Exploring Correlation of Dependency Relation Paths for Answer Extraction

Dan Shen
Department of Computational Linguistics
Saarland University
Saarbruecken, Germany
dshen@coli.uni-sb.de

Dietrich Klakow
Spoken Language Systems
Saarland University
Saarbruecken, Germany
klakow@lsv.uni-saarland.de

Abstract

In this paper, we explore correlation of dependency relation paths to rank candidate answers in answer extraction. Using the correlation measure, we compare dependency relations of a candidate answer and mapped question phrases in sentence with the corresponding relations in question. Different from previous studies, we propose an approximate phrase mapping algorithm and incorporate the mapping score into the correlation measure. The correlations are further incorporated into a Maximum Entropy-based ranking model which estimates path weights from training. Experimental results show that our method significantly outperforms state-of-the-art syntactic relation-based methods by up to 20% in MRR.

1 Introduction

Answer Extraction is one of basic modules in open domain Question Answering (QA). It is to further process relevant sentences extracted with Passage / Sentence Retrieval and pinpoint exact answers using more linguistic-motivated analysis. Since QA turns to find exact answers rather than text snippets in recent years, answer extraction becomes more and more crucial.

Typically, answer extraction works in the following steps:

- Recognize expected answer type of a question.
- Annotate relevant sentences with various types of named entities.
- Regard the phrases annotated with the expected answer type as candidate answers.
- Rank candidate answers.

In the above work flow, answer extraction heavily relies on named entity recognition (NER). On one hand, NER reduces the number of candidate answers and eases answer ranking. On the other hand, the errors from NER directly degrade answer extraction performance. To our knowledge, most top ranked QA systems in TREC are supported by effective NER modules which may identify and classify more than 20 types of named entities (NE), such as *abbreviation*, *music*, *movie*, etc. However, developing such named entity recognizer is not trivial. Up to now, we haven't found any paper relevant to QA-specific NER development. So, it is hard to follow their work. In this paper, we just use a general MUC-based NER, which makes our results reproducible.

A general MUC-based NER can't annotate a large number of NE classes. In this case, all noun phrases in sentences are regarded as candidate answers, which makes candidate answer sets much larger than those filtered by a well developed NER. The larger candidate answer sets result in the more difficult answer extraction. Previous methods working on surface word level, such as density-based ranking and pattern matching, may not perform well. Deeper linguistic analysis has to be conducted. This paper proposes a statistical method which exploring correlation of dependency relation paths to rank candidate answers. It is motivated by the observation that relations between proper answers and question phrases in candidate sentences are always similar to the corresponding relations in question. For example, the question "*What did Alfred Nobel invent?*" and the

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 889–896,
Sydney, July 2006.

candidate sentence "*... in the will of Swedish industrialist Alfred Nobel, who invented* ***dynamite***."

For each question, firstly, dependency relation paths are defined and extracted from the question and each of its candidate sentences. Secondly, the paths from the question and the candidate sentence are paired according to question phrase mapping score. Thirdly, correlation between two paths of each pair is calculated by employing Dynamic Time Warping algorithm. The input of the calculation is correlations between dependency relations, which are estimated from a set of training path pairs. Lastly, a Maximum Entropy-based ranking model is proposed to incorporate the path correlations and rank candidate answers. Furthermore, sentence supportive measure are presented according to correlations of relation paths among question phrases. It is applied to re-rank the candidate answers extracted from the different candidate sentences. Considering phrases may provide more accurate information than individual words, we extract dependency relations on phrase level instead of word level.

The experiment on TREC questions shows that our method significantly outperforms a density-based method by 50% in MRR and three state-of-the-art syntactic-based methods by up to 20% in MRR. Furthermore, we classify questions by judging whether NER is used. We investigate how these methods perform on the two question sets. The results indicate that our method achieves better performance than the other syntactic-based methods on both question sets. Especially for more difficult questions, for which NER may not help, our method improves MRR by up to 31%.

The paper is organized as follows. Section 2 discusses related work and clarifies what is new in this paper. Section 3 presents relation path correlation in detail. Section 4 and 5 discuss how to incorporate the correlations for answer ranking and re-ranking. Section 6 reports experiment and results.

