Studying Feature Generation from Various Data Representations for Answer Extraction

Dan Shen[†‡] **Geert-Jan M. Kruijff**[†] **Dietrich Klakow**[‡]

[†] Department of Computational Linguistics
Saarland University
Building 17,Postfach 15 11 50
66041 Saarbruecken, Germany

[‡] Lehrstuhl Sprach Signal Verarbeitung
Saarland University
Building 17, Postfach 15 11 50
66041 Saarbruecken, Germany

{dshen,gj}@coli.uni-sb.de
{dietrich.klakow}@lsv.uni-saarland.de

Abstract

In this paper, we study how to generate features from various data representations, such as surface texts and parse trees, for answer extraction. Besides the features generated from the surface texts, we mainly discuss the feature generation in the parse trees. We propose and compare three methods, including feature vector, string kernel and tree kernel, to represent the syntactic features in Support Vector Machines. The experiment on the TREC question answering task shows that the features generated from the more structured data representations significantly improve the performance based on the features generated from the surface texts. Furthermore, the contribution of the individual feature will be discussed in detail.

1 Introduction

Open domain question answering (QA), as defined by the TREC competitions (Voorhees, 2003), represents an advanced application of natural language processing (NLP). It aims to find exact answers to open-domain natural language questions in a large document collection. For example:
Q2131: Who is the mayor of San Francisco?
Answer: Willie Brown

A typical QA system usually consists of three basic modules: 1. Question Processing (QP) Module, which finds some useful information from the 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 question key words. 3. Answer Extraction (AE) Module, which analyzes the relevant sentences using the information provided by the QP module and identify the answer phrase.

In recent years, QA systems trend to be more and more complex, since many other NLP techniques, such as named entity recognition, parsing, semantic analysis, reasoning, and external resources, such as WordNet, web, databases, are incorporated. The various techniques and resources may provide the indicative evidences to find the correct answers. These evidences are further combined by using a pipeline structure, a scoring function or a machine learning method.

In the machine learning framework, it is critical but not trivial to generate the features from the various resources which may be represented as surface texts, syntactic structures and logic forms, etc. The complexity of feature generation strongly depends on the complexity of data representation. Many previous QA systems (Echihabi et al., 2003; Ravichandran, et al., 2003; Ittycheriah and Roukos, 2002; Ittycheriah, 2001; Xu et al., 2002) have well studied the features in the surface texts. In this paper, we will use the answer extraction module of QA as a case study to further explore how to generate the features for the more complex sentence representations, such as parse tree. Since parsing gives the deeper understanding of the sentence, the features generated from the parse tree are expected to improve the performance based on the features generated from the surface text. The answer extraction module is built using Support Vector Machines (SVM). We propose three methods to represent the features in the parse tree: 1. features are designed by domain experts, extracted from the parse tree and represented as a feature vector; 2. the parse tree is transformed to a node sequence and a string kernel is employed; 3. the parse tree is retained as the original representation and a tree kernel is employed.

Although many textual features have been used in the others' AE modules, it is not clear that how much contribution the individual feature makes. In this paper, we will discuss the effectiveness of each individual textual feature in detail. We further evaluate the effectiveness of the syntactic features we proposed. Our experiments using TREC questions show that the syntactic features improve the performance by 7.57 MRR based on the textual features. It indicates that the new features based on a deeper language understanding are necessary to push further the machine learning-based QA technology. Furthermore, the three representations of the syntactic features are compared. We find that keeping the original data representation by using the data-specific kernel function in SVM may capture the more comprehensive evidences than the predefined features. Although the features we generated are specific to the answer extraction task, the comparison between the different feature representations may be helpful to explore the syntactic features for the other NLP applications.

2 Related Work

In the machine learning framework, it is crucial to capture the useful evidences for the task and integrate them effectively in the model. Many researchers have explored the rich textual features for the answer extraction.

