Semantic parsing with Structured SVM Ensemble Classification Models

Le-Minh Nguyen, Akira Shimazu, and **Xuan-Hieu Phan**
Japan Advanced Institute of Science and Technology (JAIST)
Asahidai 1-1, Nomi, Ishikawa, 923-1292 Japan
{nguyenml,shimazu,hieuxuan}@jaist.ac.jp

Abstract

We present a learning framework for structured support vector models in which boosting and bagging methods are used to construct ensemble models. We also propose a selection method which is based on a switching model among a set of outputs of individual classifiers when dealing with natural language parsing problems. The switching model uses subtrees mined from the corpus and a boosting-based algorithm to select the most appropriate output. The application of the proposed framework on the domain of semantic parsing shows advantages in comparison with the original large margin methods.

1 Introduction

Natural language semantic parsing is an interesting problem in NLP (Manning and Schutze, 1999) as it would very likely be part of any interesting NLP applications (Allen, 1995). For example, the necessary of semantic parsing for most of NLP application and the ability to map natural language to a formal query or command language is critical for developing more user-friendly interfaces.

Recent approaches have focused on using structured prediction for dealing with syntactic parsing (B. Taskar et. al., 2004) and text chunking problems (Lafferty et al, 2001). For semantic parsing, Zettlemoyer and Collins (2005) proposed a method for mapping a NL sentence to its logical form by structured classification using a log-linear model that represents a distribution over syntactic and semantic analyses conditioned on the input sentence. Taskar et al (B. Taskar et. al., 2004) present a discriminative approach to parsing inspired by the large-margin criterion underlying support vector machines in which the loss function is factorized analogous to the decoding process. Tsochantaridis et al (Tsochantaridis et al., 2004) propose a large-margin models based on SVMs for structured prediction (SSVM) in general and apply it for syntactic parsing problem so that the models can adapt to overlap features, kernels, and any loss functions.

Following the successes of the SSVM algorithm to structured prediction, in this paper we exploit the use of SSVM to the semantic parsing problem by modifying the loss function, feature representation, maximization algorithm in the original algorithm for structured outputs (Tsochantaridis et al., 2004).

Beside that, forming committees or ensembles of learned systems is known to improve accuracy and bagging and boosting are two popular ensemble methods that typically achieve better accuracy than a single classifier (Dietterich, 2000). This leads to employing ensemble learning models for SSVM is worth to investigate. The first problem of forming an ensemble learning for semantic parsing is how to obtain individual parsers with respect to the fact that each individual parser performs well enough as well as they make different types of errors. The second one is that of combining outputs from individual semantic parsers. The natural way is to use the majority voting strategy that the semantic tree with highest frequency among the outputs obtained by individual parsers is selected. However, it is not sure that the majority voting technique is effective for combining complex outputs such as a logical form structure. Thus, a better combination method for semantic tree output should be investigated.

To deal with these problems, we proposed an

Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 619–626,
Sydney, July 2006.

ensemble method which consists of learning and averaging phases in which the learning phases are either a boosting or a bagging model, and the averaging phase is based on a switching method on outputs obtained from all individual SSVMs. For the averaging phase, the switching model is used subtrees mined from the corpus and a boosting-based algorithm to select the most appropriate output.

Applications of SSVM ensemble in the semantic parsing problem show that the proposed SSVM ensemble is better than the SSVM in term of the F-measure score and accuracy measurements.

The rest of this paper are organized as follows: Section 2 gives some background about the structured support vector machine model for structured predictions and related works. Section 3 proposes our ensemble method for structured SVMs on the semantic parsing problem. Section 4 shows experimental results and Section 5 discusses the advantage of our methods and describes future works.

