Mixing Weak Learners in Semantic Parsing

Rodney D. Nielsen
Dept of Computer Science
University of Colorado, UCB-430
Boulder, CO 80309-0430
USA
Rodney.Nielsen@Colorado.edu

Sameer Pradhan
Center for Spoken Language Research
University of Colorado
Boulder, CO 80303
USA
Sameer.Pradhan@Colorado.edu

Abstract

We apply a novel variant of Random Forests (Breiman, 2001) to the shallow semantic parsing problem and show extremely promising results. The final system has a semantic role classification accuracy of 88.3% using PropBank gold-standard parses. These results are better than all others published except those of the Support Vector Machine (SVM) approach implemented by Pradhan et al. (2003) and Random Forests have numerous advantages over SVMs including simplicity, faster training and classification, easier multi-class classification, and easier problem-specific customization. We also present new features which result in a 1.1% gain in classification accuracy and describe a technique that results in a 97% reduction in the feature space with no significant degradation in accuracy.

1 Introduction

Shallow semantic parsing is the process of finding sentence constituents that play a semantic role relative to a target predicate and then labeling those constituents according to their respective roles. Specifying an event's agent, patient, location, time of occurrence, etc, can be useful for NLP tasks such as information extraction (c.f., Surdeanu et al., 2003), dialog understanding, question answering, text summarization, and machine translation. Example 1 depicts a semantic parse.

(1) [$_{Agent}$ She] [$_{P}$ bought] [$_{Patient}$ the vase] [$_{Locative}$ in Egypt]

We expand on previous semantic parsing work (Gildea and Jurafsky, 2002; Pradhan et al., 2003; Surdeanu et al., 2003) by presenting a novel algorithm worthy of further exploration, describing a technique to drastically reduce feature space size, and presenting statistically significant new features. The accuracy of the final system is 88.3% on the classification task using the PropBank (Kingsbury et al., 2002) corpus. This is just 0.6% off the best accuracy reported in the literature.

The classification algorithm used here is a variant of Random Forests (RFs) (Breiman, 2001). This was motivated by Breiman's empirical studies of numerous datasets showing that RFs often have lower generalize error than AdaBoost (Freund and Schapire, 1997), are less sensitive to noise in the training data, and learn well from weak inputs, while taking much less time to train. RFs are also simpler to understand and implement than SVMs, leading to, among other things, easier interpretation of feature importance and interactions (c.f., Breiman, 2004), easier multi-class classification (requiring only a single training session versus one for each class), and easier problem-specific customization (e.g., by introducing prior knowledge). The algorithm described here is considerably different from those in (Breiman, 2001). It was significantly revised to better handle high dimensional categorical inputs and as a result provides much better accuracy on the shallow semantic parsing problem.

The experiments reported here focus on the classification task – given a parsed constituent known to play a semantic role relative to a given predicate, decide which role is the appropriate one to assign to that constituent. Gold-standard sentence parses for test and training are taken from the PropBank dataset. We report results on two feature sets from the literature and a new feature set described here.

In section 2, we describe the data used in the experiments. Section 3 details the classification algorithm. Section 4 presents the experimental results and describes each experiment's feature set. Section 5 provides a discussion and thoughts on future work.

2 The Data

The classifiers were trained on data derived from the PropBank corpus (Kingsbury et al., 2002). The same observations and features are used as described by (Pradhan et al., 2003). They acquired the original data from the July 15, 2002 release of PropBank, which the University of Pennsylvania created by manually labeling the constituents

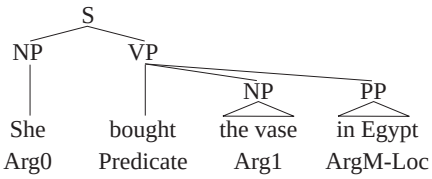

Figure 1: Syntactic parse of the sentence in (2)

Section	# sent	# words	# preds	# args
training	28	651	50	129
development	1.2	28	2.2	5.7
test	1.5	33	2.7	7.0

Table 1: Number of sentences, words, marked predicates, and labeled arguments in thousands

of the Penn TreeBank gold-standard parses (Marcus et al., 1994). Predicate usages (at present, strictly verbs) are hand annotated with 22 possible semantic roles plus the null role to indicate grammatical constituents that are not arguments of the predicate. The argument labels can have different meanings depending on their target predicate, but the annotation method attempted to assign consistent meanings to labels, especially when associated with similar verbs. There are seven core roles or arguments, labeled ARG0-5 and ARG9. ARG0 usually corresponds to the semantic agent and ARG1 to the entity most affected by the action. In addition to the core arguments, there are 15 adjunctive arguments, such as ARGM-LOC which identifies locatives. Thus our previous example, "She bought the vase in Egypt", would be parsed as shown in example 2. Figure 1 shows the associated syntactic parse without the parts of speech.

