Question Classification using Support Vector Machines

Dell Zhang[1,2]
[1]Department of Computer Science
School of Computing
S15-05-24, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2]Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744251
dell.z@ieee.org

Wee Sun Lee[1,2]
[1]Department of Computer Science
School of Computing
SOC1-05-26, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2]Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744526
leews@comp.nus.edu.sg

ABSTRACT
Question classification is very important for question answering. This paper presents our research work on automatic question classification through machine learning approaches. We have experimented with five machine learning algorithms: Nearest Neighbors (NN), Naïve Bayes (NB), Decision Tree (DT), Sparse Network of Winnows (SNoW), and Support Vector Machines (SVM) using two kinds of features: bag-of-words and bag-of-ngrams. The experiment results show that with only surface text features the SVM outperforms the other four methods for this task. Further, we propose to use a special kernel function called the tree kernel to enable the SVM to take advantage of the syntactic structures of questions. We describe how the tree kernel can be computed efficiently by dynamic programming. The performance of our approach is promising, when tested on the questions from the TREC QA track.

Categories and Subject Descriptors
H.3.1. [**Content Analysis and Indexing**], H.3.3 [**Information Search and Retrieval**].

General Terms
Algorithms, Experimentation.

Keywords
question answering, text classification, machine learning, support vector machine, kernel method.

1. INTRODUCTION
What a current information retrieval system or search engine can do is just "document retrieval", i.e., given some keywords it only returns the relevant documents that contain the keywords. However, what a user really wants is often a precise answer to a question. For instance, given the question "Who was the first American in space?", what a user really wants is the answer "Alan Shepard", but not to read through lots of documents that contain the words "first", "American" and "space" etc.

The Text Retrieval Conference, TREC (http://trec.nist.gov/), has launched a QA track to support the competitive research on question answering, from 1999 (TREC8). The focus of the TREC QA track is to build a fully automatic open-domain question answering system, which can answer factual questions based on very large document sets. Today, the TREC QA track [18][19][20] is the major large-scale evaluation environment for open-domain question answering systems.

In order to correctly answer a question, usually one needs to understand what the question asks for. Question Classification, i.e., putting the questions into several semantic categories, can not only impose some constraints on the plausible answers but also suggest different processing strategies. For instance, if the system understands that the question "Who was the first American in space?" asks for a person name, the search space of plausible answers will be significantly reduced. In fact, almost all the open-domain question answering systems include a question classification module. The accuracy of question classification is very important to the overall performance of the question answering system.

While document classification has been intensively studied [22], question classification is still a rather new research issue. There appears to be nontrivial differences between these two problems. For example, the common words like 'what', 'is', etc. should be neglected for document classification, but these "stop-words" are actually very important for question classification.

Although it is possible to manually write some heuristic rules for question classification, it usually requires tremendous amount of tedious work. One has to correctly figure out various forms of each specific type of questions in order to achieve reasonable accuracy for a manually constructed classification program. In contrast, a machine learning approach can automatically construct a high performance question classification program which leverages thousands or more features of questions. Given more training data, the performance of a learned classification

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGIR'03, July 28-August 1, 2003, Toronto, Canada.
Copyright 2003 ACM 1-58113-646-3/03/0007…$5.00.

program usually improves. Moreover, a learned classification program is more flexible than a manual one since it can be easily adapted to a new domain. However, there are only a few papers describing machine learning approaches to question classification, and some of them such as [17] are pessimistic.

This paper presents our research work on automatic question classification through machine learning approaches, especially the Support Vector Machines. It is organized as follows: Section 2 presents the question classification problem; Section 3 compares several machine learning approaches to question classification with conventional surface text features; Section 4 describes a special kernel function called tree kernel to enable the Support Vector Machines to take advantage of the syntactic structures of questions; Section 5 is the related work; and Section 6 concludes the paper.

2. QUESTION CLASSIFICATION

Question Classification means putting the questions into several semantic categories. Here only the TREC-style questions, i.e., open-domain factual questions, are considered.

We follow the two-layered question taxonomy proposed in [12], which contains 6 coarse grained categories and 50 fine grained categories, as shown in Table 1. Each coarse grained category contains a non-overlapping set of fine grained categories.

Most question answering systems use a coarse grained category definition. Usually the number of question categories is less than 20. However, it is obvious that a fine grained category definition is more beneficial in locating and verifying the plausible answers.

Table 1. The coarse and fine grained question categories.

