Tree similarity measurement for classifying questions by syntactic structures

Zhiwei Lin, Hui Wang, and Sally McClean

Faculty of Computing and Engineering, Ulster University at Jordanstown , United Kingdom
{z.lin,h.wang,si.mcclean}@ulster.ac.uk

The accepted manuscript to appear in the 2016 International Conference on Intelligent Computation

Abstract. Question classification plays a key role in question answering systems as the classification result will be useful for effectively locating correct answers. This paper addresses the problem of question classification by syntactic structure. To this end, questions are converted into parsed trees and each corresponding parsed tree is represented as a multi-dimensional sequence (MDS). Under this transformation from questions to MDSs, a new similarity measurement for comparing questions with MDS representations is presented. The new measurement, based on the all common subsequences, is proved to be a kernel, and can be computed in quadratic time. Experiments show that the proposed method is competitive in terms of classification accuracy and efficiency.

1 Introduction

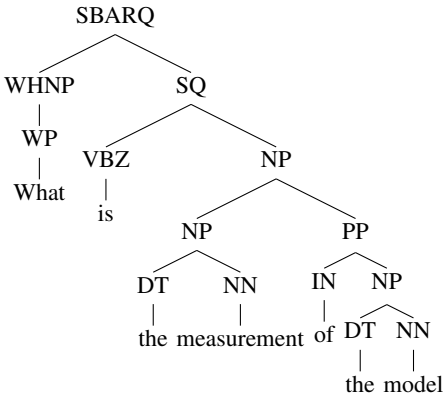
In Information retrieval (IR), *question answering* (QA) systems aim to find answers from a collection of documents, by matching a question with documents and then extracting information from the relevant documents. Key word based systems fail to find sensible answers because they ignore the inherent information from questions, such as syntactic structure information. For example, a keyword-based QA system can not differentiate the following two questions:

- What is the measurement of the model?
- What is the model of the measurement?

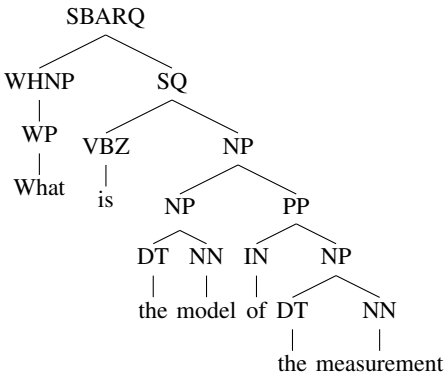
The keyword based approach does not use any structure information from the questions. However, if these two questions are converted into tree structures according to the parser (Fig. 1 shows the parsed trees for these two questions.), the difference between these two questions becomes obvious [1].

Syntactic information have been found very useful in QA systems and tree similarity measurement is key to this approach to utilizing syntactic information [3,14,17,10,11,15] [13,6,19,7,8,9]. In this paper we focus on question classification in QA systems, and present a new method for measuring similarity of questions based on their syntactic structures. Figure 2 shows a typical architecture of question answering systems [4]. It

[1] In the trees, SBARQ, WHNP, and *etc al* are tags defined in Penn Treebank II at http://bulba.sdsu.edu/jeanette/thesis/PennTags.html.



(a) 'What is the measurement of the model?'



(b) ' What is the model of the measurement?'

Fig. 1. An example of two parsed tree for two questions

is clear that question classification is a fundamental component of an efficient and effective QA system. Question classification is to determine semantic role (i.e., class) for a question so that the potential answers can be found according to the constraints imposed by their semantic roles. For example, to answer the question of 'What city will host the 2012 Olympics?', the question is firstly classified into class of *location* or class of *location* of a *city* in the fine-grained level. This class is then used to locate relevant candidate answers and the most irrelevant answers will be filtered out.

To improve question classification accuracy, syntactic structure representations for questions, where each question is converted into a tree by using natural language parser, have been found useful [15,12,13,19,7]. However, most of these studies investigated the problems at a coarse-grained level (e.g., *location*), and rarely at the fine-grained level (e.g., *city*). For example, for the question 'What city will host the 2012 Olympics?', the *location* is a coarse-grained class but it has to be further specified into a sub-class, e.g., *city*, as *location* alone may indicate different objects, such as city, country or mountain. To find the most sensible answers for QA systems, it is therefore necessary to use the