2 Backgrounds

2.1 Related Works

Zelle and Mooney initially proposed the empirically based method using a corpus of NL sentences and their formal representation for learning by inductive logic programming (Zelle, 1996). Several extensions for mapping a NL sentence to its logical form have been addressed by (Tang, 2003). Transforming a natural language sentence to a logical form was formulated as the task of determining a sequence of actions that would transform the given input sentence to a logical form (Tang, 2003). The main problem is how to learn a set of rules from the corpus using the ILP method. The advantage of the ILP method is that we do not need to design features for learning a set of rules from corpus. The disadvantage is that it is quite complex and slow to acquire parsers for mapping sentences to logical forms. Kate et al presented a method (Kate et al., 2005) that used transformation rules to transform NL sentences to logical forms. Those transformation rules are learnt using the corpus of sentences and their logical forms. This method is much simpler than the ILP method, while it can achieve comparable result on the CLANG (Coach Language) and Query corpus. The transformation based method has the condition that the formal language should be in the form of LR grammar.

Ge and Mooney also presented a statistical method (Ge and Mooney, 2005) by merging syntactic and semantic information. Their method relaxed the condition in (Kate et al., 2005) and achieved a state-of the art performance on the CLANG and query database corpus. However the distinction of this method in comparison with the method presented in (Kate et al., 2005) is that Ge and Mooney require training data to have SAPTs, while the transforation based method only needs the LR grammar for the formal language.

The work proposed by (Zettlemoyer and Collins, 2005) that maps a NL sentence to its logical form by structured classification, using a log-linear model that represents a distribution over syntactic and semantic analyses conditioned on the input sentence. This work is quite similar to our work in considering the structured classification problem. The difference is that we used the kernel based method instead of a log-linear model in order to utilize the advantage of handling a very large number of features by maximizing the margin in the learning process.

2.2 Structured Support Vector Models

Structured classification is the problem of predicting y from x in the case where y has a meaningful internal structure. Elements $y \in Y$ may be, for instance, sequences, strings, labelled trees, lattices, or graphs.

The approach we pursue is to learn a discriminant function $F : X \times Y \rightarrow R$ over $< input, output >$ pairs from which we can derive a prediction by maximizing F over the response variable for a specific given input x. Hence, the general form of our hypotheses f is

$$f(x; w) = \arg\max_{y \in Y} F(x; y; w)$$

where w denotes a parameter vector.

As the principle of the maximum-margin presented in (Vapnik, 1998), in the structured classification problem, (Tsochantaridis et al., 2004) proposed several maximum-margin optimization problems.

For convenience, we define

$$\delta\psi_i(y) \equiv \psi(x_i, y_i) - \psi(x_i, y)$$

where (x_i,y_i) is the training data.

The hard-margin optimization problem is:

$$\text{SVM}_0 : min_w \frac{1}{2} \|w\|^2 \quad (1)$$

$$\forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y) \rangle > 0 \quad (2)$$

where $\langle w, \delta\psi_i(y) \rangle$ is the linear combination of feature representation for input and output.

The soft-margin criterion was proposed (Tsochantaridis et al., 2004) in order to allow errors in the training set, by introducing slack variables.

$$\mathrm{SVM}_1 : \min \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i, \text{s.t.} \forall i, \xi_i \geq 0 \quad (3)$$

$$\forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y) \rangle \geq 1 - \xi_i \quad (4)$$

Alternatively, using a quadratic term $\frac{C}{2n} \sum_i \xi_i^2$ to penalize margin violations, we obtained SVM_2. Here $C > 0$ is a constant that control the trade-off between training error minimization and margin maximization.

To deal with problems in which $|Y|$ is very large, such as semantic parsing, (Tsochantaridis et al., 2004) proposed two approaches that generalize the formulation SVM_0 and SVM_1 to the cases of arbitrary loss function. The first approach is to re-scale the slack variables according to the loss incurred in each of the linear constraints.

$$\mathrm{SVM}^{\Delta_s} : \underbrace{\min}_{w,\xi} \frac{1}{2} \|w\|^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i, \text{s.t.} \forall i, \xi_i \geq 0 \quad (5)$$

$$\forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y) \rangle \geq \frac{1 - \xi_i}{\Delta(y_i, y)} \quad (6)$$

The second approach to include loss function is to re-scale the margin as a special case of the Hamming loss. The margin constraints in this setting take the following form:

$$\forall i, \forall y \in Y \backslash y_i : \langle w, \delta\psi_i(y) \rangle \geq \Delta(y_i, y) - \xi_i \quad (7)$$

This set of constraints yields an optimization problem, namely $\mathrm{SVM}_1^{\Delta_m}$.

