Active Learning and the Total Cost of Annotation

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

Abstract

Active learning (AL) promises to reduce the cost of annotating labeled datasets for trainable human language technologies. Contrary to expectations, when creating labeled training material for HPSG parse selection and later *reusing* it with *other* models, gains from AL may be negligible or even negative. This has serious implications for using AL, showing that additional cost-saving strategies may need to be adopted. We explore one such strategy: using a model during annotation to automate some of the decisions. Our best results show an 80% reduction in annotation cost compared with labeling randomly selected data with a single model.

1 Introduction

AL methods such as uncertainty sampling (Cohn et al., 1995) or query by committee (Seung et al., 1992) have all been shown to dramatically reduce the cost of creating highly informative labeled sets for speech and language technologies. However, experiments using AL assume a model that is fixed ahead of time: the model used in AL is the same one we are currently developing training material for. For many complex tasks, we are unlikely to have a clear idea how best to model the task at the time of annotation; thus, in practice, we will need to *reuse* the labeled training material with *other* models.

In this paper, we show that AL can be brittle: under a variety of natural reuse scenarios (for example, allowing the later model to improve in quality, or else reusing the labeled training material using a different machine learning algorithm) performance of later models can be significantly undermined when training upon material created using AL. The key to knowing how well one model will be able to use material selected by another is their relatedness – yet there may be no means to determine this prior to annotation, leading to a chicken-and-egg problem.

Our reusability results thus demonstrate that, additionally, other strategies must be adopted to ensure we reduce the total cost of annotation. In Osborne and Baldridge (2004), we showed that ensemble models can increase model performance and also produce annotation savings when incorporated into the AL process. An obvious next step is automating some decisions. Here, we consider a simple automation strategy that reduces annotation costs *independently* of AL and examine its effect on reusability. We find that using both semi-automation and AL with high-quality models can eliminate the performance gap found in many reuse scenarios. However, for weak models, we show that semi-automation with random sampling is *more* effective for improving reusability than using it with AL – demonstrating further cause for caution with AL.

Finally, we show that under the standard assumption of reuse by the selecting model, using a strategy which combines AL, ensembling, and semi-automated annotation, we are able to achieve our highest annotation savings to date on the complex task of parse selection for HPSG: an 80% reduction in annotation cost compared with labeling randomly selected data with our best single model.

2 Parse selection for Redwoods

We now briefly describe the Redwoods treebanking environment (Oepen et al., 2002), our parse selection models and their performance.

2.1 The Redwoods Treebank

The Redwoods treebank project provides tools and annotated training material for creating parse selection models for the English Resource Grammar (ERG, Flickinger (2000)). The ERG is a hand-built broad-coverage HPSG grammar that provides an explicit grammar for the treebank. Using this approach has the advantage that analyses for within-coverage sentences convey more information than just phrase structure: they also contain derivations, semantic interpretations, and basic dependencies.

For each sentence, Redwoods records all analyses licensed by the ERG and indicates which of them, if any, the annotators selected as being contextually correct. When selecting such distinguished parses, rather than simply enumerating all parses and presenting them to the annotator, annotators make use of *discriminants* which disambiguate the parse forest more rapidly, as described in section 3.

In this paper, we report results using the third growth of Redwoods, which contains English sentences from appointment scheduling and travel planning domains of Verbmobil. In all, there are 5302 sentences for which there are at least two parses and a unique preferred parse is identified. These sentences have 9.3 words and 58.0 parses on average.

2.2 Modeling parse selection

As is now standard for feature-based grammars, we mainly use log-linear models for parse selection (Johnson et al., 1999). For log-linear models, the conditional probability of an analysis t_i given a sentence with a set of analyses $\tau = \{t \ldots\}$ is given as:

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

where $f_j(t_i)$ returns the number of times feature j occurs in analysis t, w_j is a weight from model M_k, and $Z(s)$ is a normalization factor for the sentence. The parse with the highest probability is taken as the preferred parse for the model. We use the limited memory variable metric algorithm to determine the weights. We do not regularize our log-linear models since labeled data -necessary to set hyperparameters- is in short supply in AL.