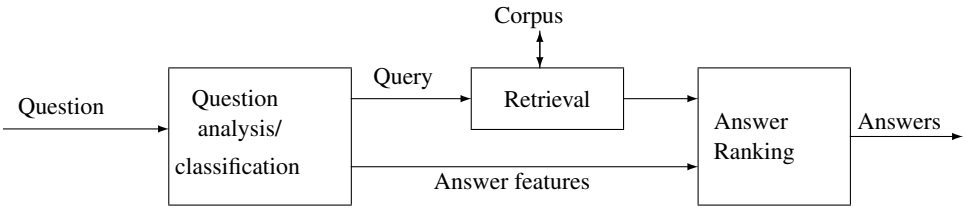


Fig. 2. A general architecture of QA system

fine-grained class information. Therefore, it is important to questions into fine-grained classes. This paper proposes a new algorithm for classifying questions into fine-grained classes. To this end, a novel tree similarity measurement for question classification is proposed by using a sequence combinatoric approach, which has quadratic time complexity. This measurement is proved to be a valid kernel, which means that it can be used in kernel based classifiers, e.g., support vector machines. Experimental results show that our method is competitive in terms of accuracy and efficiency.

This paper is organized as follows. Section 2 briefly reviews question classification. Section 3 describes in detail how to use multi-dimensional sequence to represent the syntactic structure of a question, how to measure the similarity of questions through the multidimensional sequence representation, and how to classify questions. Experimental evaluation results are presented in Section 4. The paper is concluded with a summary in Section 5.

2 A Review of question classification by syntactic structure

This section presents some key concepts about trees and sequences, followed by a review of question classification by syntactic structure.

2.1 Preliminaries

In this section, notations are introduced through a review of some key concepts about trees and sequences.

A *rooted, labeled and ordered tree* $\mathcal{T}$ is a tuple $\mathcal{T} = (V, E)$, where V and E are the set of nodes and the set of edges of tree $\mathcal{T}$. In this paper, we assume trees are *rooted, labeled*, and *left-right ordered* unless otherwise stated. The *size* of $\mathcal{T} = (V, E)$, denoted by $|\mathcal{T}|$, is the number of nodes in $\mathcal{T}$, i.e., $|\mathcal{T}| = |V|$. Let v be a node of $\mathcal{T}$, we write $|v|$ for the number of children of v, and a *leaf* v is a node without any child, i.e., $|v| = 0$. Let v be the i-th leaf in tree $\mathcal{T}$ in left-right order, we use $path(\mathcal{T}, i)$ or $path(\mathcal{T}, v)$ to denote the acyclic path starting from $root(\mathcal{T})$ to v. The *depth* of $v \in V, depth(v)$, is the number of nodes in the path between $root(\mathcal{T})$ and v. Let $v_1, v_2, \cdots, v_n$ be the children of v, the *pre-order traversal* of $\mathcal{T}(v)$ first visits v and then the subtrees routed at v_k (for all $1 \leq k \leq n$).

A *sequence* is a special tree where each node in the tree does not have any siblings. An *n-long* sequence s over Σ is an ordered set $\{s_1, s_2, \cdots, s_n\}$, where $s_i \in \Sigma$. The length

of sequence s is denoted by $|s|$. The *empty sequence*, which has a length of zero, is denoted by λ, i.e., $|\lambda| = 0$. A k-long sequence y is a *subsequence* of s , denoted by $y \preceq s$, if there exist $k + 1$ sequences $x^1, \ldots, x^{k+1}$ such that

$$s = x^1 y_1 \ldots x^k y_k x^{k+1} \tag{1}$$

We use $\Theta(s)$ to denote the set of all subsequences of sequence s.

2.2 Question classification by syntactic structure

Figure 2 shows a general architecture of a QA system, and in this figure, question classification is a fundamental component. In this section we review question classification and the state of art of question classification by syntactic structures.

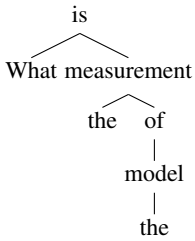


Fig. 3. Dependency tree of question 'What is the measurement of the model?'.

Question classification uses machine learning methods to classify a question into one of the pre-defined classes [3,14,19,15,12,13]. These classes imposes semantic constraints on candidate answers. There are 6 coarse grained classes and 50 fine grained subclasses in question classifications (more details can be found in [7] and their website [2]) .

Question classification by syntactic structures converts a question into a tree with natural language parser[3,14,15,12,13,19,8,9]. Figure 1 shows constituent trees and Figure 3 shows dependency tree. We can see that a constituent tree appears to be more informative than a dependency tree for the same sentence. However, the study in [12] shows that the dependency structure may lead to better accuracy than constituent tree. Additionally, for a given questions with $O(n)$ words, question classification by dependency structure should be more efficient than that by constituent tree because a constituent tree has $O(n \log n)$ nodes whereas a dependency tree has only $O(n)$ nodes.