2.3 Support Vector Machine Learning

The support vector learning algorithm aims to find a small set of active constraints that ensures a sufficiently accurate solution. The detailed algorithm, as presented in (Tsochantaridis et al., 2004) can be applied to all SVM formulations mentioned above. The only difference between them is the cost function in the following optimization problems:

$$\mathrm{SVM}_1^{\Delta_s} : H(y) \equiv (1 - \langle \delta\psi_i(y), w \rangle)\Delta(y_i, y)$$
$$\mathrm{SVM}_2^{\Delta_s} : H(y) \equiv (1 - \langle \delta\psi_i(y), w \rangle)\sqrt{\Delta(y_i, y)}$$
$$\mathrm{SVM}_1^{\Delta_m} : H(y) \equiv (\Delta(y_i, y) - \langle \delta\psi_i(y), w \rangle)$$
$$\mathrm{SVM}_2^{\Delta_m} : H(y) \equiv (\sqrt{\Delta(y_i, y)} - \langle \delta\psi_i(y), w \rangle)$$

Typically, the way to apply structured SVM is to implement feature mapping $\psi(x, y)$, the loss function $\Delta(y_i, y)$, as well as the maximization algorithm. In the following section, we apply a structured support vector machine to the problem of semantic parsing in which the mapping function, the maximization algorithm, and the loss function are introduced.

3 SSVM Ensemble for Semantic Parsing

Although the bagging and boosting techniques have known to be effective for improving the performance of syntactic parsing (Henderson and Brill, 2000), in this section we focus on our ensemble learning of SSVM for semantic parsing and propose a new effective switching model for either bagging or boosting model.

3.1 SSVM for Semantic Parsing

As discussed in (Tsochantaridis et al., 2004), the major problem for using the SSVM is to implement the feature mapping $\psi(x, y)$, the loss function $\Delta(y_i, y)$, as well as the maximization algorithm. For semantic parsing, we describe here the method of structure representation, the feature mapping, the loss function, and the maximization algorithm.

3.1.1 Structure representation

A tree structure representation incorporated with semantic and syntactic information is named semantically augmented parse tree (SAPT) (Ge and Mooney, 2005). As defined in (Ge and Mooney, 2005), in an SAPT, each internal node in the parse tree is annotated with a semantic label. Figure 1 shows the SAPT for a simple sentence in the CLANG domain. The semantic labels which are shown after dashes are concepts in the domain. Some concepts refer to predicates and take an ordered list of arguments. Concepts such as "team" and "unum" might not have arguments. A special semantic label, "null", is used for a node that does not correspond to any concept in the domain.

3.1.2 Feature mapping

For semantic parsing, we can choose a mapping function to get a model that is isomorphic to a probabilistic grammar in which each rule within the grammar consists of both a syntactic rule and a semantic rule. Each node in a parse tree y for a sentence x corresponds to a grammar rule g_j with a score w_j.

