Exploring Syntactic Relation Patterns for Question Answering

Dan Shen[1,2], Geert-Jan M. Kruijff [1], and Dietrich Klakow[2]

[1] Department of Computational Linguistics, Saarland University, Building 17, Postfach 15 11 50, 66041 Saarbruecken, Germany
{dshen, gj}@coli.uni-sb.de
[2] Lehrstuhl Sprach Signal Verarbeitung,Saarland University, Building 17, Postfach 15 11 50, 66041 Saarbruecken, Germany
{dietrich.klakow}@lsv.uni-saarland.de

Abstract. In this paper, we explore the syntactic relation patterns for open-domain factoid question answering. We propose a pattern extraction method to extract the various relations between the proper answers and different types of question words, including target words, head words, subject words and verbs, from syntactic trees. We further propose a QA-specific tree kernel to partially match the syntactic relation patterns. It makes the more tolerant matching between two patterns and helps to solve the data sparseness problem. Lastly, we incorporate the patterns into a Maximum Entropy Model to rank the answer candidates. The experiment on TREC questions shows that the syntactic relation patterns help to improve the performance by 6.91 MRR based on the common features.

1 Introduction

Question answering is to find answers for open-domain natural language questions in a large document collection. A typical QA system usually consists of three basic modules: 1. Question Processing (QP) Module, which finds some useful information from questions, such as expected answer type and key words; 2. Information Retrieval (IR) Module, which searches a document collection to retrieve a set of relevant sentences using the key words; 3. Answer Extraction (AE) Module, which analyzes the relevant sentences using the information provided by the QP module and identify the proper answer. In this paper, we will focus on the AE module.

In order to find the answers, some evidences, such as expected answer types and surface text patterns, are extracted from answer sentences and incorporated in the AE module using a pipelined structure, a scoring function or some statistical-based methods. However, the evidences extracted from plain texts are not sufficient to identify a proper answer. For examples, for *"Q1910: What are pennies made of?*", the expected answer type is unknown; for "*Q21: Who was the first American in space?*", the surface patterns may not detect the long-distance relations between the question key phrase "*the first American in space*" and the answer "*Alan Shepard*" in "*… that carried Alan Shepard on a 15 - minute suborbital flight in 1961 , making him the first*

R. Dale et al. (Eds.): IJCNLP 2005, LNAI 3651, pp. 507 – 518, 2005.

American in space." To solve these problems, more evidences need to be extracted from the more complex data representations, such as parse trees.

In this paper, we explore the syntactic relation patterns (SRP) for the AE module. An SRP is defined as a kind of relation between a question word and an answer candidate in the syntactic tree. Different from the textual patterns, the SRPs capture the relations based on the sentence syntactic structure rather than the sentence surface. Therefore, they may get the deeper understanding of the relations and capture the long range dependency between words regardless of their ordering and distance in the surface text. Based on the observation of the task, we find that the syntactic relations between different types of question words and answers vary a lot with each other. We classify the question words into four classes, including target words, head words, subject phrases and verbs, and generate the SRPs for them respectively. Firstly, we generate the SRPs from the training data and score them based on the support and confidence measures. Next, we propose a QA-specific tree kernel to calculate the similarity between two SRPs in order to match the patterns from the unseen data into the pattern set. The tree kernel makes the partial matching between two patterns and helps to solve the data sparseness problem. Lastly, we incorporate the SRPs into a Maximum Entropy Model along with some common features to classify the answer candidates. The experiment on TREC questions shows that the syntactic relation patterns improve the performance by 6.91 MRR based on the common features.

Although several syntactic relations, such as subject-verb and verb-object, have been also considered in some other systems, they are basically extracted using a small number of hand-built rules. As a result, they are limited and costly. In our task, we automatically extract the various relations between different question words and answers and more tolerantly match the relation patterns using the tree kernel.