We also make use of simpler perceptron models for parse selection, which assign scores rather than probabilities. Scores are computed by taking the inner product of the analysis' feature vector with the parameter vector:

$$\text{score}(t_i, s, M_k) = \sum_{j=1}^{m} f_j(t_i)w_j \quad (2)$$

The preferred parse is that with the highest score out of all analyses. We do not use voted perceptrons here (which indeed have better performance) as for the reuse experiments described later in section 6 we really do wish to use a model that is (potentially) worse than a log-linear model.

Later for AL , it will be useful to map perceptron scores into probabilities, which we do by exponentiating and renormalizing the score:

$$P_p(t_i \mid s, M_k) = \frac{exp(\text{score}(t_i, s, M_k))}{Z(s)} \quad (3)$$

$Z(s)$ is again a normalizing constant.

The previous parse selection models (equations 1 and 3) use a single model (feature set). 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):

$$P(t_i|s, M_1 \ldots M_n) = \frac{\prod_{j=1}^{n} P(t_i|s, M_j)}{Z(s)} \quad (4)$$

Note that each individual model M_i is a well-defined distribution usually taken from a fixed set of models. $Z(s)$ is a constant 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. Though simple, this model is sufficient to show enhanced performance when using multiple models.

2.3 Parse selection performance

Osborne and Baldridge (2004) describe three distinct feature sets – **configurational**, **ngram**, and **conglomerate** – which utilize the various structures made available in Redwoods: derivation trees, phrase structures, semantic interpretations, and elementary dependency graphs. They incorporate different aspects of the parse selection task; this is crucial for creating diverse models for use in product parse selection models as well as for ensemble-based AL methods. Here, we also use models created from a subset of the conglomerate feature set: the **mrs** feature set. This only has features from the semantic interpretations.

The three main feature sets are used to train three log-linear models – LL-CONFIG, LL-NGRAM, and LL-CONGLOM— and a product ensemble of those three feature sets, LL-PROD, using equation 4. Additionally, we use a perceptron with the conglomerate feature set, P-CONGLOM. Finally, we include a log-linear model that uses the mrs feature set, LL-MRS, and a perceptron, P-MRS.

Parse selection accuracy is measured using exact match. A model is awarded a point if it picks some parse for a sentence and that parse is the correct analysis indicated by the corpus. 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.

The results for a chance baseline (selecting a parse at random), the base models and the product model are given in Table 1. These are 10-fold cross-validation results, using all the training data for estimation and the test split for evaluation. See section 5 for more details.

Model	Perf.	Model	Perf.
LL-CONFIG	75.05	LL-PROD	77.78
LL-NGRAM	74.01	LL-MRS	64.98
LL-CONGLOM	74.85	P-CONGLOM	73.00
Chance	22.70	P-MRS	62.11

Table 1: Parse selection accuracy.

3 Measuring annotation cost

To aid identification of the best parse out of all those licensed by the ERG, the Redwoods annotation environment provides local *discriminants* which the annotator can mark as true or false properties for the analysis of a sentence in order to disambiguate large portions of the parse forest. As such, the annotator does not need to inspect all parses and so parses are narrowed down quickly (usually exponentially so) 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).

Data about how many discriminants were needed to annotate each sentence is recorded in Redwoods. Typically, more ambiguous sentences require more discriminant values to be set, reflecting the extra effort put into identifying the best parse. We showed in Osborne and Baldridge (2004) that discriminant cost does provide a more accurate approximation of annotation cost than assigning a fixed unit cost for each sentence. We thus use discriminants as the basis of calculating annotation cost to evaluate the effectiveness of different experiment AL conditions.

Specifically, we set the cost of annotating a given sentence as the number of discriminants whose value were set by the human annotator plus one to indicate a final 'eyeball' step where the annotator selects the best parse of the few remaining ones.[1] The *discriminant cost* of the examples we use averages 3.34 and ranges from 1 to 14.

4 Active learning

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 $D_n \cup \{\langle x^i, y^i\rangle\}$ are minimized (Cohn et al., 1995).