(2) [$_{Arg0}$ She] [$_{P}$ bought] [$_{Arg1}$ the vase] [$_{ArgM\text{-}Loc}$ in Egypt]

Development tuning is based on PropBank section 00 and final results are reported for section 23. We trained and tested on the same subset of observations as did Pradhan et al. (2003). They indicated that a small number of sentences (less than 1%) were discarded due to manual tagging errors in the original PropBank labeling process, (e.g., an empty role tag). This one percent reduction applies to all sections of the corpus (training, development and test). They removed an additional 2% of the training data due to issues involving the named entity tagger splitting corpus tokens into multiple words. However, where these issues occurred in tagging the section 23 test sentences, they were manually corrected. The size of the dataset is shown in Table 1.

3 The Algorithm

3.1 Random Forests

Breiman (2001) defines a random forest as "a classifier consisting of a collection of tree structured classifiers $\{h(\mathbf{x}, \mathbf{\Theta}_k), k{=}1, ...\}$ where the $\{\mathbf{\Theta}_k\}$ are independently identically distributed random [training] vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$." Thus Bagging (Breiman, 1996) is a form of Random Forest, where each tree is grown based on the selection, with replacement, of N random training examples, where N is the number of total examples in the training set.

Breiman (2001) describes two new subclasses of Random Forests, Forest-RI and Forest-RC. In each, he combines Bagging, using the CART methodology to create trees, with random feature selection (Amit and Geman, 1997) at each node in the tree. That is, at each node he selects a different random subset of the input features and considers only these in establishing the decision at that node.

The big idea behind Random Forests is that by injecting randomness into the individual trees via random feature selection, the correlation between their classification results is minimized. A lower correlation combined with reasonably good classification accuracy for individual trees leads to a much higher accuracy for the composite forest. In fact, Breiman shows that a theoretical upper bound can be established for the generalization error in terms of the strength of the forest, s, and the mean value of the classification correlation from individual trees, $\bar{\rho}$. The strength, s, is the expected margin over the input space, where the margin of an ensemble classifier is defined as the difference between the fraction of the ensemble members that vote for the correct class versus the fraction voting for the most popular alternative class. See (Breiman, 2001) for a detailed description of s and $\bar{\rho}$ and how they are calculated. The upper bound on the generalization error is given by the following equation:

$$E^* \leq \bar{\rho}\frac{(1-s^2)}{s^2} \quad (1)$$

Breiman found that Forest-RI and Forest-RC compare favorably to AdaBoost in general, are far less sensitive to noise in the training data, and can learn well using weak inputs.

3.2 Feature Issues

Before describing the variant of Random Forests we use here, it is helpful to discuss a couple of impor-

tant issues related to the input features. In the experiments here, the true input features to the algorithm are all categorical. Breiman's approach to handling categorical inputs is as follows. He modifies their selection probability such that they are V-1 times as likely as a numeric input to be selected for evaluation at each node, where V is the number of values the categorical feature can take. Then when a categorical input is selected he randomly chooses a subset of the category values and converts the input into a binary-valued feature whose value is one if the training observation's corresponding input value is in the chosen subset and zero otherwise.

In many machine learning approaches, a categorical feature having V different values would be converted to V (or V-1) separate binary-valued features (e.g., this is the case with SVMs). Here, we process them as categorical features, but conceptually think of them as separate binary-valued features. In an attempt to minimize confusion, we will refer to the categorical input features simply as *inputs* or as *input features*, the equivalent set of binary-valued features as the *binary-valued features*, and the features that are randomly composed in the tree building process (via random category value subset selection) as *composed features*.

3.3 Algorithm Description

Take any tree building algorithm (e.g., C5.0 (Quinlan, 2002)) and modify it such that instead of examining all of the input features at each node, it considers only a random subset of those features. Construct a large number of trees using all of the training data (we build 128 trees in each experiment). Finally, allow the trees to individually cast unit votes for each test observation. The majority vote determines the classification and ties are broken in favor of the class that occurs most frequently in the training set.

