Ensemble-based Active Learning for Parse Selection

Miles Osborne and Jason Baldridge
School of Informatics
University of Edinburgh
Edinburgh EH8 9LW, UK
{miles,jbaldrid}@inf.ed.ac.uk

Abstract

Supervised estimation methods are widely seen as being superior to semi and fully unsupervised methods. However, supervised methods crucially rely upon training sets that need to be manually annotated. This can be very expensive, especially when skilled annotators are required. Active learning (AL) promises to help reduce this annotation cost. Within the complex domain of HPSG parse selection, we show that ideas from ensemble learning can help further reduce the cost of annotation. Our main results show that at times, an ensemble model trained with randomly sampled examples can outperform a single model trained *using* AL. However, converting the single-model AL method into an ensemble-based AL method shows that even this much stronger baseline model can be improved upon. Our best results show a 73% reduction in annotation cost compared with single-model random sampling.

1 Introduction

Active learning (AL) methods, such as uncertainty sampling (Cohn et al., 1995) or query by committee (Seung et al., 1992), can dramatically reduce the cost of creating an annotated dataset. In particular, they enable rapid creation of labeled datasets which can then be used for trainable speech and language technologies. Progress in AL will therefore translate into even greater savings in annotation costs and hence faster creation of speech and language systems.

In this paper, we:

- Present a novel way of improving uncertainty sampling by generalizing it from using a single model to using an ensemble model. This generalization easily outperforms single-model uncertainty sampling.
- Introduce a new, extremely simple AL method (called *lowest best probability selection*) which is competitive with uncertainty sampling and can also be improved using ensemble techniques.
- Show that an ensemble of models trained using randomly sampled examples can outperform a single model trained *using* (single model) AL methods.
- Demonstrate further reductions in annotation cost when we train the ensemble parse selection model using examples selected by an ensemble-based active learner. This result shows that ensemble learning can improve both the underlying model and also the way we select examples for it.

Our domain is parse selection for Head-Driven Phrase Structure Grammar (HPSG). Although annotated corpora exist for HPSG, such corpora do not exist in significant volumes and are limited to a few small domains (Oepen et al., 2002). Even if it were possible to bootstrap from the Penn Treebank, it is still unlikely that there would be sufficient quantities of high quality material necessary to improve parse selection for detailed linguistic formalisms such as HPSG. There is thus a pressing need to efficiently create significant volumes of annotated material.

AL applied to parse selection is much more challenging than applying it to simpler tasks such as text classification or part-of-speech tagging. Our labels are complex objects rather than discrete values drawn from a small, fixed set. Furthermore, the fact that sentences are of variable length and have variable numbers of parses potentially adds to the complexity of the task.

Our results specific to parse selection show that:

- An ensemble of three parse selection models is able to achieve a 10.8% reduction in error rate over the best single model.
- Annotation cost should not assume a unit expenditure per example. Using a more refined cost met-

ric based upon efficiently selecting the correct parse from a set of possible parses, we are able to show that some AL methods are more effective than others, even though they perform similarly when making the unit cost per example assumption.

- Ad-hoc selection methods based upon superficial characteristics of the data, such as sentence length or ambiguity rate, are typically worse than random sampling. This motivates using AL methods.

- Labeling sentences in the order they appear in the corpus – as is typically done in annotation – performs much worse than using random selection.

Throughout this paper, we shall treat the terms *sentences* and *examples* as interchangeable; we shall also consider *parses* and *labels* as equivalent. Also, we shall use the term *method* whenever we are talking about AL, and *model* whenever we are talking about parse selection.

2 Parse selection

2.1 The Redwoods treebank

Many broad coverage grammars providing detailed syntactic and semantic analyses of sentences exist for a variety of computational grammar frameworks, but their purely symbolic nature means that when ordering licensed analyses, parse selection models are necessary. To overcome this limitation for the HPSG English Resource Grammar (ERG, Flickinger (2000)), the Redwoods treebank has been created to provide annotated training material (Oepen et al., 2002).

For each utterance in Redwoods, analyses licensed by the ERG are enumerated and the correct one, if present, is indicated. Each analysis is represented as a tree that records the grammar rules which were used to derive it. For example, Figure 1a shows the preferred derivation tree, out of three analyses, for *what can I do for you?*.