2 Related Work

In recent years' TREC Evaluation, most top ranked QA systems use syntactic information in answer extraction. Next, we will briefly discuss the main usages.

(Kaisser and Becker, 2004) match a question into one of predefined patterns, such as "*When did Jack Welch retire from GE?*" to the pattern "*When+did+NP+Verb+NPorPP*". For each question pattern, there is a set of syntactic structures for potential answer. Candidate answers are ranked by matching the syntactic structures. This method worked well on TREC questions. However, it is costing to manually construct question patterns and syntactic structures of the patterns.

(Shen et al., 2005) classify question words into four classes *target word*, *head word*, *subject word* and *verb*. For each class, syntactic relation patterns which contain one question word and one proper answer are automatically extracted and scored from training sentences. Then, candidate answers are ranked by partial matching to the syntactic relation patterns using tree kernel. However, the criterion to classify the question words is not clear in their paper. Proper answers may have absolutely different relations with different *subject words* in sentences. They don't consider the corresponding relations in questions.

(Tanev et al., 2004; Wu et al., 2005) compare syntactic relations in questions and those in answer sentences. (Tanev et al., 2004) reconstruct a basic syntactic template tree for a question, in which one of the nodes denotes expected answer position. Then, answer candidates for this question are ranked by matching sentence syntactic tree to the question template tree. Furthermore, the matching is weighted by lexical variations. (Wu et al., 2005) combine n-gram proximity search and syntactic relation matching. For syntactic relation matching, question tree and sentence subtree around a candidate answer are matched from node to node.

Although the above systems apply the different methods to compare relations in question and answer sentences, they follow the same hypothesis that proper answers are more likely to have same relations in question and answer sentences. For example, in question "*Who founded the Black Panthers organization?*", where, the question word "*who*" has the dependency relations "*subj*" with "*found*" and "*subj obj nn*" with "*Black Panthers organization*", in sentence "*Hilliard introduced* ***Bobby Seale****, who co-founded the Black Panther Party here ...*", the proper answer "*Bobby Seale*" has the same relations with most question phrases. These methods achieve high precision, but poor recall due to relation variations. One meaning is often represented as different relation combinations. In the above example, appositive rela-

tion frequently appears in answer sentences, such as "*Black Panther Party co-founder **Bobby Seale** is ordered bound and gagged ...*" and indicates proper answer *Bobby Seale* although it is asked in different way in the question.

(Cui et al., 2004) propose an approximate dependency relation matching method for both passage retrieval and answer extraction. The similarity between two relations is measured by their co-occurrence rather than exact matching. They state that their method effectively overcomes the limitation of the previous exact matching methods. Lastly, they use the sum of similarities of all path pairs to rank candidate answers, which is based on the assumption that all paths have equal weights. However, it might not be true. For example, in question "*What book did Rachel Carson write in 1962?*", the phrase "*Rachel Carson*" looks like more important than "*1962*" since the former is question topic and the latter is a constraint for expected answer. In addition, lexical variations are not well considered and a weak relation path alignment algorithm is used in their work.

Based on the previous works, this paper explores correlation of dependency relation paths between questions and candidate sentences. Dynamic time warping algorithm is adapted to calculate path correlations and approximate phrase mapping is proposed to cope with phrase variations. Finally, maximum entropy-based ranking model is developed to incorporate the correlations and rank candidate answers.

3 Dependency Relation Path Correlation

In this section, we discuss how the method performs in detail.

3.1 Dependency Relation Path Extraction

We parse questions and candidate sentences with MiniPar (Lin, 1994), a fast and robust parser for grammatical dependency relations. Then, we extract relation paths from dependency trees.

Dependency relation path is defined as a structure $P =< N_1, R, N_2 >$ where, N_1, N_2 are two phrases and R is a relation sequence $R =< r_1, ..., r_i >$ in which r_i is one of the predefined dependency relations. Totally, there are 42 relations defined in MiniPar. A relation sequence R between two phrases N_1, N_2 is extracted by traversing from the N_1 node to the N_2 node in a dependency tree.

Q: What book did Rachel Carson write in 1962?

Paths for Answer Ranking

N1 (EAP)	R	N2
What	det	book
What	det obj subj	Rachel Carson
What	det obj	write
What	det obj mod pcomp-n	1962

Paths for Answer Re-ranking

book	obj subj	Rachel Carson
book	obj	write
book	obj mod pcomp-n	1962

...