Coarse	Fine
ABBR	abbreviation, expansion
DESC	definition, description, manner, reason
ENTY	animal, body, color, creation, currency, disease/medical, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
HUM	description, group, individual, title
LOC	city, country, mountain, other, state
NUM	code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight

To simplify the following experiments, we assume that one question resides in only one category. That is to say, an ambiguous question is labeled with its most probable category.

3. COMPARING SVM AND OTHER MACHINE LEARNING ALGORITHMS

3.1 Datasets

We used the publicly available training and testing datasets provided by USC [10], UIUC [12] and TREC [18][19][20]. All these datasets have been manually labeled by UIUC [12] according to the coarse and fine grained categories in Table 1. There are about 5,500 labeled questions randomly divided into 5 training datasets of sizes 1,000, 2,000, 3,000, 4,000 and 5,500 respectively. The testing dataset contains 500 labeled questions from the TREC10 QA track.

3.2 Features

In this section, we only consider the surface text features of questions. For each question, we extract two kinds of features: bag-of-words and bag-of-ngrams (all continuous word sequences in the question). Every question is represented as binary feature vectors, because the term frequency (tf) of each word or ngram in a question usually is 0 or 1.

3.3 Support Vector Machines

Support Vector Machines (SVM) [2] are linear functions of the form $f(x) = \langle \mathbf{w} \bullet \mathbf{x} \rangle + b$, where $\langle \mathbf{w} \bullet \mathbf{x} \rangle$ is the inner product between the weight vector $\mathbf{w}$ and the input vector $\mathbf{x}$. The SVM can be used as a classifier by setting the class to 1 if $f(x) > 0$ and to -1 otherwise. The main idea of SVM is to select a hyper-plane that separates the positive and negative examples while maximizing the minimum margin, where the margin for example $\mathbf{x}_i$ is $y_i f(\mathbf{x}_i)$ and $y_i \in \{-1, 1\}$ is the target output. This corresponds to minimizing $\langle \mathbf{w} \bullet \mathbf{w} \rangle$ subject to $y_i(\langle \mathbf{w} \bullet \mathbf{x}_i \rangle + b) \geq 1$ for all i. Large margin classifiers are known to have good generalization properties (see e.g. [2]).

To deal with cases where there may be no separating hyper-plane, the soft margin SVM has been proposed. The soft margin SVM minimizes $\langle \mathbf{w} \bullet \mathbf{w} \rangle + C \sum_i \xi_i$ subject to $y_i(\langle \mathbf{w} \bullet \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$ for all i, where C is a parameter that controls the amount of training errors allowed.

For multi-class SVM, one-against-one strategy has been demonstrated to work well [11]. An adaptation of the *LIBSVM* implementation [5] is used in the following.

3.4 Some Other Learning Algorithms

Besides the Support Vector Machines (SVM), we have tried four other machine learning algorithms: Nearest Neighbors (NN), Naïve Bayes (NB), Decision Tree (DT), and Sparse Network of Winnows (SNoW).

The Nearest Neighbors (NN) algorithm is a simplified version of the well-known kNN algorithm which has been successfully applied in document classification [22]. Given an unlabeled instance, the NN algorithm finds its nearest (most similar) neighbors among the training examples, and uses the dominant class label of these nearest neighbors as its class label. Here the similarity between two instances is simply defined as the number of overlapping features between them. If the instances are represented as binary feature vectors, the similarity function turns out to be the dot product function. There are more sophisticated versions of kNN algorithm, such as the one used in [22] and it may be interesting to test them in the future.

The Naïve Bayes (NB) algorithm [13] is commonly studied in machine learning. It is regarded as one of the top performing

methods for document classification [14][22]. Its basic idea is to estimate the parameters of a multinomial generative model for instances, then find the most probable class for a given instance using the Bayes' rule and the Naïve Bayes assumption that the features occur independently of each other inside a class.

The Decision Tree (DT) algorithm [13] is a method for approximating discrete valued target function, in which the learned function is represented by a tree of arbitrary degree that classifies instances. The C4.5 [15] software, a widely used implementation of the DT algorithm, is used in the following.

The Sparse Network of Winnows (SNoW) algorithm [16] is specifically tailored for learning in the presence of a very large number of features and can be used as a general purpose multi-class classifier. The learned classifier is a sparse network of linear functions. The SNoW software developed by the Cognitive Computation Group at UIUC [3] is used in the following.