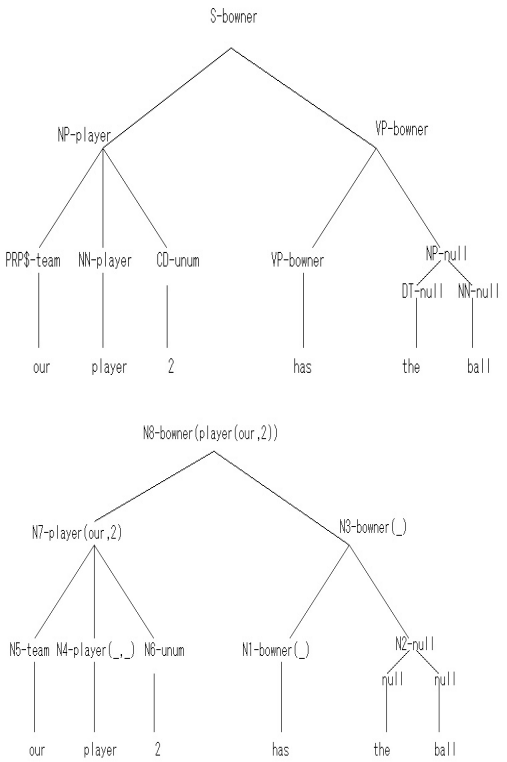

Figure 1: An Example of tree representation in SAPT

All valid parse trees y for a sentence x are scored by the sum of the w_j of their nodes, and the feature mapping $\psi(x, y)$ is a history gram vector counting how often each grammar rule g_j occurs in the tree y. Note that the grammar rules are lexicalized. The example shown in Figure 2 clearly explains the way features are mapped from an input sentence and a tree structure.

3.1.3 Loss function

Let z and z_i be two semantic tree outputs and $|z_i|$ and $|z_i|$ be the number of brackets in z and z_i, respectively. Let n be the number of common brackets in the two trees. The loss function between z_i and z is computed as bellow.

$$F - loss(z_i, z) = 1 - \frac{2 \times n}{|z_i| + |z|} \quad (8)$$

$$\text{zero} - \text{one}(z_i, z) = \begin{cases} 1 \text{ if } z_i \neq z \\ 0 \text{ otherwise} \end{cases} \quad (9)$$

3.1.4 Maximization algorithm

Note that the learning function can be efficiently computed by finding a structure $y \in Y$ that maximizes $F(x, y; w)$=$\langle w, \delta\psi_i(y)\rangle$ via a maximization algorithm. Typically we used a variant of

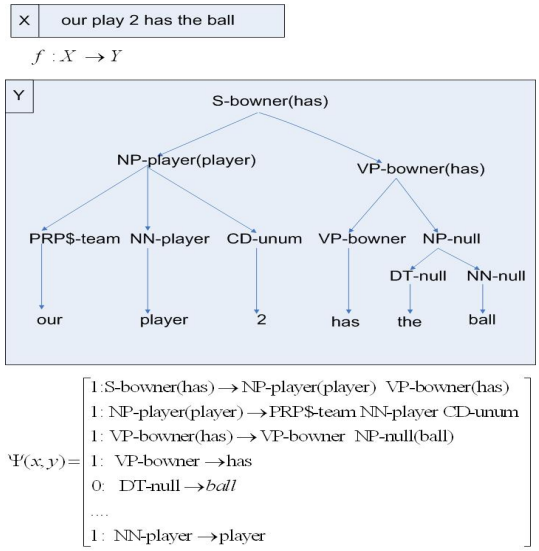

Figure 2: Example of feature mapping using tree representation

CYK maximization algorithm which is similar to the one for the syntactic parsing problem (Johnson,1999). There are two phases in our maximization algorithm for semantic parsing. The first is to use a variant of CYK algorithm to generate a SAPT tree. The second phase then applies a deterministic algorithm to output a logical form. The score of the maximization algorithm is the same with the obtained value of the CYK algorithm.

The procedure of generating a logical form using a SAPT structure originally proposed by (Ge and Mooney, 2005) and it is expressed as Algorithm 1. It generates a logical form based on a knowledge database K for given input node N. The predicate argument knowledge, K, specifies, for each predicate, the semantic constraints on its arguments. Constraints are specified in terms of the concepts that can fill each argument, such as player(team, unum) and bowner(player).

The GETsemanticHEAD determines which of node's children is its semantic head based on they having matching semantic labels. For example, in Figure 1 $N3$ is determined to be the semantic head of the sentence since it matches N8's semantic label. ComposeMR assigns their meaning representation (MR) to fill the arguments in the head's MR to construct the complete MR for the node. Figure 1 shows an example of using BuildMR to generate a semantic tree to a logical form.