S: Rachel Carson 's 1962 book " Silent Spring " said dieldrin causes mania.

Paths for Answer Ranking

N1 (CA)	R	N2
Silent Spring	title	book
Silent Spring	title gen	Rachel Carson
Silent Spring	title num	1962

Paths for Answer Re-ranking

book	gen	Rachel Carson
book	num	1962

...

Figure 1: Relation Paths for sample question and sentence. *EAP* indicates expected answer position; *CA* indicates candidate answer

For each question, we extract relation paths among noun phrases, main verb and question word. The question word is further replaced with "*EAP*", which indicates the expected answer position. For each candidate sentence, we firstly extract relation paths between answer candidates and mapped question phrases. These paths will be used for answer ranking (Section 4). Secondly, we extract relation paths among mapped question phrases. These paths will be used for answer reranking (Section 5). Question phrase mapping will be discussed in Section 3.4. Figure 1 shows some relation paths extracted for an example question and candidate sentence.

Next, the relation paths in a question and each of its candidate sentences are paired according to their phrase similarity. For any two relation path P_i and P_j which are extracted from the question and the candidate sentence respectively, if $Sim(N_{i1}, N_{j1}) > 0$ and $Sim(N_{i2}, N_{j2}) > 0$, P_i and P_j are paired as $< P_i, P_j >$. The question phrase "*EAP*" is mapped to candidate answer phrase in the sentence. The similarity between two

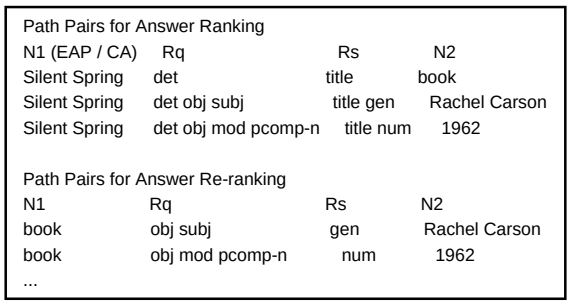

Figure 2: Paired Relation Path

phrases will be discussed in Section 3.4. Figure 2 further shows the paired relation paths which are presented in Figure 1.

3.2 Dependency Relation Path Correlation

Comparing a proper answer and other wrong candidate answers in each sentence, we assume that relation paths between the proper answer and question phrases in the sentence are more correlated to the corresponding paths in question. So, for each path pair $< P_1, P_2 >$, we measure the correlation between its two paths P_1 and P_2.

We derive the correlations between paths by adapting dynamic time warping (DTW) algorithm (Rabiner et al., 1978). DTW is to find an optimal alignment between two sequences which maximizes the accumulated correlation between two sequences. A sketch of the adapted algorithm is as follows.

Let $R_1 =< r_{11}, ..., r_{1n} >, (n = 1, ..., N)$ and $R_2 =< r_{21}, ..., r_{2m} >, (m = 1, ..., M)$ denote two relation sequences. R_1 and R_2 consist of N and M relations respectively. $R_1(n) = r_{1n}$ and $R_2(m) = r_{2m}$. $Cor(r_1, r_2)$ denotes the correlation between two individual relations r_1, r_2, which is estimated by a statistical model during training (Section 3.3). Given the correlations $Cor(r_{1n}, r_{2m})$ for each pair of relations (r_{1n}, r_{2m}) within R_1 and R_2, the goal of DTW is to find a path, $m = map(n)$, which map n onto the corresponding m such that the accumulated correlation Cor^* along the path is maximized.

$$Cor^* = \max_{map(n)} \left\{ \sum_{n=1}^{N} Cor(R_1(n), R_2(map(n)) \right\}$$

A dynamic programming method is used to determine the optimum path $map(n)$. The accumulated correlation Cor_A to any grid point (n, m) can be recursively calculated as

$$Cor_A(n, m) = Cor(r_{1n}, r_{2m}) + \max_{q \le m} Cor_A(n-1, q)$$
$$Cor^* = Cor_A(N, M)$$