Tree similarity measurement is key to classify questions in syntactic structure and tree kernel is widely used in question classification and wider applications in language processing [2,19]. In [2], a pre-terminal tree kernel (PTTK) was proposed and its time complexity is of $O(|\mathcal{S}| \times |\mathcal{T}|)$. While, The pre-terminal condition was relaxed and a new tree kernel (called QTK for short) was proposed for question classification [19].

[2] http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/definition.html

Different from tree kernel, tree edit distance is a distance function and it is measured by the minimum number of edit operations to transform one tree to another. The most common edit operations are: (1) insert a node into a tree; (2)delete a node from a tree; (3)change label of a node [20]. The optimal TED (OTED for short), allowing edit operations on any nodes, can provide a full picture of similarity between two trees and a dynamic programming algorithm by Zhang and Shasha in [20] to compute edit distance between two ordered trees needs $O(n^2 \times bound(n)^2)$, where $bound(n)$ is determined by the minimum of depth and number of leaves of trees. A recent study with TED is the approximate TED algorithm, (pq_Gram for short), which may be of $O(n \times \log n)$ time complexity [1]. Both OTED and pq_Gram are not kernel functions since their feature spaces do not comply with Hilbert space [16].

3 Subsequence combinatoric approach to tree Similarity

This section introduces how questions are classified by measuring similarity between them in tree representation, which includes 4 steps:

1. Parsed questions into trees with Stanford parser[3];
2. Convert parsed trees of questions into MDSs;
3. Use QACS algorithm to measure similarity between MDSs;
4. Integrate QACS measurements for question classification with kNN or SVM classifier.

3.1 Encoding a question into a multi-dimensional sequence

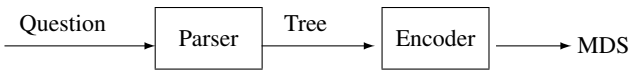


Fig. 4. Transform a question to an MDS

The first step to classify questions is to encode each question into an MDS through its parsed tree. Figure 4 shows how we encode a tree into a multi-dimensional sequence. First of all, a question is parsed into a tree by Stanford parser . The parsed tree could be a constituent tree or a dependency tree. For example, Figure 1 shows two constituent trees and Figure 3 shows a dependency tree.

We further encode a parsed tree (either in constituent or dependency structure) into an MDS so that the similarity between questions can be efficiently measured in terms of MDS representation. Suppose the parsed tree $\mathcal{T}$ has l leaves and let $path(\mathcal{T} : i)$ $(1 \leq i \leq l)$ be an acyclic path starting from $root(\mathcal{T})$ to the i-th leaf. We let T be a multi-dimensional sequence of $\mathcal{T}$ where each element T_i $(1 \leq i \leq l)$ in T is $path(\mathcal{T} : i)$.

[3] http://nlp.stanford.edu/software/lex-parser.shtml

Example 1. Table 1 shows an MDS for the parsed tree of question 'What is the measurement of the model?' (shown in Figure 1(a)).

Table 1. An MDS for the parsed tree in Figure 1(a)

1	2	3	4	5	6
SBARQ	SBARQ	SBARQ	SBARQ	SBARQ	SBARQ
WHNP	SQ	SQ	SQ	SQ	SQ
WP	VBZ	NP	NP	NP	NP
What	is	NP	NP	PP	PP
		DT	NN	IN	NP
		the	measurement	of	NN
					model

3.2 Subsequence combinatoric for MDSs

This section introduces how to measure the similarity between questions by MDS representation, following the previous section of how to encode each question into MDS through its parsed tree.

Subsequence combinatorics have been extensively studied [5], and one of the popular approaches is the counting of *all common subsequences* (ACS) in [18].

Let s and t be two sequences, and let $\Theta(s,t) = \Theta(s) \cap \Theta(t)$ be the set of all common subsequences of s and t, i.e., $\forall x \in \Theta(s,t)$,

$$x \preceq s \text{ and } x \preceq t,$$

we use $\theta(s,t) = |\Theta(s,t)|$ to denote the number of all common subsequences of s and t.

