems with relation matching as significantly as compared to the improvement over the baseline lexical based system without query expansion. Comparing Tables 1 and 2, the improvement in performance for a system with query expansion is about 2% in MRR (from 0.4756 to 0.4924 when using MI training and from 0.4761 to 0.4935 when using EM training). We believe that this is caused by the simple policy we use to integrate lexical matching with relation matching. Since we just sum up matching scores, our relation matching model does not take full advantage of query expansion because external expansion terms do not have relation paths with the original question terms in the question. As such, expansion terms do not improve the relation path pairing process in our current system.

5. Case Study: Constructing a Simple System for TREC QA Passage Task

In the above experiments, we conducted component evaluations for passage retrieval for factoid questions. A natural question is whether the incorporation of relation matching into a standard QA system can yield good performance. Such a fully-fledged QA system adds query expansion, question typing and named entity extraction on top of simple passage similarity. In this case study, we construct a simple QA system on top of the NUS passage retrieval module reinforced by fuzzy relation matching and query expansion. Both question typing and NE extraction modules are rule-based, as employed in a TREC QA system [5]. We return the first top-ranked sentence that contains the expected named entity as the answer passage. The average length of the returned passages is 181 bytes.

We evaluate the QA system in the context of the QA passage task of TREC-12 [20]. Our system answers 175 questions correctly out of the total 324 questions, resulting in an accuracy of 0.540. When averaging over all 383 questions that do not have NIL answers, the accuracy is 0.457, which is still better than the second ranked system in the official TREC evaluations [20].

6. Conclusions

In this paper, we have presented a novel fuzzy relation matching technique for factoid QA passage retrieval. Our evaluation results show that our technique produces significant improvements in retrieval performance in current systems: a vast 50~138%

improvement in MRR, and over 95% in precision at top one passage. Fuzzy matching of dependency relations is calculated based on the degree of match between relation paths in candidate sentences and the question. For learning a model of relationship matching from training data, we have presented two methods based on mutual information and iterative EM. While these two methods do not make an obvious difference given our test data, we believe that EM scales better and may improve when given a larger amount of training data. Furthermore, our relation matching technique has shown itself capable of bringing significant improvement in retrieval performance across all the architectures we have tested, regardless of whether or not query expansion is used. As such, we recommend that future passage retrieval systems should incorporate approximate relation matching to achieve state-of-the-art performance.

Past work has shown that strict matching does not perform well in answer extraction. We have shown that this conclusion does not generalize to all QA modules. A contribution of this paper is the demonstration that even strict matching of relations significantly augments the performance of current passage retrieval modules. This may be explained by the fact that passage retrieval imposes less constraint in matching relations than answer extraction. Future work is expected to improve answer extraction by using relations effectively.

Our empirical evaluation results and qualitative error analysis reveal that the relation matching method can be improved by better alignment of relation paths. Relation paths often cannot be paired due to few matched question terms or paraphrasing, both of which could be alleviated by query expansion. While we have benchmarked the performance of relation matching with query expansion, our experiment has not fully integrated the modules in the sense that we have not taken advantage of expanded terms in relation matching. Seamless integration of query expansion with relation matching is likely to produce further gains in performances and is a logical next step in future research.

7. ACKNOWLEDGMENTS

The authors are grateful to Jimmy Lin for his excellent suggestions on the draft of the paper. We also thank Alexia Leong for proofreading the paper. Thanks also go to anonymous reviewers whose valuable comments have helped improve this paper. The first author is supported by the Singapore Millennium Foundation Scholarship in his PhD studies.

8. REFERENCES

[1] Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, D. Melamed, F. Och, D. Purdy, N. Smith, and D. Yarowsky, *Statistical machine translation,* Final Report, JHU Summer Workshop, 1999.

[2] G. Attardi, A. Cisternino, F. Formica, M. Simi and A. Tommasi, *PiQASso: Pisa Question Answering System*, Proc. of TREC-2001, 2001, pp. 599-607.

[3] A. Berger and J. Lafferty, *Information retrieval as statistical translation*, Proc. of SIGIR '99, 1999, pp. 222-229.

[4] P. Brown, S. Della, V. Della Pietra and R. Mercer, *The mathematics of statistical machine translation: Parameter estimation*, Computational Linguistics, 19(2), 1993, pp. 263-311.

[5] 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.

[6] A. Echihabi and D. Marcu, *A noisy-channel approach for question answering,* Proc. of ACL '03, 2003.

[7] J. Gao, J.-Y. Nie, G. Wu and G. Cao, *Dependency language model for information retrieval,* Proc. of SIGIR '04, Sheffield, UK, 2004, pp. 170-177.

[8] S. Harabagiu, D. Moldovan, C. Clark, M. Bowden, J. Williams and J. Bensley, *Answer Mining by Combining Extraction Techniques with Abductive Reasoning*, Proc. of TREC-12, 2003, pp. 375-382.

[9] D. Hull, *Using statistical testing in the evaluation of retrieval experiments*, Proc. of SIGIR '93, 1993.

[10] A. Ittycheriah, M. Franz, and S. Roukos, *IBM's statistical question answering system - TREC-10*, Proc. of TREC-10, 2001.

[11] M. Kaszkeil and J. Zobel, *Passage retrieval revisited*, Proc. of SIGIR '97, Philadelphia, PA, USA, 1997, pp. 178-185.

[12] B. Katz and J. Lin, *Selectively Using Relations to Improve Precision in Question Answering*, Proc. of the EACL-2003 Workshop on Natural Language Processing for Question Answering, April 2003.

[13] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim, *SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP*, Proc. of TREC-10, 2001, pp. 442-451.

[14] M. Light, G. S. Mann, E. Riloff, and E. Breck, *Analyses for elucidating current question answering technology*, Journal of Natural Language Engineering, Special Issue on Question Answering, Fall–Winter, 2001.

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

[16] J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz and D. R. Karger, *What makes a good answer? The role of context in question answering*, Proc. of the ninth IFIP TC13 International Conference on Human-Computer Interaction, 2003.

[17] D. Lin and P. Pantel, *Discovery of Inference Rules for Question Answering,* Natural Language Engineering, 2001, 7(4): pp. 343-360.

[18] F. Song and B. Croft, *A general language model for information retrieval*, Proc. of CIKM'99, 1999, pp. 316-321.

[19] S.Tellex, B.Katz, J.Lin, A.Fernandes and G.Marton, *Quantitative evaluation of passage retrieval algorithms for question answering*, Proc. of SIGIR '03, 2003, Toronto, Canada, pp. 41-47.

[20] E.M. Voorhees, Overview of the TREC 2003 Question Answering Track, Proc. of TREC-12, pp. 54-68.