Feature Selection for a Rich HPSG Grammar Using Decision Trees

Kristina Toutanova and **Christopher D. Manning**
Computer Science Department
Stanford University
Stanford, CA 94305-9040, USA

Abstract

This paper examines feature selection for log linear models over rich constraint-based grammar (HPSG) representations by building decision trees over features in corresponding probabilistic context free grammars (PCFGs). We show that single decision trees do not make optimal use of the available information; constructed ensembles of decision trees based on different feature subspaces show significant performance gains (14% parse selection error reduction). We compare the performance of the learned PCFG grammars and log linear models over the same features.

1 Introduction

Hand-built NLP grammars frequently have a depth of linguistic representation and constraints not present in current treebanks, giving them potential importance for tasks requiring deeper processing. On the other hand, these manually built grammars need to solve the disambiguation problem to be practically usable.

This paper presents work on the problem of probabilistic parse selection from among a set of alternatives licensed by a hand-built grammar in the context of the newly developed Redwoods HPSG treebank (Oepen et al., 2002). HPSG (Head-driven Phrase Structure Grammar) is a modern constraint-based lexicalist (unification) grammar, described in Pollard and Sag (1994).

The Redwoods treebank makes available syntactic and semantic analyses of much greater depth than, for example, the Penn Treebank. Therefore there are a large number of features available that could be used by stochastic models for disambiguation. Other researchers have worked on extracting features useful for disambiguation from unification grammar analyses and have built log linear models a.k.a. Stochastic Unification Based Grammars (Johnson et al., 1999; Riezler et al., 2000). Here we also use log linear models to estimate conditional probabilities of sentence analyses. Since feature selection is almost prohibitive for these models, because of high computational costs, we use PCFG models to select features for log linear models. Even though this method may be expected to be suboptimal, it proves to be useful. We select features for PCFGs using decision trees and use the same features in a conditional log linear model. We compare the performance of the two models using equivalent features.

Our PCFG models are comparable to branching process models for parsing the Penn Treebank, in which the next state of the model depends on a history of features. In most recent parsing work the history consists of a small number of manually selected features (Charniak, 1997; Collins, 1997). Other researchers have proposed automatically selecting the conditioning information for various states of the model, thus potentially increasing greatly the space of possible features and selectively choosing the best predictors for each situation. Decision trees have been applied for feature selection for statistical parsing models by Magerman (1995) and Haruno et al. (1998). Another example of automatic feature selection for parsing is in the context of a deterministic parsing model that chooses parse actions based on automatically induced decision structures over a very rich feature set (Hermjakob and Mooney, 1997).

Our experiments in feature selection using decision trees suggest that single decision trees may not be able to make optimal use of a large number of relevant features. This may be due to the greedy search procedures or to the fact that trees combine information from different features only through partitioning of the space. For example they have difficulty in weighing evidence from different features without fully partitioning the space.

A common approach to overcoming some of the problems with decision trees – such as reducing their variance or increasing their representational power – has been building ensembles of decision trees by, for example, bagging (Breiman, 1996) or boosting (Freund and Schapire, 1996). Haruno et al. (1998) have experimented with boosting decision trees, reporting significant gains. Our approach is to build separate decision trees using different (although not disjoint) subsets of the feature space and then to combine their estimates by using the average of their predictions. A similar method based on random feature subspaces has been proposed by Ho (1998), who found that the random feature subspace method outperformed bagging and boosting for datasets with a large number of relevant features where there is redundancy in the features. Other examples of ensemble combination based on different feature subspaces include Zheng (1998) who learns combinations of Naive Bayes classifiers and Zenobi and Cunningham (2001) who create ensembles of kNN classifiers.

We begin by describing the information our HPSG corpus makes available and the subset we have attempted to use in our models. Next we describe our ensembles of decision trees for learning parameterizations of branching process models. Finally, we report parse disambiguation results for these models and corresponding conditional log linear models.