In practice, 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. One such approximation is *uncertainty sampling*. Uncertainty sampling (also called *tree entropy* by Hwa (2000)), measures the uncertainty of a model over the set of parses of a given sentence, based on the conditional

[1] This eyeball step is not always taken, but Redwoods does not contain information about when this occurred, so we apply the cost for the step uniformly for all examples.

distribution it assigns to them. Following Hwa, we use the following measure to quantify uncertainty:

$$f_{us}(s, \tau, M_k) = -\sum_{t \in \tau} P(t|s, M_k) \log P(t|s, M_k) \quad (5)$$

τ denotes the set of analyses produced by the ERG for the sentence and M_k is some model. Higher values of $f_{us}(s, \tau, M_k)$ indicate examples on which the learner is most uncertain . Calculating f_{us} is trivial with the conditional log-linear and perceptrons models described in section 2.2.

Uncertainty sampling as defined above is a single-model approach. It can be improved by simply replacing the probability of a single log-linear (or perceptron) model with a product probability:

$$f^{en}_{us}(s, \tau, \mathcal{M}) = -\sum_{t \in \tau} P(t|s, \mathcal{M}) \log P(t|s, \mathcal{M}) \quad (6)$$

$\mathcal{M}$ is the set of models $M_1 \ldots M_n$. As we mentioned earlier, AL for parse selection is potentially problematic as sentences vary both in length and the number of parses they have. Nonetheless, the above measures do not use any extra normalization as we have found no major differences after experimenting with a variety of normalization strategies.

We use random sampling as a baseline and uncertainty sampling for AL. Osborne and Baldridge (2004) show that uncertainty sampling produces good results compared with other AL methods.

5 Experimental framework

For all experiments, we used a 20-fold cross-validation strategy by randomly selecting 10% (roughly 500 sentences) for the test set and selecting samples from the remaining 90% (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 random selection or uncertainty sampling until models reach certain desired accuracies. We select 20 examples for annotation at each round, and exclude all examples that have more than 500 parses.[2]

[2]Other parameter settings (such as how many examples to label at each stage) 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 2.3.

6 Reusing training material

AL can be considered as selecting some labeled training set which is 'tuned' to the needs of a particular model. Typically, we might wish to reuse labeled training material, so a natural question to ask is how general are training sets created using AL. So, if we later improved upon our feature set, or else improved upon our learner, would the previously created training set still be useful? If AL selects highly idiosyncratic datasets then we would not be able to reuse our datasets and thus it might, for example, actually be better to label datasets using random sampling. This is a realistic situation since models typically change and evolve over time — it would be very problematic if the training set itself inherently limits the benefit of later attempts to improve the model.

We use two baselines to evaluate how well a model is able to reuse data selected for labeling by another model: (1) **Selecting the data randomly.** This provides the essential baseline; if AL in reuse situations is going to be useful, it ought to outperform this model-free approach. (2) **Reuse by the AL model itself.** This is the standard AL scenario; against this, we can determine if reused data can be as good as when a model selects data for itself.

We evaluate a variety of reuse scenarios. We refer to the model used with AL as the *selector* and the model that is reusing that labeled data as the *reuser*. Models can differ in the machine learning algorithm and/or the feature set they use. To measure relatedness, we use Spearman's rank correlation on the rankings that two models assign to the parses of a sentence. The overall relatedness of two models is calculated as the average rank correlation on all examples tested in a 10-fold parse selection experiment using all available training material.

Figure 1 shows complete learning curves for LL-CONFIG when it reuses material selected by itself, LL-CONGLOM, P-MRS, and random sampling. The

graph shows that self-reuse is the most effective of all strategies – this is the idealized situation commonly assumed in active learning studies. However, the graph reveals that random sampling is actually more effective than selection both by LL-CONGLOM until nearly 70% accuracy is reached and by P-MRS until about 73%. Finally, we see that the material selected by LL-CONGLOM is always more effective for LL-CONFIG than that selected by P-MRS. The reason for this can be explained by the relatedness of each of these selector models to LL-CONFIG: LL-CONGLOM and LL-CONFIG have an average rank correlation of 0.84 whereas P-MRS and LL-CONFIG have a correlation of 0.65.

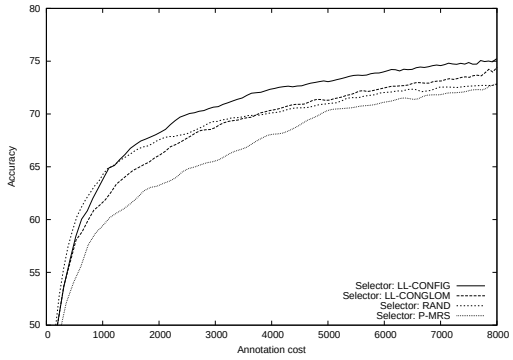

Figure 1: Learning curves for LL-CONFIG when reusing material by different selectors.