The overall correlation measure has to be normalized as longer sequences normally give higher correlation value. So, the correlation between two sequences R_1 and R_2 is calculated as

$$Cor(R_1, R_2) = Cor^* / \max(N, M)$$

Finally, we define the correlation between two relation paths P_1 and P_2 as

$$Cor(P_1, P_2) = Cor(R_1, R_2) \times Sim(N_{11}, N_{21}) \times Sim(N_{12}, N_{22})$$

Where, $Sim(N_{11}, N_{21})$ and $Sim(N_{12}, N_{22})$ are the phrase mapping score when pairing two paths, which will be described in Section 3.4. If two phrases are absolutely different $Cor(N_{11}, N_{21}) = 0$ or $Cor(N_{12}, N_{22}) = 0$, the paths may not be paired since $Cor(P_1, P_2) = 0$.

3.3 Relation Correlation Estimation

In the above section, we have described how to measure path correlations. The measure requires relation correlations $Cor(r_1, r_2)$ as inputs. We apply a statistical method to estimate the relation correlations from a set of training path pairs. The training data collecting will be described in Section 6.1.

For each question and its answer sentences in training data, we extract relation paths between "*EAP*" and other phrases in the question and paths between proper answer and mapped question phrases in the sentences. After pairing the question paths and the corresponding sentence paths, correlation of two relations is measured by their bipartite co-occurrence in all training path pairs. Mutual information-based measure (Cui et al., 2004) is employed to calculate the relation correlations.

$$Cor(r_i^Q, r_j^S) = log \frac{\sum \alpha \times \delta(r_i^Q, r_j^S)}{f_Q(r_i^Q) \times f_S(r_j^S)}$$

where, r_i^Q and r_j^S are two relations in question paths and sentence paths respectively. $f_Q(r_i^Q)$ and $f_S(r_j^S)$ are the numbers of occurrences of r_i^Q in question paths and r_j^S in sentence paths respectively. $\delta(r_i^Q, r_j^S)$ is 1 when r_i^Q and r_j^S co-occur in a path pair, and 0 otherwise. α is a factor to discount the co-occurrence value for long paths. It is set to the inverse proportion of the sum of path lengths of the path pair.

3.4 Approximate Question Phrase Mapping

Basic noun phrases (BNP) and verbs in questions are mapped to their candidate sentences. A BNP is defined as the smallest noun phrase in which there are no noun phrases embedded. To address lexical and format variations between phrases, we propose an approximate phrase mapping strategy.

A BNP is separated into a set of heads $H = \{h_1, ..., h_i\}$ and a set of modifiers $M = \{m_1, ...m_j\}$. Some heuristic rules are applied to judge heads and modifiers: 1. If BNP is a named entity, all words are heads. 2. The last word of BNP is head. 3. Rest words are modifiers.

The similarity between two BNPs $Sim(BNP_q, BNP_s)$ is defined as:

$$Sim(BNP_q, BNP_s) = \lambda Sim(H_q, H_s) + (1-\lambda) Sim(M_q, M_s)$$

$$Sim(H_q, H_s) = \frac{\sum_{h_i \in H_q} \sum_{h_j \in H_s} Sim(h_i, h_j)}{|H_q \bigcup H_s|}$$

$$Sim(M_q, M_s) = \frac{\sum_{m_i \in M_q} \sum_{m_j \in M_s} Sim(m_i, m_j)}{|M_q \bigcup M_s|}$$

Furthermore, the similarity between two heads $Sim(h_i, h_j)$ are defined as:

- $Sim = 1$, if $h_i = h_j$ after stemming;
- $Sim = 1$, if $h_i = h_j$ after format alternation;
- $Sim = SemSim(h_i, h_j)$

These items consider morphological, format and semantic variations respectively. 1. The morphological variations match words after stemming, such as "*Rhodes scholars*" and "*Rhodes scholarships*". 2. The format alternations cope with special characters, such as "-" for "*Ice-T*" and "*Ice T*", "*&*" for "*Abercrombie and Fitch*" and "*Abercrombie & Fitch*". 3. The semantic similarity $SemSim(h_i, h_j)$ is measured using WordNet and eXtended WordNet. We use the same semantic path finding algorithm, relation weights and semantic similarity measure as (Moldovan and Novischi, 2002). For efficiency, only hypernym, hyponym and entailment relations are considered and search depth is set to 2 in our experiments.