3.5 Experiment Results

We have trained the above learning algorithms on 5 different size training datasets respectively and tested them on the TREC10 questions. The parameters for each of the learning algorithms (e.g., the C in the SVM) are all with their default values untouched. The question classification performance is measured by accuracy, i.e., the proportion of the correctly classified questions among all test questions.

Table 2. The question classification accuracy using different machine learning algorithms, with the bag-of-words features, under the coarse grained category definition.

algorithm	*1000*	*2000*	*3000*	*4000*	*5500*
NN	70.0%	73.6%	74.8%	74.8%	75.6%
NB	53.8%	60.4%	74.2%	76.0%	77.4%
DT	78.8%	79.8%	82.0%	83.4%	84.2%
SNoW	71.8%	73.4%	74.2%	78.2%	66.8%
SVM	76.8%	83.4%	87.2%	87.4%	85.8%

Table 3. The question classification accuracy using different machine learning algorithms, with the bag-of-ngrams features, under the coarse grained category definition.

algorithm	*1000*	*2000*	*3000*	*4000*	*5500*
NN	72.0%	81.0%	79.8%	80.8%	79.8%
NB	73.0%	79.2%	80.0%	81.8%	83.2%
DT	73.8%	82.6%	83.0%	84.6%	84.2%
SNoW	59.8%	85.2%	80.6%	87.0%	86.6%
SVM	77.6%	82.6%	84.0%	84.8%	87.4%

Table 4. The question classification accuracy using different machine learning algorithms, with the bag-of-words features, under the fine grained category definition.

algorithm	*1000*	*2000*	*3000*	*4000*	*5500*
NN	57.4%	62.8%	65.2%	67.2%	68.4%
NB	48.8%	52.8%	56.6%	56.2%	58.4%
DT	67.0%	70.0%	73.6%	75.4%	77.0%
SNoW	42.2%	66.2%	69.0%	66.6%	74.0%
SVM	68.0%	75.0%	77.2%	77.4%	80.2%

Table 5. The question classification accuracy using different machine learning algorithms, with the bag-of-ngrams features, under the fine grained category definition.

algorithm	*1000*	*2000*	*3000*	*4000*	*5500*
NN	59.4%	64.6%	67.2%	67.4%	68.6%
NB	54.4%	58.4%	63.0%	65.0%	67.8%
DT	62.8%	72.2%	72.6%	73.0%	77.0%
SNoW	44.0%	67.0%	75.0%	55.8%	75.8%
SVM	65.0%	74.0%	74.8%	77.4%	79.2%

Not surprisingly, classifiers trained on larger training dataset usually get better performance.

For the SVM algorithm, we observed that the bag-of-ngrams features are not much better than the bag-of-words features. Also, the SVM based on linear kernel turns out to be as good as the SVM based on polynomial kernel, RBF kernel or sigmoid kernel, so we choose to include only the accuracy of the linear SVM in the above tables.

We do not take into account the category hierarchy in the above experiments. It has been reported that the hierarchical SNoW classifier trained on 5,500 examples, with the bag-of-words features and under the fine grained category definition, can achieve an accuracy of 77.6% [12].

In summary, the experiment results show that with only surface text features the SVM outperforms the other four methods for this task.

4. A TREE KERNEL

We might be hitting a limit imposed by the representation of questions which ignores syntax, so including syntactic information might be helpful. For example, the two questions "*Which university did the president graduate from?*" and "*Which president is a graduate of the Harvard University?*" could be discriminated by their different syntactic structures.

In this section, we propose to use a special kernel function called tree kernel to enable the SVM to take advantage of the syntactic structures of questions.

4.1 Concept

A key property of the Support Vector Machines [2] is that the only operation it requires is the computation of dot products between pairs of examples. One may therefore replace the dot product with a Mercer kernel, implicitly mapping feature vectors in $\mathbb{R}^d$ into a new space $\mathbb{R}^m$, and applying the original algorithm in this new feature space. The kernel methods provide an efficient way to carry out such computation when m is large or even infinite.

Given a question, we first parse it into a syntactic tree using a parser like [4], and then we represent it as a vector in a high dimensional space which is defined over tree fragments.

The tree fragments of a syntactic tree are all its sub-trees which include at least one terminal symbol (word) or one production

rule, with the restriction that no production rules can be broken into incomplete parts.

Implicitly we enumerate all the possible tree fragments $1, 2, ..., m$. These tree fragments are the axis of this m dimensional space. Note that this could be done only implicitly, since the number m is extremely large.