2 Related Work

The relations between answers and question words have been explored by many successful QA systems based on certain sentence representations, such as word sequence, logic form, parse tree, etc.

In the simplest case, a sentence is represented as a sequence of words. It is assumed that, for certain type of questions, the proper answers always have certain surface relations with the question words. For example, "*Q: When was X born?*", the proper answers often have such relation "<*X*> *(* <*Answer*>--" with the question phrase *X* . [14] first used a predefined pattern set in QA and achieved a good performance at TREC10. [13] further developed a bootstrapping method to learn the surface patterns automatically. When testing, most of them make the partial matching using regular expression. However, such surface patterns strongly depend on the word ordering and distance in the text and are too specific to the question type.

LCC [9] explored the syntactic relations, such as subject, object, prepositional attachment and adjectival/adverbial adjuncts, based on the logic form transformation. Furthermore they used a logic prover to justify the answer candidates. The prover is accurate but costly.

Most of the QA systems explored the syntactic relations on the parse tree. Since such relations do not depend on the word ordering and distance in the sentence, they may cope with the various surface expressions of the sentence. ISI [7] extracted the

relations, such as "subject-verb" and "verb-object", in the answer sentence tree and compared with those in the question tree. IBM's Maximum Entropy-based model [10] integrated a rich feature set, including words co-occurrence scores, named entity, dependency relations, etc. For the dependency relations, they considered some predefined relations in trees by partial matching. BBN [15] also considered the verb-argument relations.

However, most of the current QA systems only focus on certain relation types, such as verb-argument relations, and extract them from the syntactic tree using some heuristic rules. Therefore, extracting such relations is limited in a very local context of the answer node, such as its parent or sibling nodes, and does not involve long range dependencies. Furthermore, most of the current systems only concern the relations to certain type of question words, such as verb. In fact, different types of question words may have different indicative relations with the proper answers. In this paper, we will automatically extract more comprehensive syntactic relation patterns for all types of question words, partially match them using a QA-specific tree kernel and evaluate their contributions by integrating them into a Maximum Entropy Model.

3 Syntactic Relation Pattern Generating

In this section, we will discuss how to extract the syntactic relation patterns. Firstly, we briefly introduce the question processing module which provides some necessary information to the answer extraction module. Secondly, we generate the dependency tree of the answer sentence and map the question words into the tree using a Modified Edit Distance (MED) algorithm. Thirdly, we define and extract the syntactic relation patterns in the mapped dependency tree. Lastly, we score and filter the patterns.

3.1 Question Processing Module

The key words are extracted from the questions. Considering that different key words may have different syntactic relations with the answers, we divide the key words into the following four types:

1. Target Words, which are extracted from *what / which* questions. Such words indicate the expected answer types, such as "*party*" in "*Q1967: What party led …?*".
2. Head Words, which are extracted from *how* questions. Such words indicate the expected answer heads, such as "*dog*" in the "*Q210: How many dogs pull …?*"
3. Subject Phrases, which are extracted from all types of questions. They are the base noun phrases of the questions except the target words and the head words.
4. Verbs, which are the main verbs extracted from non-definition questions.

The key words described above are identified and classified based on the question parse tree. We employ the Collins Parser [2] to parse the questions and the answer sentences.

3.2 Question Key Words Mapping

From this section, we start to introduce the AE module. Firstly, the answer sentences are tagged with named entities and parsed. Secondly, the parse trees are transformed

to the dependency trees based on a set of rules. To simplify a dependency tree, some special rules are used to remove the non-useful nodes and dependency information. The rules include

1. Since the question key words are always NPs and verbs, only the syntactic relations between NP and NP / NP and verb are considered.
2. The original form of Base Noun Phrase (BNP) is kept and the dependency relations within the BNPs are not considered, such as adjective-noun. A base noun phrase is defined as the smallest noun phrase in which there are no noun phrases embedded.

An example of the dependency tree is shown in Figure 1. We regard all BNP nodes and leaf nodes as answer candidates.