Our implementation is the most similar to Forest-RI, but has several differences, some significant. These differences involve not using Bagging, the use of a single forest rather than two competing forests, the assumed size of $\hat{V}_i$ (the number of relevant values for input i), the probability of selecting individual inputs, how composed features are created, and the underlying tree building algorithm. We delineate each of these differences in the following paragraphs.

Forest-RI combines random feature selection with Bagging. Surprisingly, we found that, in our experiments, the use of Bagging was actually hurting the classification accuracy of the forests and so we removed this feature from the algorithm. This means that we use all training observations to construct each tree in the forest. This is somewhat counter-intuitive given that it should increase correlation in the outputs of the trees. However, the strength of the forest is based in part on the accuracy of its trees, which will increase when utilizing more training data. We also hypothesize that, given the feature sets here, the correlation isn't affected significantly by the removal of Bagging. The reason for this is the massive number of binary-valued features in the problem (577,710 in just the baseline feature set). Given this fact, using random feature selection alone might result in substantially uncorrelated trees. As seen in equation 1 and shown empirically in (Breiman, 2001), the lack of correlation produced by random feature selection directly improves the error bound.

Forest-RI involves growing two forests and selecting the one most likely to provide the best results. These two forests are constructed using different values for F, the number of random features evaluated at each node. The choice of which forest is more likely to provide the best results is based on estimates using the observations not included in the training data (the out-of-bag observations). Since we did not use Bagging, all of our observations are used in the training of each tree and we could not take this approach. Additionally, it is not clear that this provided better results in (Breiman, 2001) and preliminary experiments (not reported here) suggest that it might be more effective to simply find a good value for F.

To create composed features, we randomly select a number of the input's category values, C, given by the following equation:

$$\begin{array}{ll} C = 1, & \hat{V} \leq 4 \\ C = \lfloor 1.5 + \log_2 \hat{V} \rfloor, & \hat{V} > 4 \end{array} \quad (2)$$

where $\hat{V}$ is the number of category values still potentially relevant. Random category value selection is consistent with Breiman's work, as noted in section 3.2. This random selection method should act to further reduce the correlation between trees and Breiman notes that it gets around the problem caused by categorical inputs with large numbers of values. However, he leaves the number of values chosen unspecified. There is also no indication of what to do as the categorical input becomes more sparse near the leaves of the tree (e.g., if the algorithm sends every constituent whose head word is in a set Φ down the right branch of the node, what effect does this have on future random value selection in each branch). This is the role of $\hat{V}$ in the above equation.

A value is potentially relevant if it is not known to have been effectively removed by a previous decision. The decision at a given node typically sends all of the observations whose input is in the selected category value subset down one branch, and the remaining observations are sent down the other (boolean compositions would result in exceptions). The list of relevant category values for a given input is immediately updated when the decision has obvious consequences (e.g., the values in Φ are removed from the list of relevant values used by the left branch in the previous example and the list for the right branch is set to Φ). However, a decision based on one input can also affect the remaining relevant category values of other inputs (e.g., suppose that at the node in our previous example, all prepositional phrase (PP) constituents had the head word *with* and *with* was a member of Φ, then the phrase type PP would no longer be relevant to decisions in the left branch, since all associated observations were sent down the right branch). Rather than update all of these lists at each node (a computationally expensive proposition), we only determine the unique category values when there are fewer than 1000 observations left on the path, or the number of observations has been cut to less than half what it was the last time unique values were determined. In early experimentation, this reduced the accuracy by about 0.4% relative to calculating the remaining category values after each decision. So when speed is not important, one should take the former approach.

Breiman indicates that, when several of the inputs are categorical, in order to increase strength enough to obtain a good accuracy rate the number of inputs evaluated at each node must be increased to two-three times $\lfloor 1 + \log_2 M \rfloor$ (where M is the number of inputs). It is not clear whether the input selection process is with or without replacement. Some of the inputs in the semantic parsing problem have five orders of magnitude more category values than others. Given this issue, if the selection is without replacement, it leads to evaluating features composed from each of our seven baseline inputs (figure 2) at each node. This would likely increase correlation, since those inputs with a very small number of category values will almost always be the most informative near the root of the tree and would be consistently used for the upper most decisions in the tree. On the other hand, if selection is with replacement, then using the Forest-RI method for calculating the input selection probability will result in those inputs with few category values almost never being chosen. For example, the baseline feature set has 577710 equivalent binary-valued features by the Forest-RI definition, including two true binary inputs. The probability of one of these two inputs *not* being chosen in a given random draw according to the Forest-RI method is 577709/577710 (see section 3.2 above). With M=7 inputs, generating $3\lfloor 1 + \log_2 M \rfloor = 9$ random composed features results in these two binary inputs having a selection probability of $1 - (577709/577710)^9$, or 0.000016.