Using these trees and the ERG, several different views of analyses can be recovered: phrase structures, semantic interpretations, and elementary dependency graphs. The phrase structures contain detailed HPSG non-terminals but are otherwise of the variety familiar from context-free grammar, as can be seen in Figure 1b.

Unlike most treebanks, Redwoods also provides semantic information for utterances. The semantic interpretations are expressed using Minimal Recursion Semantics (MRS) (Copestake et al., 2001), which provides the means to represent interpretations with a flat, underspecified semantics using terms of the predicate calculus and generalized quantifiers. An example MRS structure is given in Figure 2.

An elementary dependency graph is a simplified abstraction on a full MRS structure which uses no underspecification and retains only the major semantic predicates and their relations to one another.

In this paper, we report results using the third growth of Redwoods, which contains 5302 sentences for which there are at least two parses and for which a unique preferred parse is identified. These sentences have 9.3 words and 58.0 parses on average. Due to the small size of Redwoods and the underlying complexity of the system, exploring the effect of AL techniques for this domain is of practical, as well as theoretical, interest.

2.2 Modeling parse selection

As is now standard for feature-based grammars, we use log-linear models for parse selection (Johnson et al., 1999). Log-linear models are popular for their ability to incorporate a wide variety of features without making assumptions about their independence.[1]

For log-linear models, the conditional probability of an analysis t given a sentence with a set of analyses $\tau = \{t \ldots\}$ is given as:

$$P(t|s, M_k) = \frac{exp(\sum_{j=1}^{m} f_j(t) w_j)}{Z(s)} \tag{1}$$

where $f_j(t)$ returns the number of times feature j occurs in analysis t, w_j is a weight, $Z(s)$ is a normalization factor for the sentence, and M_k is a model. The parse with the highest probability is taken as the preferred parse for the model. We use the limited memory variable metric algorithm (Malouf, 2002) to determine the weights. Note that because the ERG usually only produces relatively few parses for in-coverage sentences, we can simply enumerate all parses and rank them.

The previous parse selection model (equation 1) uses a single model. It is possible to improve performance using an *ensemble* parse selection model. We create our ensemble model (called a *product model*) using the *product-of-experts* formulation (Hinton, 1999):

$$P(t|s, M_1 \ldots M_n) = \frac{\prod_{i=1}^{n} P(t|s, M_i)}{Z(s)} \tag{2}$$

Note that each individual model M_i is a well-defined distribution and is usually taken from a fixed set of models. $Z(s)$ is a normalization factor to ensure the product distribution sums to one over the set of possible parses. A product model effectively averages the contributions made by each of the individual models. Our product model, although simple, is sufficient to show enhanced performance when using multiple models. Of course, other ensemble techniques could be used instead.

[1]We also discussed perceptron models in Baldridge and Osborne (2003); here, we keep the model class fi xed to compare different AL methods.

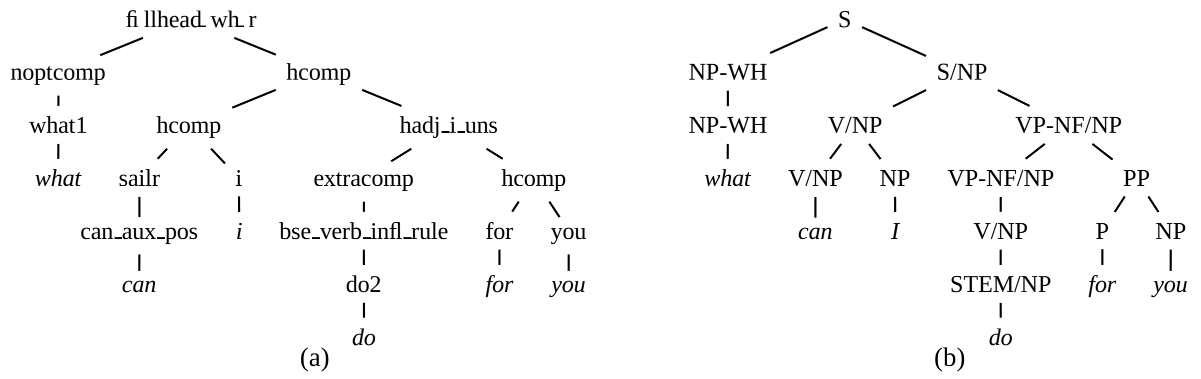


Figure 1: Example ERG derivation tree (a) and phrase structure tree (b).