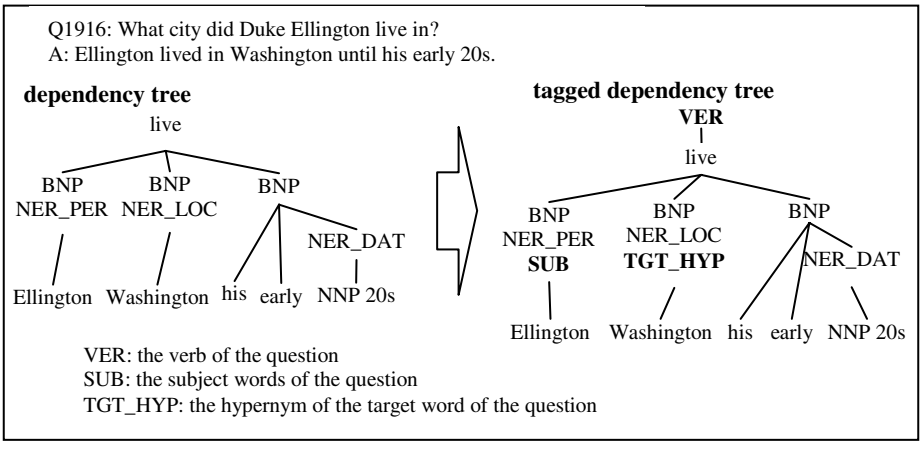

Fig. 1. Dependency tree and Tagged dependency tree

Next, we map the question key words into the simplified dependency trees. We propose a weighted edit distance (WED) algorithm, which is to find the similarity between two phrases by computing the minimal cost of operations needed to transform one phrase into the other, where an operation is an insertion, deletion, or substitution.

Different from the commonly-used edit distance algorithm [11], the WED defines the more flexible cost function which incorporates the morphological and semantic alternations of the words. The morphological alternations indicate the inflections of noun/verb. For example, for *Q2149: How many Olympic gold medals did Carl Lewis win?* We map the verb *win* to the nominal *winner* in the answer sentence "*Carl Lewis, winner of nine Olympic gold medals, thinks that …*". The morphological alternations are found based on a stemming algorithm and the "derivationally related forms" in WordNet [8]. The semantic alternations consider the synonyms of the words. Some types of the semantic relations in WordNet enable the retrieval of synonyms, such as hypernym, hyponym, etc. For example, for *Q212: Who invented the electric guitar?* We may map the verb *invent* to its direct hypernym *create* in answer sentences. Based on the observation of the task, we set the substitution costs of the alternations as follows: Identical words have cost 0; Words with the same morphological root have cost 0.2; Words with the hypernym or hyponym relations have cost

0.4; Words in the same SynSet have cost 0.6; Words with subsequence relations have cost 0.8; otherwise, words have cost 1. Figure 1 also shows an example of the tagged dependency tree.

3.3 Syntactic Relation Pattern Extraction

A syntactic relation pattern is defined as the smallest subtree which covers an answer candidate node and one question key word node in the dependency tree. To capture different relations between answer candidates and different types of question words, we generate four pattern sets, called *PSet_target*, *PSet_head*, *PSet_subject* and *PSet_verb*, for the answer candidates. The patterns are extracted from the training data. Some pattern examples are shown in Table 1. For a question *Q*, there are a set of relevant sentences *SentSet*. The extraction process is as follows:

1. for each question *Q* in the training data
2. question processing model extract the key words of Q
3. for each sentence *s* in *SentSet*
   a) parse *s*
   b) map the question key words into the parse tree
   c) tag all BNP nodes in the parse tree as answer candidates.
   d) for each answer candidate (*ac*) node
      for each question word (*qw*) node
         extract the syntactic relation pattern (*srp*) for *ac* and *qw*
         add *srp* to *PSet_target*, *PSet_head*, *PSet_subject* or *PSet_verb* based on the types of *qw*.