Input: The root node N of a SAPT
Predicate knowledge K
Notation: X_{MR} is the MR of node X
Output: N_{MR}
Begin
C_i= the ith child node of N
C_h= GETsemanticHEAD(N)
$C_{h_{MR}}$ =BuildMR(C_h, K)
for *each other child* C_i *where* $i \neq h$ **do**
  $C_{i_{MR}}$ =BuildMR(C_i, K)
  ComposeMR($C_{h_{MR}}$,$C_{i_{MR}}, K$)
end
N_{MR}=$C_{h_{MR}}$
End

Algorithm 1: BuildMR(N, K): Computing a logical form form an SAPT(Ge and Mooney, 2005)

Input: $S = (x_i; y_i; z_i), i = 1, 2, ..., l$ in which x_i is the sentence and y_i, z_i is the pair of tree structure and its logical form
Output: SSVM model
repeat
  for $i = 1$ *to* n **do**

$\text{SVM}_1^{\Delta_s} : H(y, z) \equiv (1 - \langle \delta\psi_i(y), w \rangle)\Delta(z_i, z)$
$\text{SVM}_2^{\Delta_s} : H(y, z) \equiv (1 - \langle \delta\psi_i(y), w \rangle)\sqrt{\Delta(z_i, z)}$
$\text{SVM}_1^{\Delta_m} : H(y, z) \equiv (\Delta(z_i, z) - \langle \delta\psi_i(y), w \rangle)$
$\text{SVM}_2^{\Delta_m} : H(y, z) \equiv (\sqrt{\Delta(z_i, z)} - \langle \delta\psi_i(y), w \rangle)$

    compute $< y^*, z^* >= \arg\max_{y,z \in Y,Z} H(Y, Z)$;
    compute $\xi_i = \max\{0, \max_{y,z \in S_i} H(y, z)\}$;
    if $H(y^*, z^*) > \xi_i + \varepsilon$ **then**
      $S_i \leftarrow S_i \cup y^*, z^*$;
      solving optimization with SVM;
    end
  end
until *no* S_i *has changed during iteration*;

Algorithm 2: Algorithm of SSVM learning for semantic parsing. The algorithm is based on the original algorithm(Tsochantaridis et al., 2004)

3.1.5 SSVM learning for semantic parsing

As mentioned above, the proposed maximization algorithm includes two parts: the first is to parse the given input sentence to the SAPT tree and the second part (BuildMR) is to convert the SAPT tree to a logical form. Here, the score of maximization algorithm is the same with the score to generate a SAPT tree and the loss function should be the measurement based on two logical form outputs. Algorithm 2 shows our generation of SSVM learning for the semantic parsing problem which the loss function is based on the score of two logical form outputs.

3.2 SSVM Ensemble for semantic parsing

The structured SVM ensemble consists of a training and a testing phase. In the training phase, each individual SSVM is trained independently by its own replicated training data set via a bootstrap method. In the testing phase, a test example is applied to all SSVMs simultaneously and a collective decision is obtained based on an aggregation strategy.

3.2.1 Bagging for semantic parsing

The bagging method (Breiman, 1996) is simply created K bootstrap with sampling m items from the training data of sentences and their logical forms with replacement. We then applied the SSVM learning in the K generated training data to create K semantic parser. In the testing phase, a given input sentence is parsed by K semantic parsers and their outputs are applied a switching model to obtain an output for the SSVM ensemble parser.

3.2.2 Boosting for semantic parsing

The representative boosting algorithm is the AdaBoost algorithm (Schapire, 1999). Each SSVM is trained using a different training set. Assuming that we have a training set $TR = (x_i; y_i)|i = 1, 2, ..., l$ consisting of l samples and each sample in the TR is assigned to have the same value of weight $p_0(x_i) = 1/l$. For training the kth SSVM classifier, we build a set of training samples

$TR_{boost_k} = (x_i; y_i)|i = 1, 2, .., l'$ that is obtained by selecting $l'(< l)$ samples among the whole data set TR according to the weight value $p_{k-1}(x_i)$ at the (k-1)th iteration. This training samples is used for training the kth SSVM classifier. Then, we obtained the updated weight values $p_k(x_i)$ of the training samples as follows. The weight values of the incorrectly classified samples are increased but the weight values of the correctly classified samples are decreased. This shows that the samples which are hard to classify are selected more frequently. These updated weight values will be used for building the training samples $TR_{boost_{k+1}} = (x_i; y_i)|i = 1, 2, ..., l'$ of the (k+1)th SSVM classifier. The sampling procedure will be repeated until K training samples set has been built for the Kth SSVM classifier.