Every syntactic tree T is represented by an m dimensional vector $\mathbf{v}(T) = \left(v_1(T), v_2(T), ..., v_m(T)\right)'$, where the i-th element $v_i(T)$ is the weight of the tree fragment in the i-th dimension if the tree fragment is in the tree T and zero otherwise. The tree kernel of two syntactic trees T_1 and T_2 is actually the inner product of $\mathbf{v}(T_1)$ and $\mathbf{v}(T_2)$.

For each specific tree fragment i contained in the syntactic tree T, i is weighted by $\sqrt{\lambda}^{s(i)} \sqrt{\mu}^{d(i)}$, where $s(i)$ is its size and λ is the weighting factor for size, $d(i)$ is its depth and μ is the weighting factor for depth. The size of a tree fragment is defined as the number of production rules it contains. The depth of a tree fragment is defined as the depth of the tree fragment root in the entire syntactic tree. We introduce the weighting factors for size and depth to reflect the different importance for different kinds of tree fragments.

A terminal node (a node labeled by word but not Part-of-Speech tag) itself is a legal tree fragment of size 0. So if we set the weighting factor for size $\lambda = 0$ and the weighting factor for depth $\mu = 1$, the tree kernel backs off to the word linear kernel.

Our tree kernel definition is similar to the tree kernel proposed in [1]. We propose a different definition because we found that using their definition without modification does not work well for question classification task. One advantage of our definition is that the tree kernel can back off to the word linear kernel, which guarantees that the tree kernel can work at least as well as the word linear kernel. Another advantage of our definition is that the tree kernel can weight the tree fragments according to their depth so as to identify the question focus. For example, the focus of the question "*Which university did the president graduate from?*" is "*university*" but not "*president*", since the depth of "*university*" is less than the depth of "*president*". Such information is helpful to correctly classify the question.

4.2 Sample

We use the question "What is an atom?" (TREC10 question No. 897) to illustrate the above concepts.

Figure 1(a) depicts the syntactic tree of the question, Figure 1(b) is one of its sub-trees, and Figure 1(c) contains all tree fragments of a within the extent of b.

Table 6 displays the size and depth of each tree fragment in Figure 1(c).

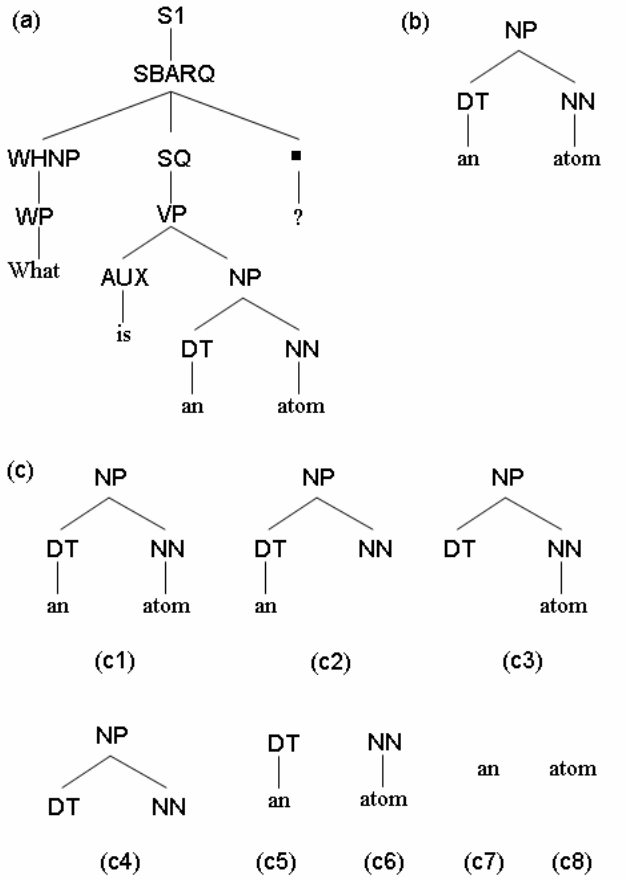

Figure 1. (a) The syntactic tree of the sample question. (b) One of the sub-trees of a. (c) All tree fragments of a within the extent of b.