Our compromise is first to use C and $\hat{V}$ from equation 2 to calculate a baseline number of *composable* features for each input i. This quantity is the total number of potentially relevant category values divided by the number used to create a composed feature:

$$f_i = \frac{\hat{V}_i}{C_i} \tag{3}$$

Second, given the large number of composable features f_i, we also evaluate a larger number, F, of random features at each node in the tree:

$$F = \max(\lceil \sqrt{f} \rceil, \min(f, \lfloor 1.5 + 3 \log_2(f) \rfloor)) \tag{4}$$

where f is the sum of f_i over all inputs. Finally, selection and feature composition is done with replacement. The final feature selection process has at least two significant effects we find positive. First, the number of composable features reflects the fact that several category values are considered simultaneously, effectively splitting on C_i binary-valued features. This has the effect of reducing the selection probability of many-valued inputs and increasing the probability of selecting inputs with fewer category values. Using the baseline feature set as an example, the probability of evaluating one of the binary-valued inputs at the root of the tree increases from 0.000016 to 0.0058. Second, as category values are used they are periodically removed from the set under consideration, reducing the corresponding size of V_i, and the input selection probabilities are then adjusted accordingly. This has the effect of continuously raising the selection probability for those inputs that have not yet been utilized.

Finally, we use ID3 to grow trees rather than CART, which is the tree algorithm Forest-RI uses. We don't believe this should have any significant effect on the final results. The choice was purely based on already having an implementation of ID3. From a set of possible split decisions, ID3 chooses the decision which leads to the minimum weighted average entropy among the training observations assigned to each branch, as determined by class labels (Quinlan, 1986; Mitchell, 1997).

These algorithm enhancements are appropriate for any task with high dimensional categorical inputs, which includes many NLP applications.

PREDICATE: the lemma of the predicate whose arguments are to be classified – the infinitive form of marked verbs in the corpus
CONSTITUENT PHRASE TYPE: the syntactic type assigned to the constituent/argument being classified
HEAD WORD (HW): the head word of the target constituent
PARSE TREE PATH (PATH): the sequence of parse tree constituent labels from the argument to its predicate
POSITION: a binary value indicating whether the target argument precedes or follows its predicate
VOICE: a binary value indicating whether the predicate was used in an active or passive phrase
SUB-CATEGORIZATION: the parse tree expansion of the predicate's grandparent constituent

Figure 2: Baseline feature set of experiment 1, see (Gildea and Jurafsky, 2002) for details

4 The Experiments

Four experiments are reported: the first uses the baseline features of Gildea and Jurafsky (2002); the second is composed of features proposed by Pradhan et al. (2003) and Surdeanu et al. (2003); the third experiment evaluates a new feature set; and the final experiment addresses a method of reducing the feature space. The experiments all focus strictly on the classification task – given a syntactic constituent known to be an argument of a given predicate, decide which argument role is the appropriate one to assign to the constituent.

4.1 Experiment 1: Baseline Feature Set

The first experiment compares the random forest classifier to three other classifiers, a statistical Bayesian approach with backoff (Gildea and Palmer, 2002), a decision tree classifier (Surdeanu et al., 2003), and a Support Vector Machine (SVM) (Pradhan et al., 2003). The baseline feature set utilized in this experiment is described in Figure 2 (see (Gildea and Jurafsky, 2002) for details).

Surdeanu et al. omit the SUB-CATEGORIZATION feature, but add a binary-valued feature that indicates the governing category of noun-phrase argument constituents. This feature takes on the value S or VP depending on which constituent type (sentence or verb phase respectively) eventually dominates the argument in the parse tree. This generally indicates grammatical subjects versus objects, respectively. They also used the predicate with its case and morphology intact, in addition to using its lemma. Surdeanu et al. indicate that, due to memory limitations on their hardware, they trained on only 75 KB of the PropBank argument constituents – about 60% of the annotated data.