$\langle h_1, e_2, \{h_5$:*which_rel*(BV:x_4,RESTR:h_6,SCOPE:h_8,DIM:v_7), h_9:*can_rel*(EVENT:e_2,ARG:v_{12},ARG4:h_{10},DIM:v_{11}),
h_{15}:*def_rel*(BV:x_{14},RESTR:h_{16},SCOPE:h_{17},DIM:v_{18}), h_{28}:*def_rel*(BV:x_{26},RESTR:h_{29},SCOPE:h_{30},DIM:v_{31}),
h_{19}:*do_rel*(EVENT:e_{20},ARG:v_{21},ARG1:x_{14},ARG3:x_4,ARG4:v_{23},DIM:v_{22}), h_1:*int_rel*(SOA:h_{32}), h_3:*thing_rel*(INST:x_4)
h_{19}:*for_rel*(EVENT:e_{25},ARG:e_{20},ARG3:x_{26},DIM:v_{24}), h_{27}:*pron_rel*(INST:x_{26}), h_{13}:*pron_rel*(INST:x_{14})$\}$,
$\{h_6 =_{qeq} h_3, h_{10} =_{qeq} h_{19}, h_{16} =_{qeq} h_{13}, h_{29} =_{qeq} h_{27}, h_{32} =_{qeq} h_9\}\rangle$

Figure 2: MRS structure for the sentence *what can I do for you?* The label of the entire structure is h_1 and the main event index is e_2. These are followed by a list of elementary predications, each of which is preceded by a label that allows it to be related to other predications. The final list is a set of constraints on how labels may be equated.

2.3 Three feature sets

Utilizing the various structures made available by Redwoods – derivation trees, phrase structures, MRS structures, and elementary dependency graphs – we create three distinct feature sets – **configurational**, **ngram**, and **conglomerate**. These three feature sets are used to train log-linear models. They incorporate different aspects of the parse selection task and so have different properties. This is crucial for creating diverse models for use in product ensembles as well as for the ensemble-based AL algorithms discussed in 4.

The configurational feature set is based on the derivation tree features described by Toutanova etal. (2003) and takes into account parent, grandparent, and sibling relationships among the nodes of the trees (such as that given in Figure 1(a)). The ngram set, described by Baldridge and Osborne (2003), also uses derivation trees; however, it uses a linearized representation of trees to create ngrams over the tree nodes. This feature creation strategy encodes many (but not all) of the relationships in the configurational set, and also captures some additional long-distance relationships.

The conglomerate feature set uses a mixture of features gleaned from phrase structures, MRS structures, and elementary dependency graphs. Each of these representations contains less information than that provided by derivation trees, but together they provide a different and comprehensive view on the ERG semantic analyses. The features contributed by phrase structures are simply ngrams of the kind described above for derivation trees. The features drawn from the MRS structures and elementary dependency graphs capture various dominance and co-occurrence relationships between nodes in the structures, as well as some global characteristics such as how many predications and nodes they contain.

2.4 Parse selection performance

Parse selection accuracy is measured using exact match, so a model is awarded a point if it picks some parse for a sentence and that parse is the correct analysis indicated in Redwoods. To deal with ties, the accuracy is given as $1/m$ when a model ranks m parses highest and the best parse is one of them.

Using the configurational, ngram, and conglomerate feature sets described in section 2.3, we create three log-linear models, which we will refer to as LL-CONFIG, LL-NGRAM, and LL-CONGLOM, respectively. We also create an ensemble model (called LL-PROD) with them using equation 2. The results for a chance baseline (selecting a parse at random), each of the three base models, and LL-PROD are given in Table 1. These are 10-fold cross-validation results, using all the training data when estimating models and the test split when evaluating them.

Though their overall accuracy is similar, the single models only agree about 80% of the time and performance varies by 3-4% between them on different folds of the cross-validation. Such variation is crucial for use in ensembles, and indeed, LL-PROD reduces the error rate

Model	Perf.	Model	Perf.
LL-CONFIG	75.05	LL-PROD	77.78
LL-NGRAM	74.01	Chance	22.70
LL-CONGLOM	74.85	–	–

Table 1: Parse selection accuracy.

of the best single model by 10.8%.[2]

Redwoods is different from other treebanks in that the treebank itself changes as the ERG is improved. LL-PROD's accuracy of 77.78% is the highest reported performance on version 3 of Redwoods. Results have also been presented for versions 1 (Baldridge and Osborne, 2003) and 1.5 (Oepen et al., 2002; Toutanova et al., 2003), both of which have considerably less ambiguity than version 3. Accordingly, LL-PROD's accuracy increases to 84.23% when tested on version 1.5, which has 3834 *ambiguous* sentences with an average length of 7.98 and average ambiguity of 11.05.