The $\theta(s,t)$ can be further interpreted as an inner product of two vectors by defining a feature space. Let $\mathbb{S}$ be a finite set of sequences, and $\mathscr{F}_{\mathbb{S}} = \{f_1, f_2, \cdots, f_n\}$ be a feature set of $\mathbb{S}$, where

$$\mathscr{F}_{\mathbb{S}} = \bigcup^{\forall s \in \mathbb{S}} \Theta(s).$$

Therefore, for each sequence s in $\mathbb{S}$, we define a feature vector $V^s = (v_1^s, v_2^s, \cdots, v_n^s)$, where

$$v_i^s = \begin{cases} 1 & \text{if } f_i \preceq s; \\ 0 & \text{otherwise.} \end{cases}$$

For two sequences $s, t \in \mathbb{S}$, the number of all common subsequences of s and t is an inner product of V^s and V^t, i.e.,

$$\theta(s,t) = \langle V^s, V^t \rangle = \sum_{i=1}^{n} v_i^s \times v_i^t. \tag{2}$$

This interpretation implies $\theta(s,t)$ is a valid kernel function, according to [16]. It is of exponential time complexity to enumerate all possible subsequences for all $s \in \mathbb{S}$

and therefore, it is necessary to find an efficient algorithm to calculate $\theta(s,t)$ instead of $\Theta(s,t)$. In the spirit of dynamic programming, an efficient algorithms is presented in [5].

Lemma 1 (Theorem 1 in [5]). *Consider two sequences s,t, where $m=|s|$ and $n=|t|$. For $x\in\Sigma$, if there exists k ($1\le k\le i$), such that $x=s_k$, let $\ell_s(i,x)=\max\{k|s_k=x\}$; otherwise, let $\ell_s(i,x)=0$. Then, for $1\le i\le m, 1\le j\le n$,*

$$\phi[i,j]=\begin{cases}\phi[i-1,j], & \text{if } \ell_t(j,s_i)=0;\\ \phi[i-1,j]+\phi[i-1,\ell_t(j,s_i)-1], & \text{if } \ell_t(j,s(i))>0, \ell_s(i-1,s_i)=0;\\ \phi[i-1,j]+\phi[i-1,\ell_t(j,s_i)-1] & \\ -\phi[\ell_s(i-1,s_i)-1,\ell_t(j,s_i)-1], & \text{otherwise.}\end{cases} \tag{3}$$

where, $\phi[i,j]$ is an $(m+1)\times(n+1)$ matrix, and $\phi[i,0]=1, \phi[0,j]=1$ and $\phi[0,0]=1$. Then, $\theta(s,t)=\phi[m,n]$.

The study in [18] presents a simplified version for $\theta(s,t)$ when there are no repeated symbols existing in s or in t. For this scenario, we can use the following equation instead of Equation 3.

$$\phi[i,j]=\begin{cases}\phi[i-1,j-1]\times 2, & \text{if } s_i=t_j;\\ \phi[i-1,j]+\phi[i,j-1]-\phi[i-1,j-1] & \text{otherwise.}\end{cases} \tag{4}$$

Extension of all common subsequences Both Equation 3 and Equation 4 have a common assumption that an equality check can be conducted for s_i and t_j if they are categorical or ordinals. However, this does not make sense to many applications with numerical sequences or multi-dimensional sequence, since equality check is not feasible.

To compare numerical sequences with the ACS approach, we need to further extend $\theta(s,t)$. One straightforward way to extend $\theta(s,t)$ is to estimate similarity degree between s_i and t_j [18]. To this end, a similarity probability $prob(s_i=t_j)$ is defined, referring to the similarity degree between s_i and t_j, where $0\le prob(\cdot)\le 1$. On the contrary, the dissimilarity probability $prob(s_i\neq t_j)$ between s_i and t_j is

$$prob(s_i\neq t_j)=1-prob(s_i=t_j)$$

Then the similarity between any two sequences $x=(x_1,x_2,\cdots,x_m)$ and $y=(y_1,y_2,\cdots,y_n)$, denoted by $\widehat{\theta}(x,y)$ [4], with an estimation of $prob(x_i=y_j)$, can be calculated as

$$\begin{aligned}\widehat{\phi}[i,j]&=(\widehat{\phi}[i-1,j]+\widehat{\phi}[i,j-1]-\widehat{\phi}[i-1,j-1])\times prob(s_i\neq t_j)\\ &\quad+\widehat{\phi}[i-1,j-1]\times 2\times prob(s_i=t_j) \qquad (5)\\ \widehat{\phi}[i,j]&=(\widehat{\phi}[i-1,j]+\widehat{\phi}[i,j-1])\times\Big(1-prob(s_i=t_j)\Big)\\ &\quad+\widehat{\phi}[i-1,j-1]\times\Big(3\times prob(s_i=t_j)-1\Big) \qquad (6)\end{aligned}$$

and $\widehat{\theta}(x,y)=\widehat{\phi}[m,n]$. We note that,

[4] We use x,y to denote more general sequences, either categorical or numerical.