Table 2 fleshes out the relationship between relatedness and reusability more fully. It shows the annotation cost incurred by various reusers to reach 65%, 70%, and 73% accuracy when material is selected by various models. The list is ordered from top to bottom according to the rank correlation of the two models. The first three lines provide the baselines of when LL-PROD, LL-CONGLOM, and LL-CONFIG select material for themselves. The last three show the amount of material needed by these models when random sampling is used. The rest gives the results for when the selector differs from the reuser.

For each performance level, the percent increase in annotation cost over self-reuse is given. For example, a cost of 2300 discriminants is required for LL-PROD to reach the 73% performance level when it reuses material selected by LL-CONGLOM; this is a 10% increase over the 2100 discriminants needed when LL-PROD selects for itself. Similarly, the 5500 discriminants needed by LL-CONGLOM to reach 73% when reusing material selected by LL-CONFIG is a 31% increase over the 4200 discriminants LL-CONGLOM needs with its own selection.

As can be seen from Table 2, reuse always leads to an increase in cost over self-reuse to reach a given level of performance. How much that increase will be is in general inversely related to the rank correlation of the two models. Furthermore, considering each reusing model individually, this relationship is almost entirely inversely related at all performance levels, with the exception of P-CONGLOM and LL-MRS selecting for LL-CONFIG at the 73% level.

The reason for some models being more related to others is generally easy to see. For example, LL-CONFIG and LL-CONGLOM are highly related to LL-PROD, of which they are both components. In both of these cases, using AL for use by LL-PROD beats random sampling by a large amount.

That LL-MRS is more related to LL-CONGLOM than to LL-CONFIG is explained by the fact the **mrs** feature set is actually a subset of the **conglom** set. The former contains 15% of the latter's features. Accordingly, material selected by LL-MRS is also generally more reusable by LL-CONGLOM than to LL-CONFIG. This is encouraging since the case of LL-CONGLOM reusing material selected by LL-MRS represents the common situation in which an initial model – that was used to develop the corpus – is continually improved upon.

A particularly striking aspect revealed by Figure 1 and Table 2 is that random sampling is overwhelmingly a better strategy when there is still little labeled material. AL tends to select examples which are more ambiguous and hence have a higher discriminant cost. So, while these examples may be highly informative for the selector model, they are not cheap — and are far less effective when reused by another model.

Considering unit cost (i.e., each sentence costs the same) instead of discriminant cost (which assigns a variable cost per sentence), AL is generally more effective than random sampling for reuse throughout all accuracy levels – but not always. For example, even using unit cost, random sampling is better than selection by LL-MRS or P-MRS for reuse by

Selector	Reuser	Rank Corr.	65% DC	65% Incr	70% DC	70% Incr	73% DC	73% Incr
LL-PROD	LL-PROD	1.00	690	0.0%	1200	0.0%	2050	0.0%
LL-CONGLOM	LL-CONGLOM	1.00	1190	0.0%	2330	0.0%	4160	0.0%
LL-CONFIG	LL-CONFIG	1.00	1160	0.0%	2530	0.0%	4780	0.0%
LL-CONFIG	LL-PROD	.92	850	23.2%	1470	22.5%	2430	18.5%
LL-CONGLOM	LL-PROD	.92	840	21.7%	1560	30.0%	2630	28.3%
LL-CONFIG	LL-CONGLOM	.84	1340	12.6%	2610	12.0%	4720	13.5%
LL-CONGLOM	LL-CONFIG	.84	1660	43.1%	3760	48.6%	6840	43.1%
P-CONGLOM	LL-CONFIG	.79	1960	69.0%	3910	54.5%	7940	66.1%
LL-MRS	LL-CONGLOM	.77	1600	34.5%	3400	45.9%	6420	54.3%
LL-MRS	LL-PROD	.76	1080	56.5%	2040	70.0%	3700	80.5%
LL-MRS	LL-CONFIG	.71	2100	81.0%	4270	68.8%	6870	43.7%
P-MRS	LL-CONFIG	.65	2650	128.4%	4870	92.5%	8260	72.8%
RAND	LL-PROD	-	820	18.8%	1950	62.5%	3680	79.5%
RAND	LL-CONGLOM	-	1400	17.6%	3470	48.9%	7150	71.9%
RAND	LL-CONFIG	-	1160	0.0%	3890	53.8%	8560	79.1%

Table 2: Comparison of various selection and reuse conditions. Values are given for discriminant cost (DC) and the percent increase (Incr) in cost over use of material selected by the reuser.