2 Characteristics of the Treebank and Features Used

The Redwoods treebank (Oepen et al., 2002) is an under-construction treebank of sentences corresponding to a particular HPSG grammar, the LinGO ERG (Flickinger, 2000). The current preliminary version contains 10,000 sentences of spoken dialog material drawn from the Verbmobil project. The Redwoods treebank makes available the entire HPSG signs for sentence analyses, but we have used in our experiments only small subsets of this representation. These are (i) derivation trees composed of identifiers of lexical items and constructions used to build the analysis, and (ii) semantic dependency trees which encode semantic head-to-head relations. The Redwoods treebank provides deeper semantics expressed in the Minimum Recursion Semantics formalism (Copestake et al., 2001), but in the present experiments we have not explored this fully.

The nodes in the derivation trees represent combining rule schemas of the HPSG grammar, and not

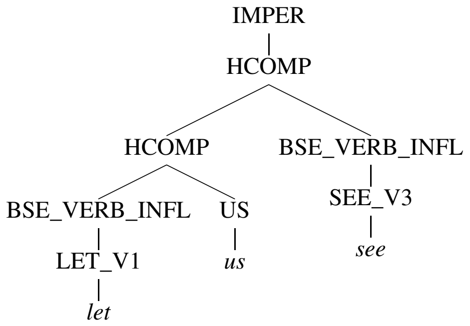
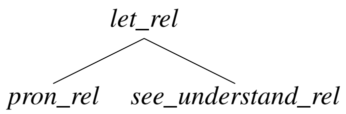


Figure 1: Derivation tree for the sentence *Let us see*

phrasal categories of the standard sort. The whole HPSG analyses can be recreated from the derivation trees, using the grammar. The preterminals of the derivation trees are lexical labels. These are much finer grained than Penn Treebank preterminals tags, and more akin to those used in Tree-Adjoining Grammar models (Bangalore and Joshi, 1999). There are a total of about $8,000$ lexical labels occurring in the treebank. One might conjecture that a supertagging approach could go a long way toward parse disambiguation. However, an upper bound for such an approach for our corpus is below 55 percent parse selection accuracy, which is the accuracy of an oracle tagger that chooses at random among the parses having the correct tag sequence (Oepen et al., 2002).

The semantic dependency trees are labelled with relations most of which correspond to words in the sentence. These labels provide some abstraction because some classes of words have the same semantic label — for example all days of week are grouped in one class, as are all numbers.

As an example the derivation tree for one analysis of the short sentence *Let us see* is shown in figure 1. The semantic dependency tree for the same sentence is:



In addition to this information we have used the main part of speech information of the lexical head to annotate nodes in the derivation trees with labels like verb, noun, preposition, etc.

Other information that we have not explored includes subcategorization information, lexical types (these are a rich set of about 500 syntactic types), individual features such as tense, aspect, gender, etc. Another resource is the type hierarchy which can be explored to form equivalence classes on which to base statistical estimation.

3 Models

3.1 Generative Models

We learn generative models that assign probabilities to the derivation trees and the dependency trees. We train these models separately and in the final stage we combine them to yield a probability or score for an entire sentence analysis. We rank the possible analyses produced by the HPSG grammar in accordance with the estimated scores.

We first describe learning such a generative model for derivation trees using a single decision tree and a set of available features. We will call the set of available features $\{f_1, \ldots, f_m\}$ a history. We estimate the probability of a derivation tree as $P(t) = \prod_{n \in t} P(expansion(n)|history(n))$. In other words, the probability of the derivation tree is the product of the probabilities of the expansion of each node given its history of available features.

Given a training corpus of derivation trees corresponding to preferred analyses of sentences we learn the distribution $P(expansion|history)$ using decision trees. We used a standard decision tree learning algorithm where splits are determined based on gain ratio (Quinlan, 1993). We grew the trees fully and we calculated final expansion probabilities at the leaves by linear interpolation with estimates one level above. This is a similar, but more limited, strategy to the one used by Magerman (1995).