- When s_i and t_j are categorical or ordinal, Equation 4 is a special case of Equation 6 if we let $prob(x_i = y_j) = 1$ for the case of $x_i = y_j$, and $prob(x_i = y_j) = 0$ otherwise.
- For numerical case, as an example, one could estimate $prob(x_i = y_j)$ by kernel functions, e.g, Gaussian kernel function with a tuning factor γ,

$$prob(x_i = y_j) = \exp(-\gamma|x_i - y_j|). \tag{7}$$

We further extend this idea for measuring similarity between multi-dimensional sequences. Consider two multi-dimensional sequences $S = (S_1, S_2, \cdots, S_m)$ and $T = (T_1, T_2, \cdots, T_n)$, and let $\widehat{\theta}(S, T)$ be a similarity function of S and T, we propose to measure $\widehat{\theta}(S, T)$ with an estimation function $prob(S_i = T_j)$ by

$$prob(S_i = T_j) = \frac{\theta(S_i, T_j)}{\sqrt{\theta(S_i, S_i) \times \theta(T_j, T_j)}}. \tag{8}$$

We have shown there is no doubt that according to Equation 2, $\theta(T_j, T_j)$ is a valid kernel as well as $prob(S_i = T_j)$. However, to apply $\widehat{\theta}(S, T)$ into SVM, we need to prove that $\widehat{\theta}(x, y)$ is a valid kernel so that an optimal solution can be guaranteed. The following theorem shows that by posing a constraint on estimating $prob(x_i = y_j)$, the $\widehat{\theta}(x, y)$ is a valid kernel.

Theorem 1. *$\widehat{\theta}(x, y)$ is a valid kernel if $prob(x_i = y_j)$ is estimated by a valid kernel function.*

Proof. The proof of this theorem is trivial. Since $\widehat{\theta}(x, y) = \widehat{\phi}[m, n]$ and $\widehat{\phi}[i, 0] = 1$ and $\widehat{\phi}[0, j] = 1$ (for $0 \leq i \leq m, 0 \leq j \leq n$), the operations (either by addition or multiplication[5]) in Equation 6 to combine $\widehat{\phi}[i-1, j]$, $\widehat{\phi}[i, j-1]$, $\widehat{\phi}[i-1, j-1]$ and $prob(x_i = y_j)$ for $\widehat{\phi}[i, j]$, result in an valid kernel if $prob(x_i = y_j)$ is estimated by a valid kernel function.

An $O(n^2)$ algorithm to compare MDSs Both $\theta(s, t)$ by Equation 3 and its extension $\widehat{\theta}(x, y)$ by Equation 6 are of quadratic complexity. If MDSs have $O(m)$ spatial dimensions and $O(n)$ sequential dimensions, then the time complexity of calculating similarity between MDSs by Equation 6 along with Equation 8 is $O(m^2n^2)$, which is very time consuming. Therefore it is necessary to find an efficient algorithm to compare MDSs. Cost reduction can be achieved through reducing redundant computation.

In QACS approach, nodes in a tree $\mathcal{S}$ are aligned in pre-order traversal sequence, and $\hat{i}$ is used to denote node's index in the sequence. Let $p(\hat{i})$ be the index of parent of

[5] The closure property of kernel function states that, let K_1 and K_2 be valid kernels, K is a valid kernel function if

$$K(x, y) = K_1(x, y) + K_2(x, y)$$
$$K(x, y) = K_1(x, y) \times K_2(x, y)$$
$$K(x, y) = c \times K_1(x, y)$$

Input: Two trees $\mathcal{S}$ and $\mathcal{T}$
Output: Similarity between $\mathcal{S}$ and $\mathcal{T}$ of QACS
`// Initiation for` $1 \le \hat{i} \le |\mathcal{S}|$ `and` $1 \le \hat{j} \le |\mathcal{T}|$ `from Line 1 to 3`
1 Initialize $\ell_{\mathcal{S}}[p(\hat{i}), \sigma(\hat{i})]$ and $\ell_{\mathcal{T}}[\hat{j}, \sigma(\hat{i})]$;
2 **for** $\hat{i} \leftarrow 1$ **to** $|\mathcal{S}|$ **do** $\phi_{\mathcal{S}}[\hat{i}] = \phi[\hat{i}, \hat{i}]$;;
3 **for** $\hat{j} \leftarrow 1$ **to** $|\mathcal{T}|$ **do** $\phi_{\mathcal{T}}[\hat{j}] = \phi[\hat{j}, \hat{j}]$;;
4 $k = l = 1$;
5 **for** $\hat{i} \leftarrow 1$ **to** $|\mathcal{S}|$ **do**
6 **for** $\hat{j} \leftarrow 1$ **to** $|\mathcal{T}|$ **do**
7 Use Equation 9 to calculate $\phi[\hat{i}, \hat{j}]$;
8 **if** $\hat{i}$, $\hat{j}$ *are leaves* **then**
9 $p = \frac{\phi[\hat{i}, \hat{j}]}{\sqrt{\phi_{\mathcal{S}}[\hat{i}] \times \phi_{\mathcal{T}}[\hat{j}]}}$;
10 Calculate $\widehat{\phi}[k, l]$ with p and Equation 6;
11 l++;
12 **end**
13 **end**
14 **if** $\hat{i}$ *is leaf* **then** k++;;
15 **end**
16 return $\widehat{\phi}[k-1, l-1]$;