Table 6. The size & depth of each tree fragment in Fig. 1(c).

tree fragment	size	depth
c1	3	4
c2	2	4
c3	2	4
c4	2	4
c5	1	5
c6	1	5
c7	0	6
c8	0	6

4.3 Computation

Note that the space of tree fragments is extremely large, i.e., the number of dimensions m is extremely large, which makes it intractable to explicitly represent the tree vectors in order to compute their inner products. So developing an efficient tree kernel computation algorithm whose complexity is independent of m is necessary. Here we describe how the tree kernel can be computed in polynomial time complexity by dynamic programming. This idea was proposed in [1] and we extend it to work with our definition of the tree kernel.

Consider a syntactic tree T. Let N represent the set of nodes in T. Define the binary indicator function $I_i(n)$ to be 1 if the tree fragment i is seen rooted at a node n and 0 otherwise. And represent the depth of the node n by $d(n)$. Then we can derive that $v_i(T) = \sum_{n \in N} \sqrt{\lambda}^{s(i)} \sqrt{\mu}^{d(i)} I_i(n) = \sum_{n \in N} \sqrt{\lambda}^{s(i)} \sqrt{\mu}^{d(n)} I_i(n)$.

Given two syntactic trees T_1 and T_2, the tree kernel $k(T_1, T_2)$ can be computed as follows.

$$k(T_1, T_2) = \mathbf{v}(T_1) \bullet \mathbf{v}(T_2) = \sum_i v_i(T_1) v_i(T_2)$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \sum_i \left(\left(\sqrt{\lambda}^{s(i)} \sqrt{\mu}^{d(n_1)} I_i(n_1) \right) \left(\sqrt{\lambda}^{s(i)} \sqrt{\mu}^{d(n_2)} I_i(n_2) \right) \right)$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \mu^{(d(n_1)+d(n_2))/2} \sum_i \lambda^{s(i)} I_i(n_1) I_i(n_2)$$

$$= \sum_{n_1 \in N_1} \sum_{n_2 \in N_2} \mu^{(d(n_1)+d(n_2))/2} C(n_1, n_2).$$

Here $C(n_1, n_2) = \sum_i \lambda^{s(i)} I_i(n_1) I_i(n_2)$.

The value of $C(n_1, n_2)$ can be computed recursively as follows.

- $C(n_1, n_2) = 0$, if $n_1 \neq n_2$;
- $C(n_1, n_2) = 1$, if $n_1 = n_2$ and n_1, n_2 are terminal nodes;
- $C(n_1, n_2) = \lambda$, if $n_1 = n_2$ and n_1, n_2 are pre-terminal nodes;
- $C(n_1, n_2) = \lambda \prod_{j=1}^{nc(n1)} \left(1 + C\left(ch(n_1, j), ch(n_2, j)\right)\right)$, if $n_1 = n_2$ and n_1, n_2 are neither terminal nor pre-terminal nodes.

Here $n_1 = n_2$ if and only if the labels of n_1 and n_2 are same and the labels of n_1's children and n_2's children are also same. That is to say, while n_1 and n_2 are terminal nodes, $n_1 = n_2$ if the word at n_1 and the word at n_2 are same, while n_1 and n_2 are not terminal nodes, $n_1 = n_2$ if the production rule at n_1 and the production rule at n_2 are same. The number of the node n's children is $nc(n)$, so if $n_1 = n_2$ then $nc(n_1) = nc(n_2)$. The j-th child of the node n is $ch(n, j)$.

According to the dynamic programming idea, we can traverse T_1 and T_2 in post-order, fill in a $|N_1| \times |N_2|$ matrix of $C(n_1, n_2)$, then sum up the matrix entries weighted by $\mu^{(d(n_1)+d(n_2))/2}$. The time complexity of this algorithm turns out to be at most $O(|N_1||N_2|)$.

4.4 Evaluation

The MEI parser [4] is used to get the syntactic tree for each question. Then the SVM based on tree kernel is applied. There are two parameters for the tree kernel: λ and μ. The default value of λ is set to 0, and the default value of μ is set to 1, so that the tree kernel with default parameter values is identical to the word linear kernel. To find appropriate parameter values, we run 4-fold cross validation on 1,000 training examples.

Table 7. Tuning λ for the tree kernel ($\mu = 1$) by 4-fold cross validation on 1,000 training examples, under the coarse grained category definition.

λ	0.0	**0.1**	0.2	0.3	0.4
accuracy	73.7%	74.9%	74.7%	74.7%	74.0%

Table 8. Tuning μ for the tree kernel ($\lambda = 0.1$) by 4-fold cross validation on 1,000 training examples, under the coarse grained category definition.