Table 1. Examples of the patterns in the four pattern sets

PatternSet	Patterns	Sup.	Conf.
PSet_target	(NPB~AC~TGT)	0.55	0.22
	(NPB~AC~null (NPB~null~TGT))	0.08	0.06
	(NPB~null~null (NPB~AC~null) (NPB~null~TGT))	0.02	0.09
PSet_head	(NPB~null~null (CD~AC~null) (NPB~null~HEAD))	0.59	0.67
PSet_subject	(VP~null~null (NPB~null~SUB) (NPB~null~null (NPB~AC~null)))	0.04	0.33
	(NPB~null~null (NPB~null~SUB) (NPB~AC~null))	0.02	0.18
PSet_verb	(VP~null~VERB (NPB~AC~null))	0.18	0.16

3.4 Syntactic Relation Pattern Scoring

The patterns extracted in section 3.3 are scored by support and confidence measures. Support and confidence measures are most commonly used to evaluate the association rules in the data mining area. The support of a rule is the proportion of times the rule applies. The confidence of a rule is the proportion of times the rule is correct. In our task, we score a pattern by measuring the strength of the association rule from the pattern to the proper answer (the pattern is matched => the answer is correct). Let p_i be any pattern in the pattern set *PSet* ,

$$\text{support}(p_i) = \frac{\text{the number of } p_i \text{ in which } ac \text{ is correct}}{\text{the size of } PSet}$$

$$\text{confidence}(p_i) = \frac{\text{the number of } p_i \text{ in which } ac \text{ is correct}}{\text{the number of } p_i}$$

We score the patterns in the *PSet_target*, *PSet_head, PSet_subject* and *PSet_verb* respectively. If the support value is less than the threshold t_{sup} or the confidence value is less than the threshold t_{conf}, the pattern is removed from the set. In the experiment, we set t_{sup} 0.01 and t_{conf} 0.5. Table 1 lists the support and confidence of the patterns.

4 Syntactic Relation Pattern Matching

Since we build the pattern sets based on the training data in the current experiment, the pattern sets may not be large enough to cover all of the unseen cases. If we make the exact match between two patterns, we will suffer from the data sparseness problem. So a partial matching method is required. In this section, we will propose a QA-specific tree kernel to match the patterns.

A kernel function $K(x_1, x_2) : \mathbf{X} \times \mathbf{X} \to [0, \boldsymbol{R}]$, is a similarity measure between two objects x_1 and x_2 with some constraints. It is the most important component of kernel methods [16]. Tree kernels are the structure-driven kernels used to calculate the similarity between two trees. They have been successfully accepted in the natural language processing applications, such as parsing [4], part of speech tagging and named entity extraction [3], and information extraction [5, 17]. To our knowledge, tree kernels have not been explored in answer extraction.

Suppose that a pattern is defined as a tree *T* with nodes $\{t_0, t_1, ..., t_n\}$ and each node t_i is attached with a set of attributes $\{a_0, a_1, ..., a_m\}$, which represent the local characteristics of t_i. In our task, the set of the attributes include Type attributes, Orthographic attributes and Relation Role attributes, as shown in Table 2. Figure 2 shows an example of the pattern tree *T_ac#target*.

The core idea of the tree kernel $K(T_1, T_2)$ is that the similarity between two trees T_1 and T_2 is the sum of the similarity between their subtrees. It can be calculated by dynamic programming and can capture the long-range relations between two nodes. The kernel we use is similar to [17] except that we define a task-specific matching function and similarity function, which are two primitive functions to calculate the similarity between two nodes in terms of their attributes.