Classifier	Accuracy
Bayesian (Gildea and Palmer, 2002)	82.8
Decision Tree (Surdeanu et al., 2003)	78.8
SVM (Pradhan et al., 2003)	87.1
First Tree	78.3
Random Forest	84.6

Table 2: Results of baseline feature set experiment

Table 2 shows the results of experiment 1, comparing the classifier accuracies as trained on the baseline feature set. Using a difference of two proportions test as described in (Dietterich, 1998), the accuracy differences are all statistically significant at p=0.01. The Random Forest approach outperforms the Bayesian method and the Decision Tree method. However, it does not perform as well as the SVM classifier. Interestingly, the classification accuracy of the first tree in the Random Forest, given in row four, is almost as high as that of the C5 decision trees (Quinlan, 2002) of Surdeanu et al.

4.2 Experiment 2: Extended Feature Set

The second experiment compares the random forest classifier to the boosted decision tree and the SVM using all of the features reported by Pradhan et al. The additional features used in this experiment are listed in Figure 3 (see sources for further details). In addition to the extra features noted in the previous experiment, Surdeanu et al. report on four more features, not included here (content word part of speech (CW PoS)[1], CW named entity class, and two phrasal verb collocation features).

Table 3 shows the results of experiment 2, comparing the classifier accuracies using the full feature sets reported in each source. Surdeanu et al. also applied boosting in this experiment and chose the outcome of the boosting iteration that performed best. Using the difference of two proportions test, the accuracy differences are all statistically significant at p=0.01. The Random Forest approach outperforms the Boosted Decision Tree method by 3.5%, but trails the SVM classifier by 2.3%. In analyzing the performance on individual argument classes using McNemar's test, Random Forest performs significantly better on ARG0 (p=0.001) then the SVM, and the SVM has significantly better results on ARG1 (p=0.001). The large number of degrees of freedom

[1]We also tested the CW PoS, but it did not improve the development results and was omitted.

NAMED ENTITIES: seven binary-valued features indicating whether specific named entities (PERSON, ORGANIZATION, DATE, TIME, MONEY, LOCATION, and PERCENT) occurred anywhere in the target constituent (Surdeanu et al., 2003)
HW POS: the grammatical part of speech of the target constituent's head word (Surdeanu et al., 2003)
CONTENT WORD (CW): "lexicalized feature that selects an informative word from the constituent, different from the head word"(Surdeanu et al., 2003)
VERB CLUSTER: a generalization of the verb predicate by clustering verbs into 64 classes (Pradhan et al., 2003)
HALF PATH: the sequence of parse tree constituent labels from the argument to the lowest common ancestor of the predicate (Pradhan et al., 2003)

Figure 3: Additional features in experiment 2

Classifier	Accuracy
Boosted Decision Tree (Surdeanu et al., 2003)	83.7
Random Forest (trained with CW)	87.2
SVM (Pradhan et al., 2003)	88.9
Random Forest (trained without CW)	86.6

Table 3: Results of experiment 2

prevent significance at p=0.1 for any other arguments, but the SVM appears to perform much better on ARG2 and ARG3.

4.3 Experiment 3: New Features

We evaluated several new features and report on the most significant here, as described in figure 4.[2] The results are reported in table 4. The accuracy improvements relative to the results from experiment 2 are all statistically significant at p=0.001 (McNemar's test is used for all significance tests in this section). Comparing the SVM results in experiment 2 to the best results here shows statistical significance

[2]Due to space, we cannot report all experiments; contact the first author for more information. The other features we evaluated involved: the phrase type of the parent constituent, the list of phrase types encompassing the sentence fragment between the target predicate and constituent, the prefix and suffix of the cw and hw, animacy, high frequency words preceding and following the predicate, and the morphological form of the predicate. All of these improved accuracy on the development set (some with statistical significance at p=0.01), but we suspect the development baseline was at a low point, since these features largely did not improve performance when combined with CW Base and GP.

GOVERNING PREPOSITION (GP): if the constituent's parent is a PP, this is the associated preposition (e.g., in "made of [$_{Arg2}$ gallium arsenide]", this feature is 'of', since the Arg2-NP is governed by an 'of'-based PP)
CW BASE: starting with the CW, convert it to its singular form, remove any prefix, and convert digits to 'n' (e.g., this results in the following CW → CW Base mappings: accidents → accident, non-binding → binding, repayments → payment, and 1012 → nnnn)

Figure 4: Features in experiment 3

Feature Set	Accuracy
Extended (see figures 2 & 3)	86.6
Extended + CW BASE	87.4
Extended + GOVERNING PREPOSITION	87.4
Extended + CW BASE & GP	88.3

Table 4: Results of experiment 2

only at p=0.1.