The features over derivation trees which we made available to the learner are shown in Table 1. The node direction features indicate whether a node is a left child, a right child, or a single child. A number of ancestor features were added to the history. The grammar used, the LinGO ERG has rules which are maximally binary, and the complements and adjuncts of a head are collected through multiple rules. Moreover, it makes extensive use of unary rules for various kinds of "type changing" operations. A simple PCFG is reasonably effective to the extent that important dependencies are jointly expressed in a local tree, as is mostly the case for the much flatter representations used in the Penn Treebank. Here, this is not the case, and the inclusion of ancestor nodes in the history makes necessary information more often local in our models. Grandparent annotation was used previously by Charniak and Carroll (1994) and Johnson (1998).

No.	Name	Example
0	Node Label	*HCOMP*
1	Parent Node Label	*HCOMP*
2	Node Direction	*left*
3	Parent Node Direction	*none*
4	Grandparent Node Label	*IMPER*
5	Great Grandparent Top ?	*yes*
6	Left Sister Node Label	*HCOMP*
7	Left Preterminal	*US*
8	Preterminal to the Left of 7	*LET_V1*
9	Category of Node	*verb*

Table 1: Features over derivation trees

No	Name	Example
0	Node Label	*let_rel*
1	Direction of Dependent	*left*
2	Number of Intervening Dependents	*1*
3	Parent Node Label	*top*
4	Label to the Left or Right	*pron_rel*

Table 2: Features over semantic dependency trees

Similarly we learn generative models over semantic dependency trees. For these trees the expansion of a node is viewed as consisting of separate trials for each dependent. Any conditional dependencies among children of a node can be captured by expanding the history. The features used for the semantic dependency trees are shown in Table 2. This set of only 5 features for semantic trees makes the feature subset selection method less applicable since there is no obvious redundancy in the set. However the method still outperforms a single decision tree. The model for generation of semantic dependents to the left and right is as follows: First the left dependents are generated from right to left given the head, its parent, right sister, and the number of dependents to the left that have already been generated. After that, the right dependents are generated from left to right, given the head, its parent, left sister and number of dependents to the right that have already been generated. We also add stop symbols at the ends to the left and right. This model is very similar to the markovized rule models in Collins (1997). For example, the joint probability of the dependents of *let_rel* in the above example would be:

$$\begin{aligned}&P(stop|let\_rel,left,0,top,none)\times\\&P(pron\_rel|let\_rel,right,0,top,stop)\times\\&P(see\_understand\_rel|let\_rel,right,1,top,pron\_rel)\times\\&P(stop|let\_rel,right,2,top,see\_understand\_rel)\end{aligned}$$

3.2 Conditional Log Linear Models

A conditional log linear model for estimating the probability of an HPSG analysis given a sentence has a set of features $\{f_1, \ldots, f_m\}$ defined over analyses and a set of corresponding weights $\{\lambda_1, \ldots, \lambda_m\}$ for them. In this work we have defined features over derivation trees and syntactic trees as described for the branching process models.

For a sentence s with possible analyses $t_1, \ldots, t_k$, the conditional probability for analysis t_i is given by:

$$P(t_i|s) = \frac{\exp\left(\sum_{j=1}^{m} f_j(t_i)\lambda_j\right)}{\sum_{i'=1}^{k}\exp\left(\sum_{j=1}^{m} f_j(t_{i'})\lambda_j\right)} \quad (1)$$

As in Johnson et al. (1999) we trained the model by maximizing the conditional likelihood of the preferred analyses and using a Gaussian prior for smoothing (Chen and Rosenfeld, 1999). We used conjugate gradient for optimization.

Given an ensemble of decision trees estimating probabilities $P(expansion|history)$ we define features for a corresponding log linear model as follows: For each path from the root to a leaf in any of the decision trees, and for each possible expansion for that path that was seen in the training set, we add a feature $f_{eh}(t)$. For a tree t, this feature has as value the number of time the expansion e occurred in t with the history h.