μ	1.0	**0.9**	0.8	0.7	0.6
accuracy	74.9%	76.6%	76.4%	73.4%	69.7%

Table 9. The question classification accuracy using SVM based on word linear kernel, ngram linear kernel, and tree kernel ($\lambda = 0.1$, $\mu = 0.9$), under the coarse grained category definition.

kernel	***1000***	***2000***	***3000***	***4000***	***5500***
word	76.8%	83.4%	87.2%	87.4%	85.8%
ngram	77.6%	82.6%	84.0%	84.8%	87.4%
tree	80.2%	86.2%	87.6%	88.6%	90.0%

Table 10. The confusion matrix of the tree kernel ($\lambda = 0.1$, $\mu = 0.9$) SVM trained on 5,500 examples, under the coarse grained category definition.

	ABBR	***DESC***	***ENTY***	***HUM***	***LOC***	***NUM***
ABBR	7	**2**	0	0	0	0
DESC	0	137	**1**	0	0	0
ENTY	0	**12**	73	**3**	6	0
HUM	0	**1**	**3**	61	0	0
LOC	0	**4**	**5**	**1**	70	**1**
NUM	0	**9**	**1**	0	**1**	102

Under the coarse grained category definition, the SVM based on tree kernel can bring about 20% errors reduction, as shown in Table 9. Table 10 presents the confusion matrix of the best classifier, i.e., the tree kernel ($\lambda = 0.1$, $\mu = 0.9$) SVM trained on 5,500 examples. For example, the 1st row of the confusion matrix indicates that on the 9 test questions in the ABBR category, 7 have been correctly classified and 2 have been misclassified into DESC category. The most common confusions

happen between ENTY and DESC, which is not surprising because the classifier are forced to do disjoint classification.

The statistical sign test can be applied for comparing two methods ***A*** and ***B*** [22]. Table 11 shows the significance test results, where method ***A*** is the SVM based on tree kernel ($\lambda = 0.1$, $\mu = 0.9$) and method ***B*** is the SVM based on word or ngram linear kernel, both are trained on 5,500 examples and tested on 500 examples, under the coarse grained category definition. The small *P*-value means that method ***A*** is significantly better than method ***B***.

Table 11. The significance test results for comparing the SVM based on tree kernel and the SVM based on word/ngram linear kernel.

A	***B***	**train**	**test**	n	k	Z	***P*-value**
tree	**word**	5,500	500	35	28	3.55	< 0.005
tree	**ngram**	5,500	500	41	27	2.03	< 0.025

Under the fine grained category definition, the SVM based on tree kernel brings slight improvements. The experiment results are omitted to save space. After carefully checking the misclassifications, we believe that the syntactic tree does not normally contain the information required to distinguish between the various fine categories within a course category. The inconsistent labeling of the questions under the fine grained category definition also makes learning and evaluation for the fine category difficult. Nevertheless, if we classify the questions into coarse grained categories first and then classify the questions inside each coarse grained category into its fine grained sub-categories, we may still expect better results using the SVM based on tree kernel. It would be interesting to extend the work using hierarchical SVM classifiers [6] in the future.

5. RELATED WORK

Most question answering systems [18][19][20] do question classification using handcrafted rules. Consequently there are only a few papers describing machine learning approaches to question classification. In [17], the machine learning tool Ripper has been utilized for question classification. They defined 17 question categories, trained their question classification system on labeled TREC8 and TREC9 dataset, and tested it on TREC10 dataset. The reported accuracy of their system is 70%, which is discouraging. In [12], the SNoW method has been utilized for question classification. However, the good performance of their system depends heavily on the features called "RelWords" (related words) which are constructed semi-automatically but not automatically. In contrast, our approach only uses automatically constructed features. Since syntactic structure and related words carry different information, it is possible that including related words will further improve the result of our tree kernel SVM approach.

The convolution kernels (string kernels, sequence kernels, tree kernels, etc.) recently emerged in bioinformatics area [8][21]. Such kernels have been successfully applied to some natural language processing tasks like parsing [1]. Today there are more research works on kernels for structured data, for example in [7].