IBM (Ittycheriah and Roukos, 2002; Ittycheriah, 2001) used a Maximum Entropy model to integrate the rich features, including query expansion features, focus matching features, answer candidate co-occurrence features, certain word frequency features, named entity features, dependency relation features, linguistic motivated features and surface patterns. ISI's (Echihabi et al. 2003; Echihabi and Marcu, 2003) statistical-based AE module implemented a noisy-channel model to explain how a given sentence tagged with an answer can be rewritten into a question through a sequence of stochastic operations. (Ravichandran et al., 2003) compared two maximum entropy-based QA systems, which view the AE as a classification problem and a re-ranking problem respectively, based on the word frequency features, expected answer class features, question word absent features and word match features. BBN (Xu et al. 2002) used a HMM-based IR system to score the answer candidates based on the answer contexts. They further re-ranked the scored answer candidates using the constraint features, such as whether a numerical answer quantifies the correct noun, whether the answer is of the correct location sub-type and whether the answer satisfies the verb arguments of the questions. (Suzuki et al. 2002) explored the answer extraction using SVM.

However, in the previous statistical-based AE modules, most of the features were extracted from the surface texts which are mainly based on the key words/phrases matching and the key word frequency statistics. These features only capture the surface-based information for the proper answers and may not provide the deeper understanding of the sentences. In addition, the contribution of the individual feature has not been evaluated by them. As for the features extracted from the structured texts, such as parse trees, only a few works explored some predefined syntactic relation features by partial matching. In this paper, we will explore the syntactic features in the parse trees and compare the different feature representations in SVM. Moreover, the contributions of the different features will be discussed in detail.

3 Answer Extraction

Given a question Q and a set of relevant sentences *SentSet* which is returned by the IR module, we consider all of the base noun phrases and the words in the base noun phrases as answer candidates ac_i. For example, for the question "*Q1956: What country is the largest in square miles?*", we extract the answer candidates { *Russia*, *largest country*, *largest*, *country*, *world*, *Canada*, *No.2.*} in the sentence "*I recently read that Russia is the largest country in the world, with Canada No. 2.*" The goal of the AE module is to choose the most probable answer from a set of answer candidates $\{ac_1, ac_2, ...ac_m\}$ for the question Q.

We regard the answer extraction as a classification problem, which classify each question and answer candidate pair <*Q*, *ac*$_i$> into the positive class (the correct answer) and the negative class (the incorrect answer), based on some features. The predication for each <*Q*, *ac*$_i$> is made independently by the classifier, then, the *ac* with the most confident positive prediction is chosen as the answer for *Q*. SVM have shown the excellent performance for the binary classification, therefore, we employ it to classify the answer candidates.

Answer extraction is not a trivial task, since it involves several components each of which is fairly sophisticated, including named entity recognition, syntactic / semantic parsing, question analysis, etc. These components may provide some indicative evidences for the proper answers. Before generating the features, we process the sentences as follows:

1. tag the answer sentences with named entities.
2. parse the question and the answer sentences using the Collins' parser (Collin, 1996).
3. extract the key words from the questions, such as the target words, query words and verbs.

In the following sections, we will briefly introduce the machine learning algorithm. Then, we will discuss the features in detail, including the motivations and representations of the features.

4 Support Vector Machines

Support Vector Machines (SVM) (Vapnik, 1995) have strong theoretical motivation in statistical learning theory and achieve excellent generalization performance in many language processing applications, such as text classification (Joachims, 1998).

SVM constructs a binary classifier that predict whether an instance $\boldsymbol{x}$ ($\boldsymbol{w} \in \boldsymbol{R}^n$) is positive ($f(\boldsymbol{x}) = 1$) or negative ($f(\boldsymbol{x}) = -1$), where, an instance may be represented as a feature vector or a structure like sequence of characters or tree. In the simplest case (linearly separable instances), the decision $f(\boldsymbol{x}) = sgn(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ is made based on a separating hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ ($\boldsymbol{w} \in \boldsymbol{R}^n$, $b \in \boldsymbol{R}$). All instances lying on one side of the hyperplane are classified to a positive class, while others are classified to a negative class.