3 Measuring annotation cost

When evaluating AL methods, we compare methods based on two metrics: the absolute number of sentences they select (*unit cost*) and the summed number of decisions needed to select an individual preferred parse from a set of possible parses (*discriminant cost*). Unit cost is commonly used in AL research (Tang et al., 2002), but discriminant cost is more fine-grained.[3]

Discriminant cost works as follows. Annotation for Redwoods does not consist of actually drawing parse trees, and instead involves picking the correct parse out those produced by the ERG. To facilitate this task, Redwoods presents local *discriminants* which disambiguate large portions of the parse forest. This means that the annotator does not need to inspect all parses when specifying the intended analysis and so possible parses are narrowed down quickly even for sentences with a large number of parses. More interestingly, it means that the labeling burden is relative to the number of possible parses rather than the number of constituents in a parse. The discriminant cost of the examples we use averages 3.34 per sentence and ranges from 1 to 14.

We measure the discriminant cost of annotating a sentence s as the number of discriminants whose values were set by the human annotators in labeling that sentence in Redwoods plus one to reflect the final decision of selecting the preferred parse from the reduced parse forest.

Although we have not measured the cognitive burden on humans, we strongly believe that simply *selecting* the best parse is far more efficient than *drawing* the best parse for some sentence (as exemplified by Hwa (2000)). However, an interesting tension here is that we are committed to the ERG producing the intended parse within the set of analyses. When drawing a parse tree, by definition, the best parse is created. This may not be always true when using a manually written grammar such as the ERG.

4 Active learning methods

Suppose we have a set of examples and labels $D_n = \{\langle x^1, y^1\rangle, \langle x^2, y^2\rangle, \ldots\}$ which is to be extended with a new labeled example $\{\langle x^i, y^i\rangle\}$. The information gain for some model is maximized after selecting, labeling, and adding a new example x^i to D_n such that the noise level of x^i is low and both the bias and variance of some model using $D_n \cup \{\langle x^i, y^i\rangle\}$ is minimized (Cohn et al., 1995). If examples are selected for labeling using a strategy of minimizing either variance or bias, then typically, the error rate of a model decreases much faster than if examples are simply selected randomly for labeling.

In reality, selecting data points for labeling such that a model's variance and/or bias is maximally minimized is computationally intractable, so approximations are typically used instead. *Ensemble* methods can improve the performance of our active learners. An ensemble active learner uses more than one component model. For example, query-by-committee is an ensemble AL method, as is our generalization of uncertainty sampling.

In this section, we describe the AL methods that we tested on Redwoods, which include both single-model and ensemble-based AL techniques. Our single-method approaches are not meant to be exhaustive. In principle, there is no reason why we could not have also tried (within a kernel-based environment) selecting examples by their distance to a separating hyperplane (Tong and Koller, 2000) or else using the computationally demanding approach of Roy and McCallum (2001).

AL for parse selection is potentially problematic as sentences vary both in length and the number of parses they have. After experimenting with, and without, a variety of normalization strategies, we found that generally, there were no major differences overall. All of our methods therefore do not have any extra normalization.

In all our methods, τ denotes the set of analyses produced by the ERG for the sentence and M_k is some model. $\mathcal{M}$ is the set of models $M_1 \ldots M_n$.

4.1 Uncertainty sampling

Uncertainty sampling (also called *tree entropy* by Hwa (2000)), measures the uncertainty of a model over the set

[2]The product model is also better than a single model which uses all of the features of LL-CONFIG, LL-NGRAM, and LL-CONGLOM. The accuracy of the latter is 76.75%, so we achieve a 4.3% error reduction over this by using the product model.

[3]Hwa (2000) measured the number of constituents in a parse tree as another annotation cost. Our approach measures the cost of a more effi cient labelling strategy than Hwa's (tree drawing).

of parses of a given sentence, based on the conditional distribution it assigns to them. Following Hwa, we use the following measure to quantify uncertainty:

$$f_{us}(s, \tau, M_k) = -\sum_{t \in \tau} P(t|s, M_k) \log P(t|s, M_k) \quad (3)$$

Higher values of $f_{us}(s, \tau, M_k)$ indicate examples on which the learner is most uncertain and thus presumably are more informative. Calculating f_{us} is trivial with the conditional log-linear models described in section 2.2.