The tree kernel SVM approach to question classification relies on a parser to get the syntactic trees. However, most off-the-shelf parsers, including the one we used in above experiments, are not targeted to parse questions. Those parsers are usually trained on the Penn Treebank corpus, which is composed of manually labeled newswire text. Therefore it's not surprising that those parsers can not achieve a high accuracy in parsing questions, because there are only very few questions in the training corpus. In [9], the accuracy of question parsing dramatically improves when complementing the Penn Treebank corpus with additional 1153 labeled questions for training. We believe that a better parser is beneficial to our approach.

6. CONCLUSION

This paper presents our research work on automatic question classification through machine learning approaches.

The main contributions of this paper are as follows.

(1) We show that with only surface text features SVM outperforms four other machine learning methods (NN, NB, DT, SNoW) for question classification.

(2) We found that the syntactic structures of questions are really helpful to question classification.

(3) We propose to use a special kernel function called tree kernel to enable the SVM to take advantage of the syntactic structures of questions. And we describe how the tree kernel can be computed efficiently by dynamic programming.

The main future direction of our research is to exploit semantic knowledge (such as WordNet) for question classification. It is also worth investigating other types of machine learning algorithms.

7. ACKNOWLEDGEMENTS

Many thanks to Dr. Thorsten Joachims for kindly being our SIGIR'03 mentor. Thanks also to the Cognitive Computation Group at UIUC for opening their datasets.

8. REFERENCES

[1] M. Collins and N. Duffy. Convolution Kernels for Natural Language. In *Proceedings of Neural Information Processing Systems (NIPS14)*, 2001.

[2] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, Cambridge, UK, 2000.

[3] A. Carlson, C. Cumby, J. Rosen, and D. Roth. SNoW User Guide. *Technical Report UIUCDCS-R-99-2101*. UIUC Computer Science Department, Aug 1999. Software available at http://l2r.cs.uiuc.edu/~danr/snow.html.

[4] E. Charniak. A Maximum-Entropy-Inspired Parser. *Technical Report CS-99-12*, Brown University, Computer Science Department, Aug 1999.

[5] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*. 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[6] S. Dumais and H. Chen. Hierarchical Classification of Web Content. In *Proceedings of the 23rd ACM Int. Conference on Research and Development in Information Retrieval (SIGIR'00)*, pp 256-263, Athens, GR, 2000.

[7] T. Gaertner, J. Lloyd, and P. Flach. Kernels for Structured Data. In *Proceedings of the 12th International Conference on Inductive Logic Programming (ILP)*, July 2002.

[8] D. Haussler. Convolution Kernels on Discrete Structures. *Technical Report*, University of Santa Cruz. 1999.

[9] U. Hermjakob. Parsing and Question Classification for Question Answering. In *Proceedings of the ACL Workshop on Open-Domain Question Answering*, Toulouse, France, 2001.

[10] E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. Towards Semantics-based Answer Pinpointing. In *Proceedings of the DARPA Human Language Technology conference (HLT)*, San Diego, CA, 1999.

[11] C. W. Hsu and C. J. Lin. A Comparison of Methods for Multi-class Support Vector Machines, *IEEE Transactions on Neural Networks*, 13, pp. 415--425, 2002.

[12] X. Li and D. Roth. Learning Question Classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, 2002.

[13] T. Mitchell. *Machine Learning*. McGraw Hill, New York, 1997.

[14] A. McCallum and K. Nigam. A Comparison of Event Models for Naïve Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[15] J. R. Quinlan. *C4.5: Programs fro Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1993.

[16] D. Roth. Learning to Resolve Natural Language Ambiguities: A Unified Approach. In *Proceedings of AAAI'98*. pp. 806-813, Madison, WI, USA, Jul 1998.

[17] D. R. Radev, W. Fan, H. Qi, H. Wu and A. Grewal. Probabilistic Question Answering from the Web. In *Proceedings of the 11th World Wide Web Conference (WWW2002)*, Hawaii, 2002.

[18] E. Voorhees. The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text Retrieval Conference (TREC8)*, pp. 77-82, NIST, Gaithersburg, MD, 1999.

[19] E. Voorhees. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, pp. 71-80, NIST, Gaithersburg, MD, 2000.

[20] E. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text Retrieval Conference (TREC10)*, pp. 157-165, NIST, Gaithersburg, MD, 2001.

[21] C. Watkins. Dynamic Alignment Kernels. In A. J. Smola, P. L. Bartlett, B. Schlkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pp. 39-50, MIT Press, 2000.

[22] Y. Yang and X. Liu. A Re-examination of Text Categorization Methods. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'99)*, pp. 42-49, 1999.