Given a set of labeled training instances $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), ..., (\boldsymbol{x}_m, y_m)\}$, where $\boldsymbol{x}_i \in \boldsymbol{R}^n$ and $y_i = \{1, -1\}$, SVM is to find the optimal hyperplane that separates the positive and negative training instances with a maximal margin. The margin is defined as the distance from the separating hyperplane to the closest positive (negative) training instances. SVM is trained by solving a dual quadratic programming problem.

Practically, the instances are non-linearly separable. For this case, we need project the instances in the original space $\boldsymbol{R}^n$ to a higher dimensional space $\boldsymbol{R}^N$ based on the kernel function $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = <\Phi(\boldsymbol{x}_1), \Phi(\boldsymbol{x}_2)>$,where, $\Phi(\boldsymbol{x}): \boldsymbol{R}^n \rightarrow \boldsymbol{R}^N$ is a project function of the instance. By this means, a linear separation will be made in the new space. Corresponding to the original space $\boldsymbol{R}^n$, a non-linear separating surface is found. The kernel function has to be defined based on the Mercer's condition. Generally, the following kernel functions are widely used.

Polynomial kernel: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1)^p$

Gaussian RBF kernel: $k(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2}$

5 Textual Features

Since the features extracted from the surface texts have been well explored by many QA systems (Echihabi et al., 2003; Ravichandran, et al., 2003; Ittycheriah and Roukos, 2002; Ittycheriah, 2001; Xu et al., 2002), we will not focus on the textual feature generation in this paper. Only four types of the basic features are used:

1. **Syntactic Tag Features**: the features capture the syntactic/POS information of the words in the answer candidates. For the certain question, such as "*Q1903: How many time zones are there in the world?*", if the answer candidate consists of the words with the syntactic tags "*CD NN*", it is more likely to be the proper answer.
2. **Orthographic Features**: the features capture the surface format of the answer candidates, such as capitalization, digits and lengths, etc. These features are motivated by the observations, such as, the length of the answers are often less than 3 words for the factoid questions; the answers may not be the subsequences of the questions; the answers often contain digits for the certain questions.
3. **Named Entity Features**: the features capture the named entity information of the answer

candidates. They are very effective for the *who*, *when* and *where* questions, such as, For "*Q1950: Who created the literary character Phineas Fogg?*", the answer "*Jules Verne*" is tagged as a PERSON name in the sentences "*Jules Verne 's Phileas Fogg made literary history when he traveled around the world in 80 days in 1873.*". For the certain question target, if the answer candidate is tagged as the certain type of named entity, one feature fires.

4. **Triggers**: some trigger words are collected for the certain questions. For examples, for "*Q2156: How fast does Randy Johnson throw?*", the trigger word "*mph*" for the question words "*how fast*" may help to identify the answer "*98-mph*" in "*Johnson throws a 98-mph fastball*".

6 Syntactic Features

In this section, we will discuss the feature generation in the parse trees. Since parsing outputs the highly structured data representation of the sentence, the features generated from the parse trees may provide the more linguistic-motivated explanation for the proper answers. However, it is not trivial to find the informative evidences from a parse tree.

The motivation of the syntactic features in our task is that the proper answers often have the certain syntactic relations with the question key words. Table 1 shows some examples of the typical syntactic relations between the proper answers (*a*) and the question target words (*qtarget*). Furthermore, the syntactic relations between the answers and the different types of question key words vary a lot. Therefore, we capture the relation features for the different types of question words respectively. The question words are divided into four types:

- Target word, which indicates the expected answer type, such as "*city*" in "*Q: What city is Disneyland in?*".
- Head word, which is extracted from *how* questions and indicates the expected answer head, such as "*dog*" in "*Q210: How many dogs pull …?*"
- Subject words, which are the base noun phrases of the question except the target word and the head word.
- Verb, which is the main verb of the question.

To our knowledge, the syntactic relation features between the answers and the question key words haven't been explored in the previous machine learning-based QA systems. Next, we will propose three methods to represent the syntactic relation features in SVM.