Algorithm 3.1: Pseudocode for calculating QACS

node $\hat{i}$ (if $\hat{i}$ is a root node, i.e., $\hat{i} = 1$, let $p(\hat{i}) = 0$), and $\sigma(\hat{i})$ be the label of $\hat{i}$, for a given label $x \in \Sigma$, we define an $|\mathcal{S}| + 1$ vector $\hat{\ell}_{\mathcal{S}}$ for tree $\mathcal{S}$:

$$\hat{\ell}_{\mathcal{S}}[\hat{i}, x] = \begin{cases} 0, & \text{if } \hat{i} = 0; \\ \hat{i}, & \text{if } \sigma(\hat{i}) = x; \\ \hat{\ell}_{\mathcal{S}}[p(\hat{i}), x], & \textit{otherwise}. \end{cases}$$

Consider trees $\mathcal{S}$ and $\mathcal{T}$, and their pre-order index $\hat{i}$ and $\hat{j}$. For $1 \le \hat{i} \le |\mathcal{S}|$, given a label $\sigma(\hat{i})$, for brevity, let $\ell_{\mathcal{S}} = \hat{\ell}_{\mathcal{S}}[p(\hat{i}), \sigma(\hat{i})]$, and similarly, for $1 \le \hat{j} \le |\mathcal{T}|$, let $\ell_{\mathcal{T}} = \hat{\ell}_{\mathcal{T}}[\hat{j}, \sigma(\hat{i})]$. Let ϕ be an $(|\mathcal{S}| + 1) \times (|\mathcal{T}| + 1)$, we define:

$$\phi[\hat{i}, \hat{j}] = \begin{cases} \phi[p(\hat{i}), \hat{j}], & \text{if } \ell_{\mathcal{T}} = 0; \\ \phi[p(\hat{i}), \hat{j}] + \phi[p(i), p(\ell_{\mathcal{T}})], & \text{if } \ell_{\mathcal{T}} > 0, \ell_{\mathcal{S}} = 0; \\ \phi[p(\hat{i}), \hat{j}] + \phi[p(i), p(\ell_{\mathcal{T}})] - \phi[p(\ell_{\mathcal{S}}), p(\ell_{\mathcal{T}})], & \text{if } \ell_{\mathcal{T}} > 0, \ell_{\mathcal{S}} > 0. \end{cases} \quad (9)$$

where $\phi[\hat{i}, 0] = 1$, $\phi[0, \hat{j}] = 1$, and $\phi[0, 0] = 1$.

Based on the definition of Equation 9, we present the pseudo-code (in Algorithm 3.1) to compute QACS, which is quadratic in time complexity. In Algorithm 3.1, $\phi_{\mathcal{S}}$ on Line 2 and $\phi_{\mathcal{T}}$ on Line 3 are initiated and used to normalize $\phi[\hat{i}, \hat{j}]$. Together with the loops from Line 5, the time complexity of QACS is $O(|\mathcal{S}| \times |\mathcal{T}|)$.

4 Evaluation

In this section, we compare our method with the optimal tree edit distance (OTED) [20], and pq_Gram [1] and tree kernels (PTTK [2] and QTK [19])in terms of question classification accuracy and runtime.

We us kNN and SVM as classifiers for this evaluation due to the fact that both kNN and SVM are able to accept text, sequence or tree as input, compared to those parametric algorithms, such as the Bayesian approach, logistic regression or neural network algorithms. Both OTED and pq_Gram are distance functions, but not kernel functions as their feature spaces do not comply with Hilbert space [16]. Therefore we only use QACS and tree kernels in SVM classifier [1,20]. On the other hand, tree kernels are usually evaluated in SVM classifier, rarely in other classifiers. To have a balanced comparison, we also use kNN classifier to evaluate these similarity measurements. For kNN, the parameter k is set to $1, 3, 5, 7, 9, 11, 13$ and 15.

The dataset used in this evaluation is downloaded from UIUC Cognitive Computation Group [6], which contains 5952 questions. These questions are categorized into 6 coarse grained classes and 50 fine grained subclasses. This paper considers the problems of classification of questions into both coarse and fine grained classes.