Matching function $$m(t_i, t_j) = \begin{cases} 1 & \text{if } t_i.type = t_j.type \text{ and } t_i.role = t_j.role \\ 0 & \text{otherwise} \end{cases}$$

Similarity function $s(t_i, t_j) = \sum_{a \in \{a_0, \ldots, a_m\}} f(t_i.a, t_j.a)$

where, $f(t_i.a, t_j.a)$ is a compatibility function between two feature values

$$f(t_i.a, t_j.a) = \begin{cases} 1 & \text{if } t_i.a = t_j.a \\ 0 & \text{otherwise} \end{cases}$$

Table 2. Attributes of the nodes

Attributes		Examples
Type	POS tag	CD, NNP, NN…
	syntactic tag	NP, VP, …
Orthographic	Is Digit?	DIG, DIGALL
	Is Capitalized?	CAP, CAPALL
	length of phrase	LNG1, LNG2#3, LNGgt3
Role1	Is answer candidate?	true, false
Role2	Is question key words?	true, false

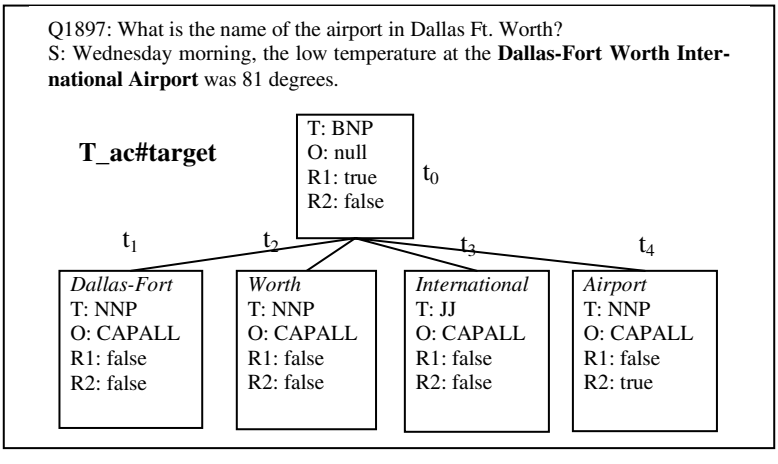

Fig. 2. An example of the pattern tree T_ac#target

5 ME-Based Answer Extraction

In addition to the syntactic relation patterns, many other evidences, such as named entity tags, may help to detect the proper answers. Therefore, we use maximum entropy to integrate the syntactic relation patterns and the common features.

5.1 Maximum Entropy Model

[1] gave a good description of the core idea of maximum entropy model. In our task, we use the maximum entropy model to rank the answer candidates for a question,

which is similar to [12]. Given a question q and a set of possible answer candidates $\{ac_1, ac_2 ... ac_n\}$, the model outputs the answer $ac \in \{ac_1, ac_2 ... ac_n\}$ with the maximal probability from the answer candidate set. We define M feature functions $f_m(ac, \{ac_1, ac_2 ... ac_n\}, q)$, m=1,...,M. The probability is modeled as

$$P(ac \mid \{ac_1, ac_2 ... ac_n\}, q) = \frac{\exp[\sum_{m=1}^{M} \lambda_m f_m(ac, \{ac_1, ac_2 ... ac_n\}, q))]}{\sum_{ac'} \exp[\sum_{m=1}^{M} \lambda_m f_m(ac', \{ac_1, ac_2 ... ac_n\}, q)]}$$

where, λ_m (m=1,...,M) are the model parameters, which are trained with Generalized Iterative Scaling [6]. A Gaussian Prior is used to smooth the ME model.