In analyzing the effect on individual argument classes, seven have high χ^2 values (ARG2-4, ARGM-DIS (discourse), ARGM-LOC (locative), ARGM-MNR (manner), and ARGM-TMP (temporal)), but given the large number of degrees of freedom, only ARGM-TMP is significant (p=0.05). Example section-00 sentence fragments including the target predicate (P) and ARG2 role whose classification was corrected by the GP feature include "[$_P$ banned] to [everyday visitors]", "[$_P$ considered] as [an additional risk for the investor]", and "[$_P$ made] of [gallium arsenide]". Comparing the SVM results to the best results here, the Random Forest performs significantly better on Arg0 (p=0.001), and the SVM is significantly better on Arg1 (p=0.001). Again the degrees of freedom prevent significance at p=0.1, but the Random Forest outperforms the SVM with a fairly high χ^2 value on ARG4, ARGM-DIS, ARGM-LOC, and ARGM-TMP.

4.4 Experiment 4: Dimensionality Reduction

We originally assumed we would be using binary-valued features with sparse matrices, much like in the SVM approach. Since many of the features have a very large number of values (e.g., the PATH feature has over 540k values), we sought ways to reduce the number of equivalent binary-valued features. This section reports on one of these methods, which should be of interest to others in resource constrained environments.

In this experiment, we preprocess the baseline in-

puts described in Figure 2 to reduce their number of category values. Specifically, for each original category value, $v_i \in V$, we determine whether it occurs in observations associated with one or more than one semantic role label, R. If it is associated with more than one R, v_i is left as is. When v_i maps to only a single R_j, we replace v_i with an arbitrary value, $v_k \notin V$, which is the same for all such v occurring strictly in association with R_j. The PATH input starts with 540732 original feature values and has only 1904 values after this process, while HEAD WORD is reduced from 33977 values to 13208 and PHRASE TYPE is reduced from 62 to 44 values. The process has no effect on the other baseline input features. The total reduction in equivalent binary-valued features is 97%. We also test the effect of disregarding feature values during training if they only occur once in the training data. This has a more modest effect, reducing PATH to 156788 values and HEAD WORD to 29482 values, with no other reductions. The total reduction in equivalent binary-valued features is 67%.

Training on the baseline feature set, the net effect of these two procedures was less than a 0.3% loss of accuracy on the development set. The McNemar test indicates this is not significant at p=0.1. In the end, our implementation used categorical features, rather than binary-valued features (e.g., rather than use 577710 binary-valued features to represent the baseline inputs, we use 7 features which might take on a large number of values – PATH has 540732 values). In this case, the method does not result in as significant a reduction in the memory requirements. While we did not use this feature reduction in any of the experiments reported previously, we see it as being very beneficial to others whose implementation may be more resource constrained, particularly those using a binary-valued feature representation.

The method also reduced training time by 17% and should lead to much larger reductions for implementations using binary-valued features. For example, the worst case training time for SVMs is quadratic in the number of features and this method reduced the dimensionality to 3% of its original size. Therefore, the method has the theoretical potential to reduce training time by up to $100(1\text{-}0.03^2) = 99.91\%$. While it is unlikely to approach this in practice, it should provide significant savings. This may be especially helpful during model selection or feature evaluation, after which, one could revert to the full dimensionality for final training to improve classification accuracy. The slight decrement in accuracy may also be overcome by the ability to handle larger datasets.

5 Discussion and Future Research

The version of Random Forests described here outperforms the Bayesian algorithm (Gildea and Jurafsky, 2002; Gildea and Palmer, 2002) by 1.8% on the same feature set and outperforms the boosted decision tree classifier (Surdeanu et al., 2003) by 3.5% on the extended feature set with 5 fewer features. The SVM classifier (Pradhan et al., 2003) was 2.3% better training on the same data, but only 0.6% better than our best results.