6.1 Feature Vector

It is the commonly used feature representation in most of the machine learning algorithms. We predefine a set of syntactic relation features, which is an enumeration of some useful evidences of the answer candidates (ac) and the question key words in the parse trees. 20 syntactic features are manually designed in the task. Some examples of the 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 the child of the qsubject node, one feature fires.

The limitation of the manually designed features is that they only capture the evidences in the local context of the answer candidates and the question key words. However, some question words, such as subject words, often have the long range syntac-

1. *a* node is the same as the *qtarget* node and *qtarget* is the hypernym of *a*.
Q: What *city* is Disneyland in? S: Not bad for a struggling actor who was working at <u>*Tokyo*</u> Disneyland a few years ago.
2. *a* node is the parent of *qtarget* node.
Q: What is the name of the *airport* in Dallas Ft. Worth? S: Wednesday morning, the low temperature at the <u>Dallas-Fort Worth International *Airport*</u> was 81 degrees.
3. *a* node is the sibling of the *qtarget* node.
Q: What *book* did Rachel Carson write in 1962? S: In her 1962 *book* <u>Silent Spring</u>, Rachel Carson, a marine biologist, chronicled DDT 's poisonous effects, ….

Table 1: Examples of the typical relations between answer and question target word. In Q, the italic word is question target word. In S, the italic word is the question target word which is mapped in the answer sentence; the underlined word is the proper answer for the question Q.

tic relations with the answers. To overcome the limitation, we will propose some special kernels which may keep the original data representation instead of explicitly enumerate the features, to explore a much larger feature space.

6.2 String Kernel

The second method represents the syntactic relation as a linked node sequence and incorporates a string kernel in SVM to handle the sequence.

We extract a path from the node of the answer candidate to the node of the question key word in the parse tree. The path is represented as a node sequence linked by symbols indicating upward or downward movement through the tree. For example, in Figure 1, the path from the answer candidate node "*211,456 miles*" to the question subject word node "*the moon*" is "$NPB \uparrow ADVP \uparrow VP \uparrow S \downarrow NPB$", where "$\uparrow$" and "$\downarrow$" indicate upward movement and downward movement in the parse tree. By this means, we represent the object from the original parse tree to the node sequence. Each character of the sequence is a syntactic/POS tag of the node. Next, a string kernel will be adapted to our task to calculate the similarity between two node sequences.

Q1980: How far is the moon from Earth in miles?
S: At its perigee, the closest approach to Earth , the moon is 221,456 miles away.

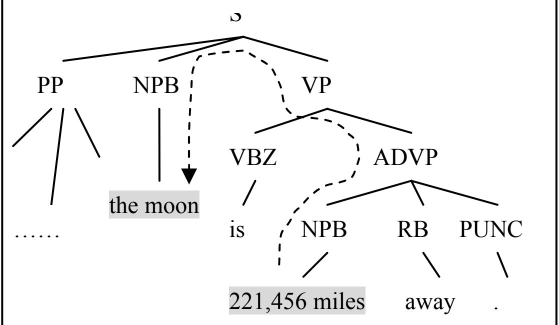


Figure 1: An example of the path from the answer candidate node to the question subject word node

(Haussler, 1999) first described a convolution kernel over the strings. (Lodhi et al., 2000) applied the string kernel to the text classification. (Leslie et al., 2002) further proposed a spectrum kernel, which is simpler and more efficient than the previous string kernels, for protein classification problem. In their tasks, the string kernels achieved the better performance compared with the human-defined features.

The string kernel is to calculate the similarity between two strings. It is based on the observation that the more common substrings the strings have, the more similar they are. The string kernel we used is similar to (Leslie et al., 2002). It is defined as the sum of the weighted common substrings. The substring is weighted by an exponentially decaying factor λ (set 0.5 in the experiment) of its length *k*. For efficiency, we only consider the substrings which length are less than 3. Different from (Leslie et al., 2002), the characters (syntactic/POS tag) of the string are linked with each other. Therefore, the matching between two substrings will consider the linking information. Two identical substrings will not only have the same syntactic tag sequences but also have the same linking symbols. For example, for the node sequences $NP \uparrow VP \uparrow VP \uparrow S \downarrow NP$ and $NP \uparrow NP \uparrow VP \downarrow NP$, there is a matched substring ($k = 2$): $NP \uparrow VP$.