Table 3. Examples of the common features

Features	Examples	Explanation
NE	NE#DAT_QT_DAT	ac is NE (DATE) and qtarget is DATE
	NE#PER_QW_WHO	ac is NE (PERSON) and qword is WHO
Ortho-graphic	SSEQ_Q	ac is a subsequence of question
	CAP_QT_LOC	ac is capitalized and qtarget is LOCATION
	LNGlt3_QT_PER	the length of ac $\leq$ 3 and qtarget is PERSON
Syntactic Tag	CD_QT_NUM	syn. tag of ac is CD and qtarget is NUM
	NNP_QT_PER	syn. tag of ac is NNP and qtarget is PERSON
Triggers	TRG_HOW_DIST	ac matches the trigger words for HOW questions which ask for distance

5.2 Features

For the baseline maximum entropy model, we use four types of common features:

1. **Named Entity Features**: For certain question target, if the answer candidate is tagged as certain type of named entity, one feature fires.
2. **Orthographic Features**: They capture the surface format of the answer candidates, such as capitalizations, digits and lengths, etc.
3. **Syntactic Tag Features**: For certain question target, if the word in the answer candidate belongs to a certain syntactic / POS type, one feature fires.
4. **Triggers**: For some *how* questions, there are always some trigger words which are indicative for the answers. For example, for "*Q2156: How fast does Randy Johnson throw?*", the word "*mph*" may help to identify the answer "*98-mph*" in "*Johnson throws a 98-mph fastball.*"

Table 3 shows some examples of the common features. All of the features are the binary features. In addition, many other features, such as the answer candidate frequency, can be extracted based on the IR output and are thought as the indicative evidences for the answer extraction [10]. However, in this paper, we are to evaluate the answer extraction module independently, so we do not incorporate such features in the current model.

In order to evaluate the effectiveness of the automatically generated syntactic relation patterns, we also manually build some heuristic rules to extract the relation features from the trees and incorporate them into the baseline model. The baseline model uses 20 rules. Some examples of the **hand-extracted relation features** are listed as follows,

- If the ac node is the same of the qtarget node, one feature fires.
- If the ac node is the sibling of the qtarget node, one feature fires.
- If the ac node is the child of the qsubject node, one feature fires.
- …

Next, we will discuss the use of the **syntactic relation features**. Firstly, for each answer candidate, we extract the syntactic relations between it and all mapped question key words in the sentence tree. Then for each extracted relation, we match it in the pattern set *PSet_target*, *PSet_head*, *PSet_subject* or *PSet_verb*. A tree kernel discussed in Section 4 is used to calculate the similarity between two patterns. Finally, if the maximal similarity is above a threshold λ, the pattern with the maximal similarity is chosen and the corresponding feature fires. The experiments will evaluate the performance and the coverage of the pattern sets based on different λ values.

6 Experiment

We apply the AE module to the TREC QA task. Since this paper focuses on the AE module alone, we only present those sentences containing the proper answers to the AE module based on the assumption that the IR module has got 100% precision. The AE module is to identify the proper answers from the given sentence collection.

We use the questions of TREC8, 9, 2001 and 2002 for training and the questions of TREC2003 for testing. The following steps are used to generate the data:

1. Retrieve the relevant documents for each question based on the TREC judgments.
2. Extract the sentences, which match both the proper answer and at least one question key word, from these documents.
3. Tag the proper answer in the sentences based on the TREC answer patterns.

In TREC 2003, there are 413 factoid questions in which 51 questions (NIL questions) are not returned with the proper answers by TREC. According to our data generation process, we cannot provide data for those NIL questions because we cannot get the sentence collections. Therefore, the AE module will fail on all of the NIL questions and the number of the valid questions should be 362 (413 – 51). In the experiment, we still test the module on the whole question set (413 questions) to keep consistent with the other's work. The training set contains 1252 questions. The performance of our system is evaluated using the mean reciprocal rank (MRR). Furthermore, we also list the percentages of the correct answers respectively in terms of the top 5 answers and the top 1 answer returned. No post-processes are used to adjust the answers in the experiments.