Uncertainty sampling as defined above is a single-model approach. It can be generalized to an ensemble by simply replacing the probability of a single log-linear model with a product probability:

$$f_{us}(s, \tau, \mathcal{M}) = -\sum_{t \in \tau} P(t|s, \mathcal{M}) \log P(t|s, \mathcal{M}) \quad (4)$$

4.2 Fixed Query-by-Committee

Another AL method is inspired by the *query-by-committee* (QBC) algorithm (Freund et al., 1997; Argamon-Engelson and Dagan, 1999). According to QBC, one should select data points when a group of models cannot agree as to the predicted labeling.

Using a fixed committee consisting of n distinct models, the examples we select for annotation are those for which the models most disagree on the preferred parse. One way of measuring this is with *vote entropy* (Argamon-Engelson and Dagan, 1999):[4]

$$f^{ve}_{qbc}(s, \tau) = -\frac{1}{\log \min(n, |\tau|)} \sum_{t \in \tau} \frac{V(t,s)}{n} \log \frac{V(t,s)}{n} \quad (5)$$

where $V(t, s)$ is the number of committee members that preferred parse t. QBC is inherently an ensemble-based method. We use a fixed set of models in our committee and refer to the resulting sample selection method as *fixed QBC*. Clearly there are many other possibilities for creating our ensemble, such as sampling from the set of all possible models.

4.3 Lowest best probability selection

Uncertainty sampling considers the overall shape of a distribution to determine how confident a model is for a given example. A radically simpler way of determining the potential informativity of an example is simply to consider the absolute probability of the most highly ranked parse. The smaller this probability, the less confident the model is for that example and the more useful it will be to know its true label.

We call this new method *lowest best probability* (LBP) selection, and calculate it as follows:

$$f_{lbp}(s, \tau, M_k) = \max_{t \in \tau} P(t \mid s, M_k) \quad (6)$$

LBP can be extended for use with an ensemble model in the same manner as uncertainty sampling (that is, replace the single model probability with a product).

5 Experiments

To test the effectiveness of the various AL strategies discussed in the previous section, we perform simulation studies of annotating version 3 of Redwoods.

For all experiments, we used a tenfold cross-validation strategy by randomly selecting 10% (roughly 500 sentences) from Redwoods for the test set and selecting samples from the remaining 90% of the corpus (roughly 4500 sentences) as training material. Each run of AL begins with a single randomly chosen annotated seed sentence. At each round, new examples are selected for annotation from a randomly chosen, fixed sized 500 sentence subset according to the method until the annotated training material made available to the learners contains at least 2000 examples and 8000 discriminants.[5] We select 20 examples for manual annotation at each round, and exclude all examples that have more than 500 parses. Other parameter settings did not produce substantially different results to those reported here.

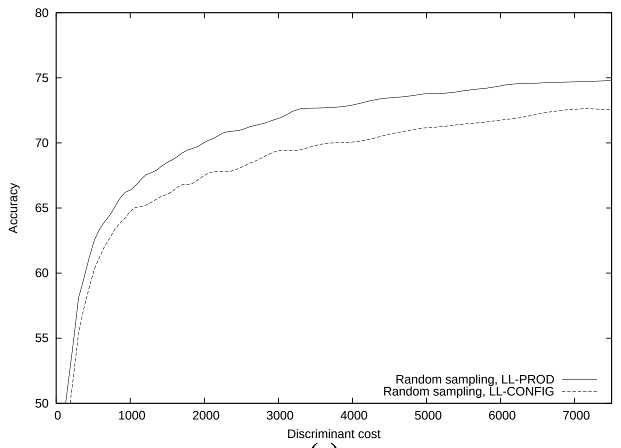
AL results are usually presented in terms of the amount of labeling necessary to achieve given performance levels. We say that one method is better than another method if, for a given performance level, less annotation is required. The performance metric used here is parse selection accuracy as described in section 2.4.