6.3 Tree Kernel

The third method keeps the original representation of the syntactic relation in the parse tree and incorporates a tree kernel in SVM.

Tree kernels are the structure-driven kernels to calculate the similarity between two trees. They have been successfully accepted in the NLP applications. (Collins and Duffy, 2002) defined a kernel on parse tree and used it to improve parsing. (Collins, 2002) extended the approach to POS tagging and named entity recognition. (Zelenko et al., 2003; Culotta and Sorensen, 2004) further explored tree kernels for relation extraction.

We define an object (a relation tree) as the smallest tree which covers one answer candidate node and one question key word node. Suppose that a relation tree *T* has 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 represents the local characteristics of t_i. In our task, the set of the attributes includes Type attributes, Orthographic attributes and Relation Role attributes, as shown in Table 2. 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 is calculated by dynamic programming and captures the long-range syntactic relations between two nodes. The kernel we use is similar to (Zelenko et al., 2003) 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

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

Figure 2 shows two examples of the relation tree *T1_ac#targetword* and *T2_ac#targetword*. The kernel we used matches the following pairs of the nodes <t_0, w_0>, <t_1, w_2>, <t_2, w_2> and <t_4, w_1>.

Attributes		Examples
Type	POS tag	CD, NNP, NN…
	syntactic tag	NP, VP, …
Ortho-graphic	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

Table 2: Attributes of the nodes

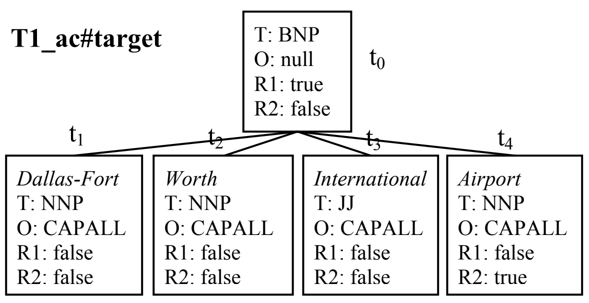
7 Experiments

We apply the AE module to the TREC QA task. To evaluate the features in the AE module independently, we suppose that the IR module has got 100% precision and only passes those sentences containing the proper answers to the AE module. 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

Q1897: What is the name of the airport in Dallas Ft. Worth?
S: Wednesday morning, the low temperature at the **Dallas-Fort Worth International Airport** was 81 degrees.



Q35: What is the name of the highest mountain in Africa?
S: **Mount Kilimanjaro**, at 19,342 feet, is Africa's highest mountain, and is 5,000 feet higher than ….

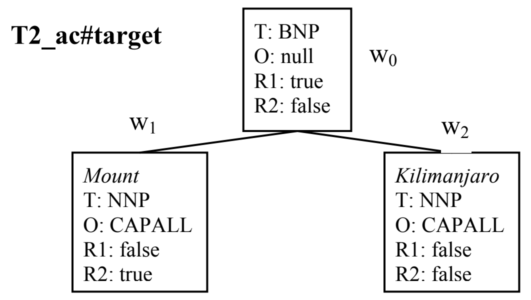


Figure 2: Two objects representing the relations between answer candidates and target words.

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. We employ the SVM^{Light} (Joachims, 1999) to incorporate the features and classify the answer candidates. No post-processes are used to adjust the answers in the experiments.

Firstly, we evaluate the effectiveness of the textual features, described in Section 5. We incorporate them into SVM using the three kernel functions: linear kernel, polynomial kernel and RBF kernel, which are introduced in Section 4. Table 3 shows the performance for the different kernels. The RBF kernel (46.24 MRR) significantly outperforms the linear kernel (33.72 MRR) and the polynomial kernel (40.45 MRR). Therefore, we will use the RBF kernel in the rest experiments.