3.2.3 The proposed SSVM ensemble model

We construct a SSVM ensemble model by using different parameters for each individual SSVM together with bagging and boosting models. The parameters we used here including the kernel function and the loss function as well as features used in a SSVM. Let N and K be the number of different parameters and individual semantic parsers in a SSVM ensemble, respectively. The motivation is to create individual parsers with respect to the fact that each individual parser performs well enough as well as they make different types of errors. We firstly create N ensemble models using either boosting or bagging models to obtain $N \times K$ individual parsers. We then select the top T parsers so that their errors on the training data are minimized and in different types. After forming an ensemble model of SSVMs, we need a process for aggregating outputs of individual SSVM classifiers. Intuitively, a simplest way is to use a voting method to select the output of a SSVM ensemble. Instead, we propose a switching method using subtrees mining from the set of trees as follows.

Let $t_1, t_2, ..., t_K$ be a set of candidate parse trees produced by an ensemble of K parsers. From the set of tree $t_1, t_2, ..., t_K$ we generated a set of training data that maps a tree to a label +1 or -1, where the tree t_j received the label +1 if it is an corrected output. Otherwise t_j received the label -1. We need to define a learning function for classifying a tree structure to two labels +1 and -1.

For this problem, we can apply a boosting technique presented in (Kudo and Matsumoto, 2004). The method is based on a generation of Adaboost (Schapire, 1999) in which subtrees mined from the training data are severed as weak decision stump functions.

The technique for mining these subtrees is presented in (Zaki, 2002) which is an efficient method for mining a large corpus of trees. Table 1 shows an example of mining subtrees on our corpus. One

Table 1: Subtrees mined from the corpus

Frequency	Subtree
20	(and(bowner)(bpos))
4	(and(bowner)(bpos(right)))
4	(bpos(circle(pt(playerour11))))
15	(and(bpos)(not(bpos)))
8	(and(bpos(penalty-areaour)))

problem for using the boosting subtrees algorithm (BT) in our switching models is that we might obtain several outputs with label +1. To solve this, we evaluate a score for each value +1 obtained by the BT and select the output with the highest score. In the case of there is no tree output received the value +1, the output of the first individual semantic parser will be the value of our switching model.

4 Experimental Results

For the purpose of testing our SSVM ensembles on semantic parsing, we used the CLANG corpus which is the RoboCup Coach Language (www.robocup.org). In the Coach Competition, teams of agents compete on a simulated soccer field and receive advice from a team coach in a formal language. The CLANG consists of 37 non-terminal and 133 productions; the corpus for CLANG includes 300 sentences and their structured representation in SAPT (Kate et al., 2005), then the logical form representations were built from the trees. Table 2 shows the statistic on the CLANG corpus.

Table 2: Statistics on CLANG corpus. The average length of an NL sentence in the CLANG corpus is 22.52 words. This indicates that CLANG is the hard corpus. The average length of the MRs is also large in the CLANG corpus.

Statistic	CLANG
No.of. Examples	300
Avg. NL sentence length	22.5
Avg. MR length (tokens)	13.42
No. of non-terminals	16
No. of productions	102
No. of unique NL tokens	337

Table 3: Training accuracy on CLANG corpus

Parameter	Training Accuracy
linear+F-loss(Δ_s)	83.9%
polynomial(d=2)+F-loss (Δ_m)	90.1%
polynomial(d=2)+F-loss(Δ_s)	98.8%
polynomial(d=2)+F-loss(Δ_m)	90.2%
RBF+F-loss(Δ_s)	86.3%

To create an ensemble learning with SSVM, we used the following parameters with the linear kernel, the polynomial kernel, and RBF kernel, respectively. Table 3 shows that they obtained different accuracies on the training corpus, and their accuracies are good enough to form a SSVM ensemble. The parameters in Table 3 is used to form our proposed SSVM model.