4 Experiments

We present experimental results comparing the parse ranking performance of different models. The accuracy results are averaged over a ten-fold cross-validation on the data set summarized in Table 3. The sentences in this data set have exactly one preferred parse selected by a human annotator. At this early stage, the treebank is expected to be noisy because all annotation was done by a single annotator. Accuracy results denote the percentage of test sentences for which the highest ranked analysis was the correct one. This measure scores whole sentence accuracy and is therefore more strict than the labelled precision/recall measures and more appropriate for the task of parse ranking. When a model ranks a set of m parses highest with equal scores and one of those parses is the preferred parse in the treebank, we compute the accuracy on this sentence as $1/m$.

To give an idea about the difficulty of the task on the corpus we have used, we also show a baseline which is the expected accuracy of choosing a parse at random and accuracy results from simpler models that have been used broadly in NLP. PCFG-S is a simple PCFG model where we only have the node label (feature 0) in the history, and PCFG-GP has only the node and its parent's labels (features 0 and 1) as in PCFG grammars with grandparent annotation.

sentences	length	lex ambig	struct ambig
5277	7.0	4.1	7.3

Table 3: Annotated corpus used in experiments: The columns are, from left to right, the total number of sentences, average length, and average lexical and structural ambiguity

Model	**Generative**		**Log Linear**	
	Test	Train	Test	Train
Random	26.00	26.00	26.00	26.00
PCFG-S	67.27	72.23	79.34	85.31
PCFG-GP	72.39	83.89	81.52	91.56
PCFG-DTAll	75.57	96.51	81.82	97.61

Table 4: Parse ranking accuracy of syntactic models: single decision tree compared to simpler models

Table 4 shows the accuracy of parse selection of the three simple models mentioned above defined over derivation trees and the accuracy achieved by a single decision tree (PCFG-DTAll) using all features in Table 1. The third column contains accuracy results for log linear models using the same features.

We can note from Table 4 that the generative models greatly benefit from the addition of more conditioning information, while the log linear model performs very well even with only simple rule features, and its accuracy does not increase so sharply with the addition of more complex features. The error reduction from PCFG-S to PCFG-DTAll is 25.36%, while the corresponding error reduction for the log linear model is 12%. The error reduction for the log linear model from PCFG-GP to PCFG-DTAll is very small which suggests an overfitting effect. PCFG-S is doing much worse than the log linear model with the same features, and this is true for the training data as well as for the test data. A partial explanation for this is the fact that PCFG-S tries to maximize the likelihood of the correct parses under strong independence assumptions, whereas the log linear model need only worry about making the correct parses more probable than the incorrect ones.

Next we show results comparing the single deci-

Type of Model	All Features		Feature Subspaces	
	PCFG	Log linear	PCFG	Log linear
Derivation Trees	75.57	81.82	78.97	82.24
Dependency Trees	67.38	69.91	68.88	73.50
Combined Feature Subspaces Accuracy			80.10	83.32

Table 5: Parse ranking accuracy: single decision trees and ensembles

sion tree model (PCFG-DTAll) to an ensemble of 11 decision trees based on different feature subspaces. The decision trees in the ensemble are used to rank the possible parses of a sentence individually and then their votes are combined using a simple majority vote. The sets of features in each decision tree are obtained by removing two features from the whole space. The left preterminal features (features with numbers 7 and 8) participate in only one decision tree. Also, features 2, 3, and 5 participate in all decision trees since they have very few possible values and should not partition the space too quickly. The feature space of each of the 10 decision trees not containing the left preterminal features was formed by removing two of the features from among those with numbers {0, 1, 4, 6, and 9} from the initial feature space (minus features 7 and 8). This method for constructing feature subspaces is heuristic, but is based on the intuition of removing the features that have the largest numbers of possible values.[1]

Table 5 shows the accuracy results for models based on derivation trees, semantic dependency trees, and a combined model. The first row shows parse ranking accuracy using derivation trees of generative and log linear models over the same features. Results are shown for features selected by a a single decision tree, and an ensemble of 11 decision tree models based on different feature subspaces as described above. The relative improvement in accuracy of the log linear model from single to multiple decision trees is fairly small.