	Top1	Top5	MRR
linear	31.28	37.91	33.72
polynomial	37.91	44.55	40.45
RBF	42.67	51.58	46.24

Table 3: Performance for kernels

In order to evaluate the contribution of the individual feature, we test out module using different feature combinations, as shown in Table 4. Several findings are concluded:

1. With only the syntactic tag features $F_{syn.}$, the module achieves a basic level MRR of 31.38. The questions "*Q1903: How many time zones are there in the world?*" is correctly answered from the sentence "*The world is divided into 24 time zones.*".
2. The orthographic features $F_{orth.}$ show the positive effect with 7.12 MRR improvement based on $F_{syn.}$. They help to find the proper answer "*Grover Cleveland*" for the question "*Q2049: What president served 2 nonconsecutive terms?*" from the sentence "*Grover Cleveland is the forgotten two-term American president.*", while $F_{syn.}$ wrongly identify "*president*" as the answer.
3. The named entity features F_{ne} are also beneficial as they make the 4.46 MRR increase based on $F_{syn.}$+$F_{orth.}$ For the question "*Q2076: What company owns the soft drink brand "Gatorade"?*", F_{ne} find the proper answer "*Quaker Oats*" in the sentence "*Marineau , 53 , had distinguished himself by turning the sports drink Gatorade into a mass consumer brand while an executive at Quaker Oats During his 18-month…*", while $F_{syn.}$+$F_{orth.}$ return the wrong answer "*Marineau*".
4. The trigger features F_{trg} lead to an improvement of 3.28 MRR based on $F_{syn.}$+F_{orth}+F_{ne}. They correctly answer more questions. For the question "*Q1937: How fast can a nuclear submarine travel?*", F_{trg} return the proper answer "*25 knots*" from the sentence "*The submarine , 360 feet (109.8 meters) long , has 129 crew members and travels at 25 knots.*", but the previous features fail on it.

F_{syn}	$F_{orth.}$	F_{ne}	F_{trg}	Top1	Top5	MRR
√				26.50	38.92	31.38
√	√			34.69	43.61	38.50
√	√	√		39.85	47.82	42.96
√	√	√	√	42.67	51.58	46.24

Table 4: Performance for feature combinations

Next, we will evaluate the effectiveness of the syntactic features, described in Section 6. Table 5 compares the three feature representation methods, *FeatureVector*, *StringKernel* and *TreeKernel*.

- *FeatureVector* (Section 6.1). We predefine some features in the syntactic tree and present them as a feature vector. The syntactic features are added with the textual features and the RBF kernel is used to cope with them.
- *StringKernel* (Section 6.2). No features are predefined. We transform the syntactic relations between answer candidates and question key words to node sequences and a string kernel is proposed to cope with the sequences. Then we add the string kernel for the syntactic relations and the RBF kernel for the textual features.
- *TreeKernel* (Section 6.3). No features are predefined. We keep the original representations of the syntactic relations and propose a tree kernel to cope with the relation trees. Then we add the tree kernel and the RBF kernel.

	Top1	Top2	MRR
$F_{syn.}$+$F_{orth.}$+F_{ne}+F_{trg}	42.67	51.58	46.24
FeatureVector	46.19	53.69	49.28
StringKernel	48.99	55.83	52.29
TreeKernel	50.41	57.46	53.81

Table 5: Performance for syntactic feature representations

Table 5 shows the performances of *FeatureVector*, *StringKernel* and *TreeKernel*. All of them improve the performance based on the textual features ($F_{syn.}$+$F_{orth.}$+F_{ne}+F_{trg}) by 3.04 MRR, 6.05 MRR and 7.57 MRR respectively. The probable reason may be that the features generated from the structured data representation may capture the more linguistic-motivated evidences for the proper answers. For example, the syntactic features help to find the answer "*nitrogen*" for the question "*Q2139: What gas is 78 percent of the earth 's atmosphere?*" in the sentence "*One thing they haven't found in the moon's atmosphere so far is nitrogen, the gas that makes up more than three-quarters of the Earth's atmosphere.*", while the textual features fail on it. Furthermore, the *StringKernel* (+3.01MRR) and *TreeKernel* (+4.53MRR) achieve the higher performance than *FeatureVector,* which may be explained that keeping the original data representations by incorporating the data-specific kernels in SVM may capture the more comprehensive evidences than the predefined features. Moreover, *TreeKernel* slightly outperforms *StringKernel* by 1.52 MRR. The reason may be that when we transform the representation of the syntactic relation from the tree to the node sequence, some information may be lost, such as the sibling node of the answer candidates. Sometimes the information is useful to find the proper answers.