The following is the performance of the SSVM[1], the boosting model, the bagging model, and the models with different parameters on the

[1]The SSVM is obtained via http://svmlight.joachims.org/

CLANG corpus[2]. Note that the numbers of individual SSVMs in our ensemble models are set to 10 for boosting and bagging, and each individual SSVM can be used the zero-one and F1 loss function. In addition, we also compare the performance of the proposed ensemble SSVM models and the conventional ensemble models to assert that our models are more effective in forming SSVM ensemble learning.

We used the standard 10-fold cross validation test for evaluating the methods. To train a BT model for the switching phase in each fold test, we separated the training data into 10-folds. We keep 9/10 for forming a SSVM ensemble, and 1/10 for producing training data for the switching model. In addition, we mined a subset of subtrees in which a frequency of each subtree is greater than 2, and used them as weak functions for the boosting tree model. Note that in testing the whole training data in each fold is formed a SSVM ensemble model to use the BT model estimated above for selecting outputs obtained by the SSVM ensemble.

To evaluate the proposed methods in parsing NL sentences to logical form, we measured the number of test sentences that produced complete logical forms, and the number of those logical forms that were correct. For CLANG, a logical form is correct if it exactly matches the correct representation, up to reordering of the arguments of commutative operators. We used the evaluation method presented in (Kate et al., 2005) as the formula below.

$$precision = \frac{\#\text{correct}-\text{representation}}{\#\text{completed}-\text{representation}}$$
$$recall = \frac{\#\text{correct}-\text{representation}}{\#\text{sentences}}$$

Table 4 shows the results of SSVM, the SCSISSOR system (Ge and Mooney, 2005), and the SILT system (Kate et al., 2005) on the CLANG corpus, respectively. It shows that SCSISSOR obtained approximately 89% precision and 72.3% recall while on the same corpus our best single SSVM method [3] achieved a recall (74.3%) and lower precision (84.2%). The SILT system achieved approximately 83.9% precision and 51.3% recall [4] which is lower than the best single SSVM.

[2]We set N to 5 and K to 6 for the proposed SSVM.

[3]The parameter for SSVM is the **polynomial(d=2)+(Δ_s)**

[4]Those figures for precision and recall described in (Kate et al., 2005) showed approximately this precision and recall of their method in this paper

Table 4: Experiment results with CLANG corpus. Each SSVM ensemble consists of 10 individual SSVM. SSVM bagging and SSVM boosting used the voting method. P-SSVM boosting and P-SSVM bagging used the switching method (BT) and voting method (VT).

System	Methods	Precision	Recall
1	SSVM	84.2%	74.3%
1	SCSISSOR	89.0%	72.3%
1	SILT	83.9%	51.3%
10	SSVM Bagging	85.7%	72.4%
10	SSVM Boosting	85.7%	72.4%
10	**P-SSVM Boosting(BT)**	**88.4%**	**79.3%**
10	**P-SSVM Bagging(BT)**	**86.5%**	**79.3%**
10	P-SSVM Boosting(VT)	86.5%	75.8%
10	P-SSVM Bagging(VT)	84.6%	75.8%

Table 4 also shows the performance of Bagging, Boosting, and the proposed SSVM ensemble models with bagging and boosting models. It is important to note that the switching model using a boosting tree method (BT) to learn the outputs of individual SSVMs within the SSVM ensemble model.

It clearly indicates that our proposed ensemble method can enhance the performance of the SSVM model and the proposed methods are more effective than the conventional ensemble method for SSVM. This was because the output of each SSVM is complex (i.e a logical form) so it is not sure that the voting method can select a corrected output. In other words, the boosting tree algorithms can utilize subtrees mined from the corpus to estimate the good weight values for subtrees, and then combines them to determine whether or not a tree is selected. In our opinion, with the boosting tree algorithm we can have a chance to obtain more accurate outputs. These results in Table 4 effectively support for this evidence.