Particularly, the semantic variations are not considered for NE heads and modifiers. Modifier similarity $Sim(m_i, m_j)$ only consider the morphological and format variations. Moreover, verb similarity measure $Sim(v_1, v_2)$ is the same as head similarity measure $Sim(h_i, h_j)$.

4 Candidate Answer Ranking

According to path correlations of candidate answers, a Maximum Entropy (ME)-based model is applied to rank candidate answers. Unlike (Cui et al., 2004), who rank candidate answers with the sum of the path correlations, ME model may estimate the optimal weights of the paths based on a training data set. (Berger et al., 1996) gave a good description of ME model. The model we use is similar to (Shen et al., 2005; Ravichandran et al., 2003), which regard answer extraction as a ranking problem instead of a classification problem. We apply Generalized Iterative Scaling for model parameter estimation and Gaussian Prior for smoothing.

If expected answer type is unknown during question processing or corresponding type of named entities isn't recognized in candidate sentences, we regard all basic noun phrases as candidate answers. Since a MUC-based NER loses many types of named entities, we have to handle larger candidate answer sets. Orthographic features, similar to (Shen et al., 2005), are extracted to capture word format information of candidate answers, such as capitalizations, digits and lengths, etc. We expect they may help to judge what proper answers look like since most NER systems work on these features.

Next, we will discuss how to incorporate path correlations. Two facts are considered to affect path weights: **question phrase type** and **path length**. For each question, we divide question phrases into four types: target, topic, constraint and verb. **Target** is a kind of word which indicates the expected answer type of the question, such as "*party*" in "*What party led Australia from 1983 to 1996?*". **Topic** is the event/person that the question talks about, such as "*Australia*". Intuitively, it is the most important phrase of the question. **Constraint** are the other phrases of the question except topic, such as "*1983*" and "*1996*". **Verb** is the main verb of the question, such as "*lead*". Furthermore, since shorter path indicates closer relation between two phrases, we discount path correlation in long question path by dividing the correlation by the length of the question path. Lastly, we sum the discounted path correlations for each type of question phrases and fire it as a feature, such as "*Target_Cor=c*, where *c* is the correlation value for question *target*. ME-based ranking model incorporate the orthographic and path

correlation features to rank candidate answers for each of candidate sentences.

5 Candidate Answer Re-ranking

After ranking candidate answers, we select the highest ranked one from each candidate sentence. In this section, we are to re-rank them according to sentence supportive degree. We assume that a candidate sentence supports an answer if relations between mapped question phrases in the candidate sentence are similar to the corresponding ones in question. Relation paths between any two question phrases are extracted and paired. Then, correlation of each pair is calculated. Re-rank formula is defined as follows:

$$Score(answer) = \alpha \times \sum_{i} Cor(P_{i1}, P_{i2})$$

where, α is answer ranking score. It is the normalized prediction value of the ME-based ranking model described in Section 4. $\sum_{i} Cor(P_{i1}, P_{i2})$ is the sum of correlations of all path pairs. Finally, the answer with the highest score is returned.

6 Experiments

In this section, we set up experiments on TREC factoid questions and report evaluation results.

6.1 Experiment Setup

The goal of answer extraction is to identify exact answers from given candidate sentence collections for questions. The candidate sentences are regarded as the most relevant sentences to the questions and retrieved by IR techniques. Qualities of the candidate sentences have a strong impact on answer extraction. It is meaningless to evaluate the questions of which none candidate sentences contain proper answer in answer extraction experiment. To our knowledge, most of current QA systems lose about half of questions in sentence retrieval stage. To make more questions evaluated in our experiments, for each of questions, we automatically build a candidate sentence set from TREC judgements rather than use sentence retrieval output.

We use TREC99-03 questions for training and TREC04 questions for testing. As to build training data, we retrieve all of the sentences which contain proper answers from relevant documents according to TREC judgements and answer patterns. Then, We manually check the sentences and remove those in which answers cannot be supported. As to build candidate sentence sets for testing, we retrieve all of the sentences from relevant documents in judgements and keep those which contain at least one question key word. Therefore, each question has at least one proper candidate sentence which contains proper answer in its candidate sentence set.