LL-CONFIG until 67% accuracy. Thus, LL-MRS and P-MRS are so divergent from LL-CONFIG that their selections are truly sub-optimal for LL-CONFIG, particularly in the initial stages.

Together, these results shows that AL cannot be used blindly and always be expected to reduce the total cost of annotation. The data is tuned to the models used during AL and how useful that data will be for other models depends on the degree of relatedness of the models under consideration.

Given that AL may or may not provide cost reductions, we consider the effect that semi-automating annotation has on reducing the total cost of annotation when used with and without AL.

7 Semi-automated labeling

Corpus building, with or without AL, is generally viewed as selecting examples and then *from scratch* labeling such examples. This can be inefficient, especially when dealing with labels that have complex internal structures, as a model may be able to rule-out some of the labeling possibilities.

For our domain, we exploit the fact that we may already have partial information about an example's label by presenting only the top n-best parses to the annotator, who then navigates to the best parse within that set using those discriminants relevant to that set of parses. Rather than using a value for n that is fixed or proportional to the ambiguity of the sentence, we simply select all parses for which the model assigns a probability higher than chance. This has the advantage of reducing the number of parses presented to the annotator as the model uses more training material and reduces its uncertainty.

When the true best parse is within the top n presented to the annotator, the cost we record is the number of discriminants needed to identify it from that subset, plus one – the same calculation as when all parses are presented, with the advantage that fewer discriminants and parses need to be inspected.

When the best parse is *not* present in the n-best subset, there is a question as to how to record the annotation cost. The discriminant decisions made in reducing the subset are still valid and useful in identifying the best parse from the entire set, but we must incur some penalty for the fact that the annotator must confirm that this is the case. To determine the cost for such situations, we add one to the usual full cost of annotating the sentence. This encodes what we feel is a reasonable reflection of the penalty since decisions taken in the n-best phase are

still valid in the context of all parses.[3]

	Performance level		
	65%	70%	73%
1. RAND	820	1950	3680
2. LL-PROD	690	1200	2050
3. RAND (NB)	670	1350	2430
4. LL-PROD (NB)	680	1120	1760

Table 3: Cost for LL-PROD to reach given performance levels when using n-best automation (NB).

Table 3 shows the effects of using semi-automated labeling with LL-PROD. As can be seen, random selection costs reduce dramatically with n-best automation (compare rows 1 and 3). It is also an early winner over basic uncertainty sampling (row 2), though the latter eventually reaches the higher accuracies more quickly. Nonetheless, the mixture of AL and semi-automation provides the biggest overall gains: to reach 73% accuracy, n-best uncertainty sampling (row 4) reduces the cost by 17% over n-best random sampling (row 3) and by 15% over basic uncertainty sampling (row 2). Similar patterns hold for n-best automation with LL-CONFIG.

Figure 2 provides an overall view on the accumulative effects of ensembling, n-best automation, and uncertainty sampling in the ideal situation of reuse by the AL model itself. Ensemble models and n-best automation show that massive improvements can be made without AL. Nonetheless, we see the largest reductions by using AL, n-best automation, and ensemble models together: LL-PROD using uncertainty sampling and n-best automation (row 4 of Table 3) reaches 73% accuracy with a cost of 1760 compared to 8560 needed by LL-CONFIG using random sampling without automation. This is our best annotation saving: a cost reduction of 80%.

8 Closing the reuse gap

The previous section's semi-automated labeling experiments did not involve reuse. If models are expected to evolve, could n-best automation fill in the cost gap created by reuse? To test this, we considered reusing examples with our best model (LL-PROD), as selected by different models using both AL and n-best automation as a combined strategy. For LL-CONFIG and LL-CONGLOM as selectors, the gap is entirely closed: costs for reuse were virtually equal to when LL-PROD selects examples for itself *without* n-best (Table 3, row 2).