8 Conclusion

In this paper, we study the feature generation based on the various data representations, such as surface text and parse tree, for the answer extraction. We generate the syntactic tag features, orthographic features, named entity features and trigger features from the surface texts. We further explore the feature generation from the parse trees which provide the more linguistic-motivated evidences for the task. We propose three methods, including feature vector, string kernel and tree kernel, to represent the syntactic features in Support Vector Machines. The experiment on the TREC question answering task shows that the syntactic features significantly improve the performance by 7.57MRR based on the textual features. Furthermore, keeping the original data representation using a data-specific kernel achieves the better performance than the explicitly enumerated features in SVM.

References

M. Collins. 1996. A New Statistical Parser Based on Bigram Lexical Dependencies. In *Proceedings of ACL-96*, pages 184-191.

M. Collins. 2002. New Ranking Algorithms for Parsing and Tagging: Kernel over Discrete Structures, and the Voted Perceptron. In *Proceedings of ACL-2002*.

M. Collins and N. Duffy. 2002. Convolution Kernels for Natural Language. *Advances in Neural Information Processing Systems 14*, Cambridge, MA. MIT Press.

A. Culotta and J. Sorensen. 2004. Dependency Tree Kernels for Relation Extraction. In *Proceedings of ACL-2004*.

A. Echihabi, U. Hermjakob, E. Hovy, D. Marcu, E. Melz, D. Ravichandran. 2003. Multiple-Engine Question Answering in TextMap. In *Proceedings of the TREC-2003 Conference*, NIST.

A. Echihabi, D. Marcu. 2003. A Noisy-Channel Approach to Question Answering. In Proceedings of the ACL-2003.

D. Haussler. 1999. Convolution Kernels on Discrete Structures. Technical Report UCS-CRL-99-10, University of California, Santa Cruz.

A. Ittycheriah and S. Roukos. 2002. IBM's Statistical Question Answering System – TREC 11. In *Proceedings of the TREC-2002 Conference*, NIST.

A. Ittycheriah. 2001. Trainable Question Answering System. Ph.D. Dissertation, Rutgers, The State University of New Jersey, New Brunswick, NJ.

T. Joachims. 1999. Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press, 1999.

T. Joachims. 1998. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In Proceedings of the European Conference on Machine Learning, Springer.

C. Leslie, E. Eskin and W. S. Noble. 2002. The spectrum kernel: A string kernel for SVM protein classification. Proceedings of the Pacific Biocomputing Symposium.

H. Lodhi, J. S. Taylor, N. Cristianini and C. J. C. H. Watkins. 2000. Text Classification using String Kernels. In *NIPS*, pages 563-569.

D. Ravichandran, E. Hovy and F. J. Och. 2003. Statistical QA – Classifier vs. Re-ranker: What's the difference? In *Proceedings of Workshop on Mulingual Summarization and Question Answering*, ACL 2003.

J. Suzuki, Y. Sasaki, and E. Maeda. 2002. SVM Answer Selection for Open-domain Question Answering. In Proc. of COLING 2002, pages 974–980.

V. N. Vapnik. 1998. Statistical Learning Theory. Springer.

E.M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the TREC-2003 Conference*, NIST.

J. Xu, A. Licuanan, J. May, S. Miller and R. Weischedel. 2002. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of the TREC-2002 Conference*, NIST.

D. Zelenko, C. Aone and A. Richardella. 2003. Kernel Methods for Relation Extraction. *Journal of Machine Learning Research*, pages 1083-1106.