In order to evaluate the effectiveness of the syntactic relation patterns in the answer extraction, we compare the modules based on different feature sets. The first ME module *ME1* uses the common features including NE features, Orthographic features,

Syntactic Tag features and Triggers. The second ME module *ME2* uses the common features and some hand-extracted relation features, described in Section 5.2. The third module *ME3* uses the common features and the syntactic relation patterns which are automatically extracted and partial matched with the methods proposed in Section 3 and 4. Table 4 shows the overall performance of the modules. Both *ME2* and *ME3* outperform *ME1* by 3.15 MRR and 6.91 MRR respectively. This may indicate that the syntactic relations between the question words and the answers are useful for the answer extraction. Furthermore, *ME3* got the higher performance (+3.76 MRR) than *ME2*. The probable reason may be that the relations extracted by some heuristic rules in *ME2* are limited in the very local contexts of the nodes and they may not be sufficient. On the contrary, the pattern extraction methods we proposed can explore the larger relation space in the dependency trees.

Table 4. Overall performance

	ME1	ME2	ME3
Top1	44.06	47.70	51.81
Top5	53.27	55.45	58.85
MRR	47.75	50.90	54.66

Table 5. Performances for two pattern matching methods

	ExactMatch (λ =1)	PartialMatch				
		λ =0.8	λ =0.6	λ =0.4	λ =0.2	λ =0
Top1	50.12	51.33	**51.81**	51.57	50.12	50.12
Top5	57.87	58.37	**58.85**	58.60	57.16	57.16
MRR	53.18	54.18	**54.66**	54.41	52.97	52.97

Furthermore, we evaluate the effectiveness of the pattern matching method in Section 4. We compare two pattern matching methods: the exact matching (*ExactMatch*) and the partial matching (*PartialMatch*) using the tree kernel. Table 5 shows the performances for the two pattern matching methods. For *PartialMatch*, we also evaluate the effect of the parameter λ (described in Section 5.2) on the performance. In Table 5, the best *PartialMatch* (λ = 0.6) outperforms *ExactMatch* by 1.48 MRR. Since the pattern sets extracted from the training data is not large enough to cover the unseen cases, *ExactMatch* may have too low coverage and suffer with the data sparseness problem when testing, especially for *PSet_subject* (24.32% coverage using *ExactMatch* vs. 49.94% coverage using *PartialMatch*). In addition, even the model with *ExactMatch* is better than *ME2* (common features + hand-extracted relations) by 2.28 MRR. It indicates that the relation patterns explored with the method proposed in Section 3 are more effective than the relations extracted by the heuristic rules.

Table 6 shows the size of the pattern sets *PSet_target*, *PSet_head*, *PSet_subject* and *PSet_verb* and their coverage for the test data based on different λ values. *PSet_verb* gets the low coverage (<5% coverage). The probable reason is that the verbs in the answer sentences are often different from those in the questions, therefore only a few question verbs can be matched in the answer sentences. *PSet_head* also gets the relatively low coverage since the head words are only exacted from *how* questions and there are only 49/413 *how* questions with head words in the test data.

Table 6. Size and coverage of the pattern sets

	size	coverage (*%)					
		λ =1	λ =0.8	λ =0.6	λ =0.4	λ =0.2	λ =0
PSet_target	45	49.85	53.73	57.01	58.14	58.46	58.46
PSet_head	42	5.82	6.48	6.69	6.80	6.80	6.80
PSet_subject	123	24.32	44.82	49.94	51.29	51.84	51.84
PSet_verb	125	2.21	3.49	3.58	3.58	3.58	3.58

We further evaluate the contributions of different types of patterns. We respectively combine the pattern features in different pattern set and the common features. Some findings can be concluded from Table 7: All of the patterns have the positive effects based on the common features, which indicates that all of the four types of the relations are helpful for answer extraction. Furthermore, *P_target* (+4.21 MRR) and *P_subject* (+2.47 MRR) are more beneficial than *P_head* (+1.25 MRR) and *P_verb* (+0.19 MRR). This may be explained that the target and subject patterns may have the effect on the more test data than the head and verb patterns since *PSet_target* and *PSet_subject* have the higher coverage for the test data than *PSet_head* and *PSet_verb*, as shown in Table 6.