5.1 Baseline results

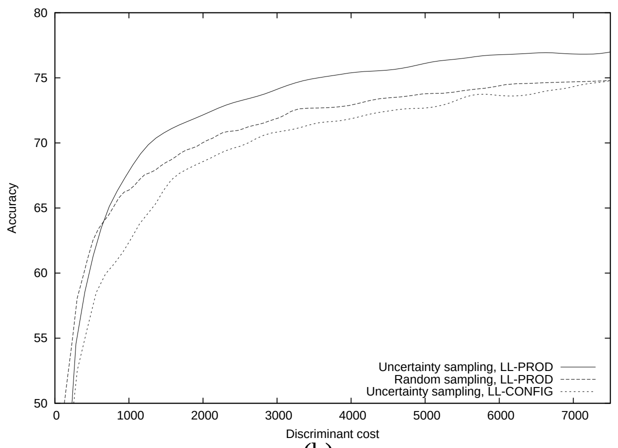
Frequently, baseline results are those produced by random sampling for a single model. Figure 3a shows a set of baseline results: LL-CONFIG (the best single model) using random sampling and the stronger baseline result of LL-PROD, also using random sampling. Quite clearly, we see that LL-PROD (which uses all three feature sets) outperforms LL-CONFIG. Although not shown, LL-PROD also outperforms LL-NGRAM and LL-CONGLOM trained using random sampling. These results show that the common practice in AL of only reporting the convergence results of a single model, trained using random sampling, can be misleading: we can improve upon the performance

[4] We experimented with *Kullback-Leibler divergence to the mean* (Pereira et al., 1993; McCallum and Nigam, 1998), but it performed no better than the simpler *vote entropy* metric.

[5] All of our AL methods reach full accuracy with this amount of material.



(a)



(b)

Figure 3: Accuracy as more annotation decisions are requested according to (a) random sampling with LL-CONFIG and LL-PROD, and (b) uncertainty sampling with LL-CONFIG and LL-PROD and random sampling with LL-PROD.

of a single model *without using AL* by using an ensemble model. Our main baseline system is therefore LL-PROD, trained progressively with randomly sampled examples.

5.2 Ensemble active learning results

Figure 3b compares uncertainty sampling using LL-CONFIG (the lower curve), random sampling using LL-PROD, and uncertainty sampling using LL-PROD.

The first thing to note is that random sampling for the ensemble outperforms uncertainty sampling for the single model. This shows that single model AL results can themselves be beaten by a model that does not use AL. Nonetheless, the graph also shows that an ensemble parse selection model using an ensemble AL method outperforms an ensemble parse selection model not using AL.

Table 2 shows the amount of labeling (as measured using our discriminant cost function) selected by some AL method necessary to achieve a given performance level. The top two methods are random baselines; the third method is uncertainty sampling using a single model, while the remaining three other methods are all ensemble active learners. There, and in the following text, labels of the form *rand-config* mean (in this case) using random sampling for LL-CONFIG; labels of the form *rand*-Π mean (again in this case) random sampling for LL-PROD; the legend *QBC* means using query-by-committee, with all three base models, when selecting examples for LL-PROD.

All three ensemble AL methods – product uncertainty sampling, QBC, and product LBP – provide large gains over random sampling (of all kinds). There is very little to distinguish the three methods, though product uncertainty sampling proves the strongest overall, providing a 53.6% reduction over rand-Π to achieve 77% accuracy and a 73.5% reduction over rand-config to reach 75% accuracy.

To understand whether product uncertainty is indeed choosing more wisely, it is important to consider the performance of an ensemble parse selection model when examples are chosen by a single-model AL method. That is, using a single-model AL method, but labeling examples using an ensemble model. If the ensemble AL method using the ensemble parse selection model performs equally to a single-model AL method also using an ensemble parse selection model, then the ensemble parse selection model would be responsible for improved performance. This contrasts with our ensemble AL method instead selecting more informative examples. We find that, as expected, selecting examples using LL-CONFIG for LL-PROD is worse than LL-PROD selecting for itself.