The second row shows corresponding models for the semantic dependency trees. Since there are a small number of features used for this task, the performance gain from using feature subspaces is not so large. It should be noted that there is a 90.9% upper bound on parse ranking accuracy using semantic trees only. This is because for many sentences there are several analyses with the same semantic dependency structure. Interestingly, for semantic trees the difference between the log linear and generative models is not so large. Finally, the last row shows the combination of models over derivation trees and semantic trees. The feature subspace ensemble of 11 decision tree models for the derivation trees is combined with the ensemble of 5 feature subspace models over semantic dependencies to yield a larger ensemble that ranks possible sentence analyses based on weighted majority vote (with smaller weights for the semantic models). The improvement for PCFG models from combining the syntactic and semantic models is about 5.4% error reduction from the error rate of the better (syntactic) models. The corresponding log linear model contains all features from the syntactic and semantic decision trees in the ensemble. The error reduction due to the addition of semantics is 6.1% for the log linear model. Overall the gains from using semantic information are not as good as we expected. Further research remains to be done in this area.

The results show that decision trees and ensembles of decision trees can be used to greatly improve the performance of generative models over derivation trees and dependency trees. The performance of generative models using a lot of conditioning information approaches the performance of log linear models although the latter remain clearly superior. The corresponding improvement in log linear models when adding more complex features is not as large as the improvement in generative models. On the other hand, there might be better ways to incorporate the information from additional history in log linear models.

5 Error Analysis

It is interesting to see what the hard disambiguation decisions are, that the combined syntactic-semantic

[1] We also preformed an experiment where we removed every combination of two features from the whole space of features 0–8 to obtain subspaces. This results in a large number of feature subspaces (36). The performance of this method was slightly worse than the result reported in Table 5 (78.52%). We preferred to work with an ensemble of 11 decision trees for computational reasons.

models can not at present get right.

We analyzed some of the errors made by the best log linear model defined over derivation trees and semantic dependency trees. We selected for analysis sentences that the model got wrong on one of the training - test splits in the 10 fold cross-validation on the whole corpus. The error analysis suggests the following breakdown:

- About 40% of errors are due to inconsistency or errors in annotation
- About 15% of the errors are due to grammar limitations
- About 45% of the errors are real errors and we could hope to get them right

The inconsistency in annotation hurts the performance of the model both when in the training data some sentences were annotated incorrectly and the model tried to fit its parameters to explain them, and when in the test data the model chose the correct analysis but it was scored as incorrect because of incorrect annotation. It is not straightforward to detect inconsistencies in the training data by inspecting test data errors. Therefore the percentages we have reported are not exact.

The log linear model seems to be more susceptible to errors in the training set annotation than the PCFG models, because it can easily adjust its parameters to fit the noise (causing overfitting), especially when given a large number of features. This might partly explain why the log linear model does not profit greatly over this data set from the addition of a large number of features.

A significant portion of the real errors made by the model are PP attachment errors. Another class of errors come from parallel structures and long distance dependencies. For example, the model did not disambiguate correctly the sentence *Is anywhere from two thirty to five on Thursday fine?*, preferring the interpretation *from [two thirty] to [five on Thursday]* rather than what would be the more common meaning *[from [two thirty] to [five]] [on Thursday]*. This disambiguation decision seems to require common world knowledge or it might be addressable with addition of knowledge about parallel structures. ((Johnson et al., 1999) add features measuring parallelism).