We conducted experiments to evaluate the measurements with 10-fold CV strategy by a comprehensive experiments with both constituent and dependency tree structures.

The coarse-grained classification results are presented in Table 2 and Table 3. From the results, we find that SVM has significantly higher accuracy but consumes more time than the kNN classifier. In kNN classifier, OTED has a very competitive performance but its runtime is far too high compared to the others. Our method QACS has highest accuracy except that in kNN in terms of dependency structure.

The fine-grained classification results are presented in Table 4 and Table 5. From the results, we find that with constituent tree structure, kNN classifier can provide a higher classification accuracy than SVM, for example, the accuracy of QACS in SVM is 57.6% while its accuracy in kNN is 60.4%.

4.1 Discussion

The experimental results show that our method QACS is competitive in question classification by syntactic structure. There are still some interesting issues behind these results.

First, because constituent trees have rich information regarding grammatical rules, the constituent tree has significantly higher accuracy, in both coarse-grained and fine-grained classification. However, a dependency tree implicitly contains some grammatical information, through utilizing which we may have a competitive performance. But in terms of efficiency, obviously , for a question with n words, the time and space complexities of comparing constituent trees are $O(n \log n)$ but those complexities for dependency are $O(n)$. That is why the runtime for constituent trees are always higher than that for dependency trees (See Tables 3, 2, 5, and 4). As such, it is a trade-off between effectiveness and efficiency.

[6] Available at `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

Table 2. Coarse-grained classification accuracy (%) and runtime (seconds) with dependency tree structure.

	SVM/	kNN								kNN
	time	$k = 1$	3	5	7	9	11	13	15	time
QACS	74.5/3122	62.1	59	56.3	53.2	51.3	50	48.2	47.3	211
PTTK [2]	74.1/1079	61.6	60.4	57.8	57.1	55.9	56	55.8	55.6	98
QTK [19]	69.8/2059	53.4	50.9	49.2	48.3	47.3	47.4	47	46.9	185
OTED	N/A	67.9	68.9	68.4	68.3	67.5	66.9	66.6	66.6	1275
pq_Gram	N/A	54.3	46.6	41.4	37.7	35.1	32.8	31.0	29.4	312

There is another example of trade-off between effectiveness and efficiency, from the classifier point of view. For the same similarity measurement, different classifiers could produce significantly different accuracy. In most case, SVM is better but needs longer time than kNN. One more problem is that SVM has very strict requirement of a kernel function to guarantee optimal solution.

Finally, we find that OTED has very good performance when it is used in kNN classifier as it can provide a full picture of how similar between two trees are. The main problem with OTED is that it is a time-consuming method and can not be used in handling large datasets.

5 Conclusion

This paper addresses question classification by syntactic structure. In this paper, each question is parsed into a tree and the corresponding parsed tree is represented as a multi-dimensional sequence. With this MDS representation, we present a new similarity measurement for comparing questions in MDS representation, by extending the all common subsequences algorithm. The new algorithm is of $O(n^2)$ time complexity and is proved to be a valid kernel. The algorithm is integrated into kNN and SVM classifiers for question classification. Evaluation results show that our method is competitive in terms of classification accuracy and efficiency. Future work includes using this framework for analyzing the answers in QA system.

Acknowledgments

The authors would like to thank anonymous reviewers for their helpful comments to this paper by pointing out relevant literature and a number of annoying flaws in the submission.

References

1. Augsten, N., Böhlen, M., Gamper, J.: Approximate matching of hierarchical data using pq-grams. In: VLDB '05: Proceedings of the 31st international conference on Very large data bases. pp. 301–312. VLDB Endowment (2005)

Table 3. Coarse-grained classification accuracy (%) and runtime (seconds) with constituent tree structure.

	SVM/	kNN								kNN
	time	$k = 1$	3	5	7	9	11	13	15	time
QACS	75.2/18076	71.5	71.0	69.8	69.5	69.3	68.3	67.4	67.1	1140
PTTK [2]	74.9/9498	67.4	66.9	66.1	65.1	63.8	63.4	62.6	61.8	615
QTK [19]	74/23683	66.8	65.9	65.1	64.3	62.9	62.1	61.4	61.1	1480
OTED	N/A	65.3	66	66.6	66.7	66.3	66.3	65.9	65.8	14427
pq_Gram	N/A	68.5	68.9	68.4	68.2	68.0	67.1	66.7	66.3	1280

Table 4. Fine-grained classification accuracy (%) and runtime (seconds) with dependency tree structure