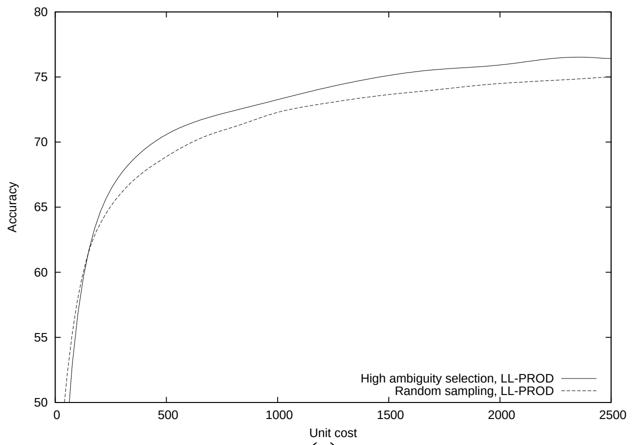
5.3 Simple selection metrics

Since sentences have variable length and ambiguity, there are four obvious selection metrics that make no use of AL methods: select sentences that are longer, shorter, more ambiguous or less ambiguous. We tested all four with LL-PROD and found none which improved on random sampling with the same model. For example, selecting the least ambiguous sentences performs the worst of all experiments we ran, with selection by shortest sentences close behind, respectively requiring 61.9% and 55.4% *increases* in discriminant cost over random sampling to reach 70% accuracy.

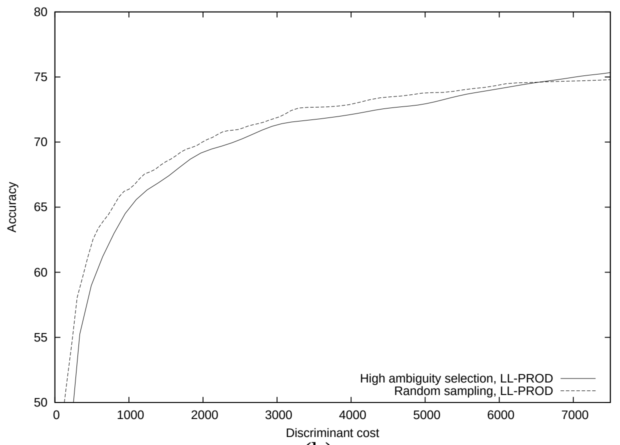
Selecting the most ambiguous examples dramatically demonstrates the difference between unit cost and discriminant cost. While that selection method requires a 17.4% increase in discriminant cost to reach 70%, it provides a 27.9% *reduction* in unit cost. Figure 4 compares (a) unit cost with (b) discriminant cost for ambiguity selection versus random sampling (with LL-PROD).

	70%			75%			77%	
		Reduction			Reduction			Reduction
	Cost	rand-config	rand-Π	Cost	rand-config	rand-Π	Cost	rand-Π
rand-config	3700	N/A	(46.2%)	13000	N/A	(36.2%)	N/A	N/A
rand-Π	1990	46.2%	N/A	8300	36.2%	N/A	13800	N/A
us-config	2600	29.7%	(25.2%)	7700	40.8%	7.2%	N/A	N/A
qbc	1300	64.9%	34.7%	3820	70.6%	54.0%	6780	50.9%
lbp-Π	**1280**	**65.4%**	**35.7%**	3660	71.9%	55.9%	7320	47.0%
us-Π	1300	64.9%	34.7%	**3450**	**73.5%**	**58.4%**	**6410**	**53.6%**

Table 2: Discriminant costs required for selection methods to reach 70%, 75%, and 77% accuracy. The reduction columns give the percentage reduction in cost compared to LL-CONFIG and LL-PROD using random sampling.



(a)



(b)

Figure 4: Comparison of cost metrics for selection by ambiguity for LL-PROD: (a) unit cost and (b) discriminant cost.

It is also important to consider sequential selection, a default strategy typically adopted by annotators. This was the worst of all AL methods, requiring an increase of 45.5% in discriminant cost over random sampling. This is most likely because the four sections of Redwoods come from two slightly different domains: appointment scheduling and travel planning dialogs. Because of this, sequential selection does not choose examples from the latter domain until all those from the former have been selected, and it thus lacks examples that are similar to those in the test set from the latter domain.

6 Related work

There is a large body of AL work in the machine learning literature, but less so within natural language processing. There is even less work on ensemble-based AL. Baram et al. (2003) consider selection of individual AL methods at run-time. However, their AL methods are only ever based on single model approaches.

Turning to parsing, most work has utilized uncertainty sampling (Thompson et al., 1999; Hwa, 2000; Tang et al., 2002). In all cases, relatively simple parsers were bootstrapped, and also, comparison was with a single model, trained using random sampling. As we pointed out earlier, our product model, not using AL, can outperform single-model active learning.

Baldridge and Osborne (2003) also applied AL to Redwoods. They only used two feature sets, did not consider product models, nor our simple LBP method. Additionally, they used the unit cost assumption.