Moreover, Table 4 depicts that the proposed ensemble method using different parameters for either bagging and boosting models can effectively improve the performance of bagging and boosting in term of precision and recall. This was because the accuracy of each individual parser in the model with different parameters is better than each one in either the boosting or the bagging model. In addition, when performing SSVM on the test set, we might obtain some 'NULL' outputs since the grammar generated by SSVM could not derive this sentence. Forming a number of individual SSVMs to an ensemble model is the way to handle this case, but it could make the numbers of completed outputs and corrected outputs increase. Ta-

ble 4 indicates that the proposed SSVM ensemble model obtained 88.4% precision and 79.3% recall. Therefore it shows substantially a better F1 score in comparison with previous work on the CLANG corpus.

Summarily, our method achieved the best recall result and a high precision on CLANG corpus. The proposed ensemble models outperformed the original SSVM on CLANG corpus and its performances also is better than that of the best published result.

5 Conclusions

This paper presents a structured support vector machine ensemble model for semantic parsing problem by employing it on the corpus of sentences and their representation in logical form.

We also draw a novel SSVM ensemble model in which the forming ensemble strategy is based on a selection method on various parameters of SSVM, and the aggregation method is based on a switching model using subtrees mined from the outputs of a SSVM ensemble model.

Experimental results show substantially that the proposed ensemble model is better than the conventional ensemble models for SSVM. It can also effectively improve the performance in term of precision and recall in comparison with previous works.

Acknowledgments

The work on this paper was supported by a Monbukagakusho 21st COE Program.

References

J. Allen. 1995. Natural Language Understanding (2nd Edition). *Mento Park, CA: Benjaming/Cumming*.

L. Breiman. 1996. Bagging predictors. *Machine Learning* 24, 123-140.

T.G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning 40* (2) 139-158.

M. Johnson 1999. PCFG models of linguistic tree representation. *Computational Linguistics.*

R. Ge and R.J. Mooney. 2005. A Statistical Semantic Parser that Integrates Syntax and Semanics. *In proceedings of CONLL 2005.*

J.C. Henderson and E. Brill 2000. Bagging and Boosting a Treebank Parser. *In proceedings ANLP 2000*: 34-41

R.J. Kate et al. 2005. Learning to Transform Natural to Formal Languages. *Proceedings of AAAI 2005*, page 825-830.

T. Kudo, Y. Matsumoto. A Boosting Algorithm for Classification of Semi-Structured Text. *In proceeding EMNLP 2004*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *In Proc. of ICML 2001*.

D.C. Manning and H. Schutze. 1999. Foundation of Statistical Natural Language Processing. *Cambridge, MA: MIT Press.*

L.S. Zettlemoyer and M. Collins. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. *In Proceedings of UAI*, pages 825–830.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. 2004. Support Vector Machine Learning for Interdependent and Structured Output Spaces. *In proceedings ICML 2004.*

V. Vapnik. 1995. *The Nature of Statistical Learning Theory. Springer*, N.Y., 1995.

L.R. Tang. 2003. Integrating Top-down and Bottom-up Approaches in Inductive Logic Programming: Applications in Natural Language Processing and Relation Data Mining. *Ph.D. Dissertation*, University of Texas, Austin, TX, 2003.

B. Taskar, D. Klein, M. Collins, D. Koller, and C.D. Manning. 2004. Max-Margin Parsing. *In proceedings of EMNLP, 2004*.

R.E. Schapire. 1999. A brief introduction to boosting. *Proceedings of IJCAI 99*

M.J. Zaki. 2002. Efficiently Mining Frequent Trees in a Forest. *In proceedings 8th ACM SIGKDD 2002.*

J.M. Zelle and R.J. Mooney. 1996. Learning to parse database queries using inductive logic programming. *In Proceedings AAAI-96, 1050-1055.*