	SVM/	kNN								kNN
	time	$k = 1$	3	5	7	9	11	13	15	time
QACS	58.6/5682	55.4	53.4	51.6	49.1	47.0	45.2	43.9	42.9	200
PTTK [2]	59.5/1769	50.5	50.3	49.3	47.8	47.2	46.4	46.5	46.4	96
QTK [19]	54.9/3146	41.2	39.7	39.0	38.2	37.3	37.3	37.0	36.5	182
OTED	N/A	56.7	58.7	59.7	59.5	59.1	58.3	57.6	57.0	1269
pq_Gram	N/A	50.5	45.8	41.1	36.9	34.6	32.5	30.7	28.9	299

2. Collins, M., Duffy, N.: Convolution kernels for natural language. In: Advances in Neural Information Processing Systems 14. pp. 625–632. MIT Press (2001)
3. Croce, D., Basili, R., Moschitti, A.: Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project, chap. Semantic Tree Kernels for Statistical Natural Language Learning, pp. 93–113. Springer International Publishing, Cham (2015)
4. Croft, B., Metzler, D., Strohman, T.: Search Engines: Information Retrieval in Practice. Addison-Wesley Publishing Company, USA, 1st edn. (2009)
5. Elzinga, C., Rahmann, S., Wang, H.: Algorithms for subsequence combinatorics. Theoretical Computer Science 409(3), 394–404 (2008)
6. Feng, G., Xiong, K., Tang, Y., Cui, A., Bai, J., Li, H., Yang, Q., Li, M.: Question classification by approximating semantics. In: Proceedings of the 24th International Conference on World Wide Web. pp. 407–417. WWW '15 Companion, ACM, New York, NY, USA (2015)

Table 5. Fine-grained classification accuracy (%) and runtime (seconds) with constituent tree structure.

	SVM/	kNN								kNN
	time	$k = 1$	3	5	7	9	11	13	15	time
QACS	57.6/36650	60	60.4	59.8	58.6	58.0	57.1	56.9	56.1	1097
PTTK [2]	56.5/19044	55.6	56.7	55.7	54.6	53.6	53.2	52.8	52.5	595
QTK [19]	55.4/46480	55.0	56.0	54.8	53.9	52.9	52.2	51.9	51.3	1411
OTED	N/A	52.9	53.9	54.2	54.9	54.3	54.2	54.6	54.3	14123
pq_Gram	N/A	56.7	58.1	58.2	57.8	57.4	56.7	55.9	55.8	1204

7. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of the 19th international conference on Computational linguistics. pp. 1–7. Association for Computational Linguistics, Morristown, NJ, USA (2002)
8. Lin, Z., Wang, H., McClean, S.: Measuring tree similarity for natural language processing based information retrieval. In: Proceedings of the Natural Language Processing and Information Systems, and 15th International Conference on Applications of Natural Language to Information Systems. pp. 13–23. NLDB'10, Springer-Verlag, Berlin, Heidelberg (2010)
9. Lin, Z., Wang, H., McClean, S.: A multidimensional sequence approach to measuring tree similarity. IEEE Transactions on Knowledge and Data Engineering 24(2), 197–208 (Feb 2012)
10. Mittendorfer, M., Winiwarter, W.: Exploiting syntactic analysis of queries for information retrieval. Journal of Data and Knowledge Engineering (2002)
11. Moschitti, A.: Efficient convolution kernels for dependency and constituent syntactic trees. In: ECML 2006 : 17th European Conference on Machine Learning, Proceedings. pp. 318–329. Berlin, Germany (September 2006)
12. Moschitti, A.: Making tree kernels practical for natural language learning. In: Proceedings of the Eleventh International Conference on European Association for Computational Linguistics. Trento, Italy (2006)
13. Moschitti, A., Quarteroni, S., Basili, R., Manandhar, S.: Exploiting syntactic and shallow semantic kernels for question answer classification. In: In Proceeding of the Association for Computational Linguistics. pp. 776–783 (2007)
14. Pan, Y., Tang, Y., Lin, L., Luo, Y.: Question classification with semantic tree kernel. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 837–838. SIGIR '08, ACM, New York, NY, USA (2008)
15. Punyakanok, V., Roth, D., tau Yih, W.: Mapping dependencies trees: An application to question answering. In: Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics (2004)
16. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press (2004)
17. Strzalkowski, T. (ed.): Natural language Information Retrieval. Kluwer, New York (1999)
18. Wang, H.: All common subsequences. In: IJCAI'07 : Proceedings of the 20th international joint conference on Artifical intelligence. pp. 635–640. Hyderabad, India (2007)
19. Zhang, D., Lee, W.S.: Question classification using support vector machines. In: SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval. pp. 26–32. ACM, New York, NY, USA (2003)
20. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. SIAM Journal on Computing 18(6), 1245–1262 (1989)