Table 7. Performance on feature combination

Combination of features	MRR
common features	47.75
common features + P_target	51.96
common features + P_head	49.00
common features + P_subject	50.22
common features + P_verb	47.94

7 Conclusion

In this paper, we study the syntactic relation patterns for question answering. We extract the various syntactic relations between the answers and different types of question words, including target words, head words, subject words and verbs and score the extracted relations based on support and confidence measures. We further propose a QA-specific tree kernel to partially match the relation patterns from the unseen data to the pattern sets. Lastly, we incorporate the patterns and some common features into a Maximum Entropy Model to rank the answer candidates. The experiment shows that the syntactic relation patterns improve the performance by 6.91 MRR based on the common features. Moreover, the contributions of the pattern matching methods are evaluated. The results show that the tree kernel-based partial matching outperforms the exact matching by 1.48 MRR. In the future, we are to further explore the syntactic relations using the web data rather than the training data.

References

1. Berger, A., Della Pietra, S., Della Pietra, V.: A maximum entropy approach to natural language processing. Computational Linguistics (1996), vol. 22, no. 1, pp. 39-71
2. Collins, M.: A New Statistical Parser Based on Bigram Lexical Dependencies. In: Proceedings of ACL-96 (1996) 184-191
3. Collins, M.: New Ranking Algorithms for Parsing and Tagging: Kernel over Discrete Structures, and the Voted Perceptron. In: Proceeings of ACL-2002 (2002).
4. Collins, M., Duffy, N.: Convolution Kernels for Natural Language. Advances in Neural Information Processing Systems 14, Cambridge, MA. MIT Press (2002)
5. Culotta, A., Sorensen, J.: Dependency Tree Kernels for Relation Extraction. In: Proceedings of ACL-2004 (2004)
6. Darroch, J., Ratcliff, D.: Generalized iterative scaling for log-linear models. The annuals of Mathematical Statistics (1972), vol. 43, pp. 1470-1480
7. Echihabi, A., Hermjakob, U., Hovy, E., Marcu, D., Melz, E., Ravichandran, D.: Multiple-Engine Question Answering in TextMap. In: Proceedings of the TREC-2003 Conference, NIST (2003)
8. Fellbaum, C.: WordNet - An Electronic Lexical Database. MIT Press, Cambridge, MA (1998)
9. Harabagiu, S., Moldovan, D., Clark, C., Bowden, M., Williams, J., Bensley, J.: Answer Mining by Combining Extraction Techniques with Abductive Reasoning. In: Proceedings of the TREC-2003 Conference, NIST (2003)
10. Ittycheriah, A., Roukos, S.: IBM's Statistical Question Answering System - TREC 11. In: Proceedings of the TREC-2002 Conference, NIST (2002)
11. Levenshtein, V. I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Doklady Akademii Nauk SSSR 163(4) (1965) 845-848
12. Ravichandran, D., Hovy, E., Och, F. J.: Statistical QA - Classifier vs. Re-ranker: What's the difference? In: Proceedings of Workshop on Multilingual Summarization and Question Answering, ACL (2003)
13. Ravichandran, D., Hovy, E.: Learning Surface Text Patterns for a Question Answering System. In: Proceedings of ACL-2002 (2002) 41-47
14. Soubbotin, M. M., Soubbotin, S. M.: Patterns of Potential Answer Expressions as Clues to the Right Answer. In: Proceedings of the TREC-10 Conference, NIST (2001)
15. Xu, J., Licuanan, A., May, J., Miller, S., Weischedel, R.: TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In: Proceedings of the TREC-2002 Conference, NIST (2002)
16. Vapnik, V.: Statistical Learning Theory, John Wiley, NY, (1998) 732.
17. Zelenko, D., Aone, C., Richardella, A.: Kernel Methods for Relation Extraction. Journal of Machine Learning Research (2003) 1083-1106.