There are 230 factoid questions (27 NIL questions) in TREC04. NIL questions are excluded from our test set because TREC doesn't supply relevant documents and answer patterns for them. Therefore, we will evaluate 203 TREC04 questions. Five answer extraction methods are evaluated for comparison:

- **Density**: Density-based method is used as baseline, in which we choose candidate answer with the shortest surface distance to question phrases.
- **SynPattern**: Syntactic relation patterns (Shen et al., 2005) are automatically extracted from training set and are partially matched using tree kernel.
- **StrictMatch**: Strict relation matching follows the assumption in (Tanev et al., 2004; Wu et al., 2005). We implement it by adapting relation correlation score. In stead of learning relation correlations during training, we predefine them as: $Cor(r_1, r_2) = 1$ if $r_1 = r_2$; 0, otherwise.
- **ApprMatch**: Approximate relation matching (Cui et al., 2004) aligns two relation paths using fuzzy matching and ranks candidates according to the sum of all path similarities.
- **CorME**: It is the method proposed in this paper. Different from *ApprMatch*, ME-based ranking model is implemented to incorporate path correlations which assigns different weights for different paths respectively. Furthermore, phrase mapping score is incorporated into the path correlation measure.

These methods are briefly described in Section 2. Performance is evaluated with Mean Reciprocal Rank (MRR). Furthermore, we list percentages of questions correctly answered in terms of top 5 answers and top 1 answer returned respectively. No answer validations are used to adjust answers.

Table 1: Overall performance

	Density	SynPattern	StrictMatch	ApprMatch	CorME
MRR	0.45	0.56	0.57	0.60	0.67
Top1	0.36	0.53	0.49	0.53	0.62
Top5	0.56	0.60	0.67	0.70	0.74

6.2 Results

Table 1 shows the overall performance of the five methods. The main observations from the table are as follows:

1. The methods *SynPattern*, *StrictMatch*, *ApprMatch* and *CorME* significantly improve MRR by 25.0%, 26.8%, 34.5% and 50.1% over the baseline method *Density*. The improvements may benefit from the various explorations of syntactic relations.

2. The performance of *SynPattern* (0.56MRR) and *StrictMatch* (0.57MRR) are close. *SynPattern* matches relation sequences of candidate answers with the predefined relation sequences extracted from a training data set, while *StrictMatch* matches relation sequences of candidate answers with the corresponding relation sequences in questions. But, both of them are based on the assumption that the more number of same relations between two sequences, the more similar the sequences are. Furthermore, since most TREC04 questions only have one or two phrases and many questions have similar expressions, *SynPattern* and *StrictMatch* don't make essential difference.

3. *ApprMatch* and *CorME* outperform *SynPattern* and *StrictMatch* by about 6.1% and 18.4% improvement in MRR. Strict matching often fails due to various relation representations in syntactic trees. However, such variations of syntactic relations may be captured by *ApprMatch* and *CorME* using a MI-based statistical method.

4. *CorME* achieves the better performance by 11.6% than *ApprMatch*. The improvement may benefit from two aspects: 1) *ApprMatch* assigns equal weights to the paths of a candidate answer and question phrases, while *CorME* estimate the weights according to phrase type and path length. After training a ME model, the weights are assigned, such as 5.72 for *topic* path ; 3.44 for *constraints* path and 1.76 for *target* path. 2) *CorME* incorporates approximate phrase mapping scores into path correlation measure.

We further divide the questions into two classes according to whether NER is used in answer extraction. If the expected answer type of a question is unknown, such as "*How did James Dean die?*" or the type cannot be annotated by NER, such as "*What ethnic group/race are Crip members?*", we put the question in ***Qw/oNE*** set, otherwise, we put it in ***QwNE***. For the questions in *Qw/oNE*, we extract all basic noun phrases and verb phrases as candidate answers. Then, answer extraction module has to work on the larger candidate sets. Using a MUC-based NER, the recognized types include person, location, organization, date, time and money. In TREC04 questions, 123 questions are put in *QwNE* and 80 questions in *Qw/oNE*.