We also compared the errors made by the best log linear model using only derivation tree features to the ones made by the combined model. The large majority of the errors made by the combined model were also made by the syntactic model. Examples of errors corrected with the help of semantic information include:

The sentence *How about on the twenty fourth Monday?* (punctuation is not present in the corpus) was analyzed by the model based on derivation trees to refer to the Monday after twenty three Mondays from now, whereas the more common interpretation would be that the day being referred to is the twenty fourth day of the month, and it is also a Monday. There were several errors of this sort corrected by the dependency trees model.

Another interesting error corrected by the semantic model was for the sentence: *We will get a cab and go*. The syntactic model chose the interpretation of that sentence in the sense: *We will become a cab and go*, which was overruled by the semantic model.

6 Conclusions and Future Work

We have presented work on building probabilistic models for HPSG parse disambiguation. As the number of available features increases it becomes more important to select relevant features automatically. We have shown that decision trees using different feature subspaces can be combined in ensembles that choose the correct parse with higher accuracy than individual models.

Our plans for future work include exploring more information from the HPSG signs and defining features that capture long distance dependencies. Another line of future research is defining models over the deeper MRS semantic representations, possibly in conjunction with clustering of semantic types.

7 Acknowledgements

We would particularly like to thank Stephan Oepen, for directing the Redwoods treebanking project and getting us set up with the HPSG development environment, and Dan Flickinger, for explanations of the LinGO ERG and help with the error analysis. Thanks also to Stuart Shieber and other participants in the Redwoods project meetings for many discussions. And our thanks to Dan Klein for letting us use his implementation of conjugate gradient, and to the anonymous CoNLL-2002 reviewers for helpful comments. This paper is based upon work supported in part by the National Science Foundation under Grant No. 0085896.

References

Srinivas Bangalore and Aravind K. Joshi. 1999. Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2).

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Eugene Charniak and G. Carroll. 1994. Context-sensitive statistics for improved grammatical language models. In *Proceedings of the Twelth National Conference on Artificial Intelligence*, pages 742 – 747, Seattle, WA.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 598 – 603, Providence, RI.

S. Chen and R. Rosenfeld. 1999. A gaussian prior for smoothing maximum entropy models. In *Technical Report CMUCS -99-108*.

Michael John Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics and the 7th Conference of the European Chapter of the ACL*, pages 16 – 23, Madrid, Spain.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of the 39th Meeting of the Association for Computational Linguistics*, Toulouse, France.

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6 (1) (Special Issue on Efficient Processing with HPSG):15 – 28.

Yoav Freund and Robert E. Schapire. 1996. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156.

Masahiko Haruno, Satoshi Shirai, and Yoshifumi Ooyama. 1998. Using decision trees to construct a practical parser. In *Proceedings of the 36th Meeting of the Association for Computational Linguistics*, pages 505 – 511.

Ulf Hermjakob and Reymond J. Mooney. 1997. Learning parse and translation decisions from examples with rich context. In *Proceedings of the 35th Meeting of the Association for Computational Linguistics and the 7th Conference of the European Chapter of the ACL*, pages 482 – 489.

Tin Kam Ho. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic ‘unification-based’ grammars. In *Proceedings of the 37th Meeting of the Association for Computational Linguistics*, pages 535 – 541, College Park, MD.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613—632.

D. M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics*.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGo Redwoods treebank: Motivation and preliminary applications. In *COLING 19*.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. University of Chicago Press.

J. R. Quinlan. 1993. *C4.5 Programs for Machine Learning*. Morgan Kaufmann.

Stefan Riezler, Detlef Prescher, Jonas Kuhn, and Mark Johnson. 2000. Lexicalized stochastic modeling of constraint-based grammars using log-linear measures and EM training. In *Proceedings of the 38th Meeting of the Association for Computational Linguistics*, Hong Kong.

Gabriele Zenobi and Padraig Cunningham. 2001. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *ECML*, pages 576–587.

Z. Zheng. 1998. Naive Bayesian classifier committees. In *ECML*, pages 196–207.