Hwa et al. (2003) showed that for parsers, AL outperforms the closely related co-training, and that some of the labeling could be automated. However, their approach requires strict independence assumptions.

7 Discussion

We have shown that simple ensemble models can help both the underlying model and the AL method. Using a state-of-the-art parse selection model, we are able to achieve a 73% decrease in annotation costs compared against the highest performing single model trained using random sampling. This is one of the most substantial decreases in annotation cost reported in the literature. Our ensemble methods are very simple, and we expect that greater savings might follow when using more complex mode combination techniques such as boosting.

We expect our parse selection-specific results to improve if we present only the top n most highly ranked parses to the annotator, rather than the full set of parses. Provided the true best parse is within the top n with sufficient regularity, this would reduce the number of discriminants which the human annotator needs to consider when compared to unaided uncertainty sampling.

Another issue we will explore in future work is that for a scenario in which we label a data set from scratch, it is quite possible that we will not know how best to model the task we are labeling that data for. Thus, it is likely in such situations that we will be able to develop better evolved models only after the data is annotated and more has been learned about the task. It is then necessary to see whether improved models benefit from the examples selected using AL techniques with an earlier model more than they would have if random sampling had been used.

Acknowledgements

We would like to thank Markus Becker, Steve Clark, and the anonymous reviewers for their comments. Jeremiah Crim developed some of the feature extraction code and conglomerate features, and Alex Lascarides made suggestions for the semantic features. This work was supported by Edinburgh-Stanford Link R36763, ROSIE project.

References

Shlomo Argamon-Engelson and Ido Dagan. 1999. Committee-based sample selection for probabilistic classifi ers. *Journal of Artifi cial Intelligence Research*, 11:335–360.

Jason Baldridge and Miles Osborne. 2003. Active learning for HPSG parse selection. In *Proc. of the 7th Conference on Natural Language Learning*, Edmonton, Canada.

Y. Baram, R. El-Yaniv, and K. Luz. 2003. Online choice of active learning algorithms. In *Proc. of ICML-2003*, pages 19–26, Washington.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. 1995. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proc. of the 39th Annual Meeting of the ACL*, pages 132–139, Toulouse, France.

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

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

G. E. Hinton. 1999. Products of experts. In *Proc. of the 9th Int. Conf. on Artifi cial Neural Networks*, pages 1–6.

Rebecca Hwa, Miles Osborne, Anoop Sarkar, and Mark Steedman. 2003. Corrected Co-training for Statistical Parsers. In *Proceedings of the ICML Workshop "The Continuum from Labeled to Unlabeled Data"*, pages 95–102. ICML-03.

Rebecca Hwa. 2000. Sample selection for statistical grammar induction. In *Proc. of the 2000 Joint SIGDAT Conference on EMNLP and VLC*, pages 45–52, Hong Kong, China, October.

Mark Johnson, Stuart Geman, Stephen Cannon, Zhiyi Chi, and Stephan Riezler. 1999. Estimators for Stochastic "Unifi cation-Based" Grammars. In *37th Annual Meeting of the ACL*.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Proc. of the Sixth Workshop on Natural Language Learning*, pages 49–55, Taipei, Taiwan.

Andrew McCallum and Kamal Nigam. 1998. Employing EM and pool-based active learning for text classifi cation. In *Proc. of the International Conference on Machine Learning*.

Stephan Oepen, Kristina Toutanova, Stuart Shieber, Christopher Manning, Dan Flickinger, and Thorsten Brants. 2002. The LinGO Redwoods Treebank: Motivation and preliminary applications. In *Proc. of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proc. of the Annual Meeting of the ACL*.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA.

H. S. Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Computational Learning Theory*, pages 287–294.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active Learning for Statistical Natural Language Parsing. In *Proc. of the 40^{th} Annual Meeting of the ACL*, pages 120–127, Philadelphia, Pennsylvania, USA, July.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proc. 16th International Conf. on Machine Learning*, pages 406–414. Morgan Kaufmann, San Francisco, CA.

Simon Tong and Daphne Koller. 2000. Support vector machine active learning with applications to text classifi cation. In Pat Langley, editor, *Proc. of ICML-00, 17th International Conference on Machine Learning*, pages 999–1006, Stanford, US. Morgan Kaufmann Publishers, San Francisco, US.

Kristina Toutanova, Mark Mitchell, and Christopher Manning. 2003. Optimizing local probability models for statistical parsing. In *Proc. of 14th European Conf. on Machine Learning*.