The Random Forest (RF) approach has advantages that might make it a better choice than an SVM in certain circumstances. Conceptually, it is simpler to understand and can be implemented more easily. This also makes it easier to modify the algorithm to evaluate new techniques. RFs allow one to more easily implement multi-class classifiers. The RFs here were implemented as a single classifier, rather than as the 22 one-against-all classifiers required by the SVM approach. Since RFs are not overly sensitive to noise in the training data (Breiman, 2001), it might be the case that they will narrow the performance gap when training is based on automatically parsed sentences. Further research is required in this area. Additionally, RFs have an advantage in training time. It takes about 40% of the SVM time (8 versus 20 hours) to train on the extended feature set for the classification task and we expect this time to be cut by up to a factor of 10 in porting from MatLab to C. Classification time is generally faster for RFs as well, which is important for real-time tasks.

In a class-by-class comparison, using the same features, the RF performed significantly better than the SVM on Arg0 roles, the same or slightly better on 12 of the other 21 arguments, and slightly better overall on the 14 adjunctive arguments (77.8% versus 77.3% accuracy on 1882 observations). Reviewing performance on data not seen during training, both algorithms degraded to about 94% of their accuracy on seen data.

The RF algorithm should be evaluated on the identification task and on the combined identification and classification task. This will provide additional comparative evidence to contrast it with the SVM approach. Further research is also required to determine how RFs generalize to new genres.

Another area for future research involves the estimation of class probabilities. MOB-ESP, a variant of Random Forests which outputs class probability estimates, has been shown to produce very good results (Nielsen, 2004). Preliminary experiments suggest that using these probability estimates in conjunction with an SVM classifier might be more ef-

fective than estimating probabilities based on the example's distance from the decision surface as in (Platt, 2000). Class probabilities are useful for several semantic parsing and more general NLP tasks, such as selective use of labeled examples during training (c.f., Pradhan et al., 2003) and N-best list processing.

6 Conclusion

The results documented in these experiments are very promising and mandate further research. The final classification accuracy of the Random Forest was 88.3%, just 0.6% behind the SVM results (Pradhan et al., 2003) and 4.6% higher than the next best results (Surdeanu et al., 2003) – results that were based on a number of additional features.

We defined several modifications to the RF algorithm that increased accuracy. These improvements are important for any application with high dimensional categorical inputs, which includes many NLP tasks. We introduced new features which provided a 1.1% improvement in accuracy over the best results using features from the literature. We also introduced a technique to reduce the dimensionality of the feature space, resulting in a reduction to just 3% of the original feature space size. This could be an important enabler for handling larger datasets and improving the efficiency of feature and model selection.

Acknowledgements

We thank Dan Jurafsky for miscellaneous support and for valuable feedback on a draft of this paper. Thanks also go to the anonymous reviewers whose feedback improved the paper.

References

Yali Amit and Donald Geman. 1997. Shape Quantization and Recognition with Randomized Trees. *Neural Computation*, 9:1545–1588.

Leo Breiman. 2001. Random Forests. *Journal of Machine Learning*, 45(1):5–32.

Leo Breiman. 2004. Random Forests. http://stat-www.berkeley.edu/users/breiman/RandomForests/

Leo Breiman. 1996. Bagging Predictors. *Machine Learning*, 26(2):123–140.

Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.

Y. Freund and R. E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and Systems Sciences*, 55(1):119–139.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

Daniel Gildea and Martha Palmer. 2002. The Necessity of Parsing for Predicate Argument Recognition. *Proceedings of ACL-02*.

Paul Kingsbury, Martha Palmer, and Mitch Marcus. 2002. Adding semantic annotation to the Penn Treebank. *Proceedings of the HLT-02*.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn TreeBank: Annotating predicate argument structure.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Boston, MA.

Rodney D. Nielsen. 2004. MOB-ESP and other Improvements in Probability Estimation. *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*.

John Platt. 2000. Probabilities for Support Vector Machines. In A. Smola, P. Bartlett, B. Scolkopf, and D. Schuurmans (Eds), *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA.

Sameer Pradhan, Kadri Hacioglu, Valerie Krugler, Wayne Ward, James H. Martin, Daniel Jurafsky. 2003. Shallow Semantic Parsing using Support Vector Machines. *University of Colorado Technical Report: TR-CSLR-2003-03*.

J. R. Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1:81–106.

J. R. Quinlan. 2002. Data Mining Tools See5 and C5.0. http://www.rulequest.com/see5-info.html.

Mihai Surdeanu, Sanda Harabagiu, John Williams and Paul Aarseth. 2003. Using Predicate-Argument Structures for Information Extraction. *Proceedings of ACL-03*.