[3]When we do not allow ourselves to benefit from such labeling decisions, our annotation savings naturally decrease, but not below when we do not use n-best labeling.

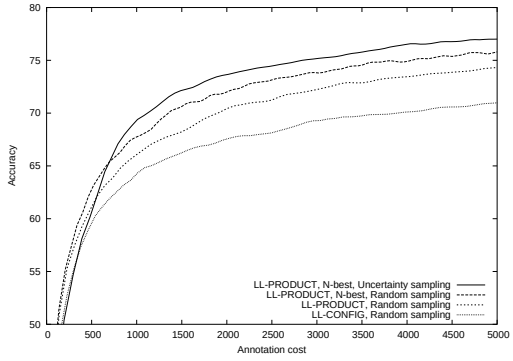

Figure 2: Learning curves for accumulative improvements to the annotation scenario starting from random sampling with LL-CONFIG: ensembling, n-best automation, and uncertainty sampling.

The gap also closes when n-best automation and AL are used with the weaker LL-MRS model. Performance (Table 4, row 1) still falls far short of LL-PROD selecting for itself *without* n-best (Table 3, row 2). However, the gap closes even more when n-best automation and random sampling are used with LL-MRS (Table 4, line 2).

	Performance level		
	65%	70%	73%
1. NB & US	1040	1920	3320
2. NB & RAND	680	1450	2890

Table 4: Cost for LL-PROD to reach given performance levels in reuse situations where n-best automation (NB) was used with LL-MRS with uncertainty sampling (US) or random sampling (RAND).

Interestingly, when using a weak selector (LL-MRS), n-best automation combined with random sampling was *more* effective than when combined with uncertainty sampling. The reason for this is clear. Since AL typically selects more ambiguous examples, a weak model has more difficulty getting

the best parse within the n-best when AL is used. Thus, the gains from the more informative examples selected by AL are surpassed by the gains that come with the easier labeling with random sampling.

For most situations, n-best automation is beneficial: the gap introduced by reuse can be reduced. n-best automation never results in an increase in cost. This is still true even if we do not allow ourselves to reuse those discriminants which were used to select the best parse from the n-best subset and the best parse was not actually present in that subset.

9 Related work

There is a large body of AL work in the machine learning literature, but less so within natural language processing (NLP). Most work in NLP has primarily focused upon uncertainty sampling (Hwa, 2000; Tang et al., 2002). Hwa (2001) considered reuse of examples selected for one parser by another with uncertainty sampling. This performed better than sequential sampling but was only half as effective as self-selection. Here, we have considered reuse with respect to many models and their co-relatedness. Also, we compare reuse performance against against random sampling, which we showed previously to be a much stronger baseline than sequential sampling for the Redwoods corpus (Osborne and Baldridge, 2004). 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.

10 Discussion

AL should only be considered for creating labeled data when the the task is either well-understood or else the model is unlikely to substantially change. Otherwise, it would be prudent to consider improving either the model itself (using, for example, ensemble techniques) or else semi-automating the labeling task. Naturally, there is a cost associated with creating the model itself, and this in turn will need to be factored into the total cost. When there is genuine uncertainty about the model, or else how the labeled data is going to be eventually used, then the best strategy may well be to use random selection rather than AL – especially when using some form of automated annotation.

Acknowledgments

We would like to thank Markus Becker, Jeremiah Crim, Dan Flickinger, Alex Lascarides, Stephan Oepen, and Andrew Smith. We'd also like to thank `pc-jbaldrid` and `pc-rosie` for their hard work and 24/7 dedication. This work was supported by Edinburgh-Stanford Link R36763, ROSIE project.

References

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.

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

G. E. Hinton. 1999. Products of experts. In *Proc. of the 9th Int. Conf. on Artificial 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 Conf. on EMNLP and VLC*, pages 45–52, Hong Kong, China.

Rebecca Hwa. 2001. On minimizing training corpus for parser acquisition. In *Proc. of the 5th Conference on Natural Language Learning*, Toulouse.

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

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.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proc. of HLT-NAACL*, Boston.

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.