Table 2: Performance on two question sets *QwNE* and *Qw/oNE*

	QwNE	Qw/oNE
Density	0.66	0.11
SynPattern	0.71	0.36
StrictMatch	0.70	0.36
ApprMatch	0.72	0.42
CorME	0.79	0.47

We evaluate the performance on *QwNE* and *Qw/oNE* respectively, as shown in Table 2. The density-based method *Density* (0.11MRR) loses many questions in *Qw/oNE*, which indicates that using only surface word information is not sufficient for large candidate answer sets. On the contrary, *SynPattern*(0.36MRR), *StrictPattern*(0.36MRR), *ApprMatch*(0.42MRR) and *CorME* (0.47MRR) which capture syntactic information, perform much better than *Density*. Our method *CorME* outperforms the other syntactic-based methods on both *QwNE* and *Qw/oNE*. Es-

pecially for more difficult questions *Qw/oNE*, the improvements (up to 31% in MRR) are more obvious. It indicates that our method can be used to further enhance state-of-the-art QA systems even if they have a good NER.

In addition, we evaluate component contributions of our method based on the main idea of relation path correlation. Three components are tested: 1. **Appr. Mapping** (Section 3.4). We replace approximate question phrase mapping with exact phrase mapping and withdraw the phrase mapping scores from path correlation measure. 2. **Answer Ranking** (Section 4). Instead of using ME model, we sum all of the path correlations to rank candidate answers, which is similar to (Cui et al., 2004). 3. **Answer Re-ranking** (Section 5). We disable this component and select top 5 answers according to answer ranking scores.

Table 3: Component Contributions

	MRR
Overall	0.67
- Appr. Mapping	0.63
- Answer Ranking	0.62
- Answer Re-ranking	0.66

The contribution of each component is evaluated with the overall performance degradation after it is removed or replaced. Some findings are concluded from Table 3. Performances degrade when replacing approximate phrase mapping or ME-based answer ranking, which indicates that both of them have positive effects on the systems. This may be also used to explain why *CorME* outperforms *ApprMatch* in Table 1. However, removing answer re-ranking doesn't affect much. Since short questions, such as "*What does AARP stand for?*", frequently occur in TREC04, exploring the phrase relations for such questions isn't helpful.

7 Conclusion

In this paper, we propose a relation path correlation-based method to rank candidate answers in answer extraction. We extract and pair relation paths from questions and candidate sentences. Next, we measure the relation path correlation in each pair based on approximate phrase mapping score and relation sequence alignment, which is calculated by DTW algorithm. Lastly, a ME-based ranking model is proposed to incorporate the path correlations and rank candidate answers. The experiment on TREC questions shows that our method significantly outperforms a density-based method by 50% in MRR and three state-of-the-art syntactic-based methods by up to 20% in MRR. Furthermore, the method is especially effective for difficult questions, for which NER may not help. Therefore, it may be used to further enhance state-of-the-art QA systems even if they have a good NER. In the future, we are to further evaluate the method based on the overall performance of a QA system and adapt it to sentence retrieval task.

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguisitics*, 22:39–71.

Hang Cui, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen Kan. 2004. National university of singapore at the trec-13 question answering. In *Proceedings of TREC2004, NIST*.

M. Kaisser and T. Becker. 2004. Question answering by searching large corpora with linguistic methods. In *Proceedings of TREC2004, NIST*.

Dekang Lin. 1994. Principar—an efficient, broad-coverage, principle-based parser. In *Proceedings of COLING1994*, pages 42–488.

Dan Moldovan and Adrian Novischi. 2002. Lexical chains for question answering. In *Proceedings of COLING2002*.

L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson. 1978. Considerations in dynamic time warping algorithms for discrete word recognition. In *Proceedings of IEEE Transactions on acoustics, speech and signal processing*.

Deepak Ravichandran, Eduard Hovy, and Franz Josef Och. 2003. Statistical qa - classifier vs. re-ranker: What's the difference? In *Proceedings of ACL2003 workshop on Multilingual Summarization and Question Answering*.

Dan Shen, Geert-Jan M. Kruijff, and Dietrich Klakow. 2005. Exploring syntactic relation patterns for question answering. In *Proceedings of IJCNLP2005*.

H. Tanev, M. Kouylekov, and B. Magnini. 2004. Combining linguisitic processing and web mining for question answering: Itc-irst at trec-2004. In *Proceedings of TREC2004, NIST*.

M. Wu, M. Y. Duan, S. Shaikh, S. Small, and T. Strzalkowski. 2005. University at albany's ilqua in trec 2005. In *Proceedings of TREC2005, NIST*.