Sample Selection for Statistical Grammar Induction *

Rebecca Hwa
Division of Engineering and Applied Sciences
Harvard University
Cambridge, MA 02138 USA
rebecca@eecs.harvard.edu

Abstract

Corpus-based grammar induction relies on using many hand-parsed sentences as training examples. However, the construction of a training corpus with detailed syntactic analysis for every sentence is a labor-intensive task. We propose to use sample selection methods to minimize the amount of annotation needed in the training data, thereby reducing the workload of the human annotators. This paper shows that the amount of annotated training data can be reduced by 36% without degrading the quality of the induced grammars.

1 Introduction

Many learning problems in the domain of natural language processing need supervised training. For instance, it is difficult to induce a grammar from a corpus of raw text; but the task becomes much easier when the training sentences are supplemented with their parse trees. However, appropriate supervised training data may be difficult to obtain. Existing corpora might not contain the relevant type of supervision, and the data might not be in the domain of interest. For example, one might need morphological analyses of the lexicon in addition to the parse trees for inducing a grammar, or one might be interested in processing non-English languages for which there is no annotated corpus. Because supervised training typically demands significant human involvement (e.g., annotating the parse trees of sentences by hand), building a new corpus is a labor-intensive task. Therefore, it is worthwhile to consider ways of minimizing the size of the corpus to reduce the effort spent by annotators.

* This material is based upon work supported by the National Science Foundation under Grant No. IRI 9712068. We thank Wheeler Ruml for his plotting tool; and Stuart Shieber, Lillian Lee, Ric Crabbe, and the anonymous reviewers for their comments on the paper.

There are two possible directions: one might attempt to reduce the amount of annotations in each sentence, as was explored by Hwa (1999); alternatively, one might attempt to reduce the number of training sentences. In this paper, we consider the latter approach using *sample selection*, an interactive learning method in which the machine takes the initiative of selecting potentially beneficial training examples for the humans to annotate. If the system could accurately identify a subset of examples with high *Training Utility Values* (TUV) out of a pool of unlabeled data, the annotators would not need to waste time on processing uninformative examples.

We show that sample selection can be applied to grammar induction to produce high quality grammars with fewer annotated training sentences. Our approach is to use *uncertainty-based* evaluation functions that estimate the TUV of a sentence by quantifying the grammar's uncertainty about assigning a parse tree to this sentence. We have considered two functions. The first is a simple heuristic that approximates the grammar's uncertainty in terms of sentence lengths. The second computes uncertainty in terms of the *tree entropy* of the sentence. This metric is described in detail later.

This paper presents an empirical study measuring the effectiveness of our evaluation functions at selecting training sentences from the Wall Street Journal (WSJ) corpus (Marcus et al., 1993) for inducing grammars. Conducting the experiments with training pools of different sizes, we have found that sample selection based on tree entropy reduces a large training pool by 36% and a small training pool by 27%. These results suggest that sample selection can significantly reduce human effort exerted in building training corpora.

2 Sample Selection

Unlike traditional learning systems that receive training examples indiscriminately, a learning system that uses sample selection actively influences its progress by choosing new examples to incorporate into its training set. Sample selection works with two types of learning systems: *a committee of learners* or *a single learner*. The committee-based selection algorithm works with multiple learners, each maintaining a different hypothesis (perhaps pertaining to different aspects of the problem). The candidate examples that led to the most disagreements among the different learners are considered to have the highest TUV (Cohn et al., 1994; Freund et al., 1997). For computationally intensive problems such as grammar induction, maintaining multiple learners may be an impracticality. In this work, we explore sample selection with a single learner that keeps just one working hypothesis at all times.

Figure 1 outlines the single-learner sample selection training loop in pseudo-code. Initially, the training set, L, consists of a small number of labeled examples, based on which the learner proposes its first hypothesis of the target concept, C. Also available to the learner is a large pool of unlabeled training candidates, U. In each training iteration, the selection algorithm, $Select(n, U, C, f)$, ranks the candidates of U according to their expected TUVs and returns the n candidates with the highest values. The algorithm computes the expected TUV of each candidate, $u \in U$, with an evaluation function, $f(u, C)$. This function may possibly rely on the hypothesis concept C to estimate the utility of a candidate u. The set of the n chosen candidates are then labeled by human and added to the existing training set. Running the learning algorithm, $Train(L)$, on the updated training set, the system proposes a new hypothesis consistent with all the examples seen thus far. The loop continues until one of three stopping conditions is met: the hypothesis is considered close enough to the target concept, all candidates are labeled, or all human resources are exhausted.

Sample selection may be beneficial for many learning tasks in natural language processing. Although there exist abundant collections of raw text, the high expense of manually annotating the text sets a severe limitation for many learning algorithms in natural language processing. Sample selection presents an attractive solution to offset this labeled data sparsity problem. Thus far, it has been successfully applied to several classification applications. Some examples include text categorization (Lewis and Gale, 1994), part-of-speech tagging (Engelson and Dagan, 1996), word-sense disambiguation (Fujii et al., 1998), and prepositional-phrase attachment (Hwa, 2000).

U is a set of unlabeled candidates.
L is a set of labeled training examples.
C is the current hypothesis.
Initialize:
    $C \leftarrow Train(L)$.
Repeat
    $N \leftarrow Select(n, U, C, f)$.
    $U \leftarrow U - N$.
    $L \leftarrow L \cup Label(N)$.
    $C \leftarrow Train(L)$.
Until $(C = C_{true})$ **or** $(U = \emptyset)$ **or** (human stops).

Figure 1: The pseudo-code for the sample selection learning algorithm

More difficult are learning problems whose objective is not classification, but generation of complex structures. One example in this direction is applying sample selection to semantic parsing (Thompson et al., 1999), in which sentences are paired with their semantic representation using a deterministic shift-reduce parser. Our work focuses on another complex natural language learning problem: inducing a stochastic context-free grammar that can generate syntactic parse trees for novel test sentences.

Although abstractly, parsing with a grammar can be seen as a classification task of determining the structure of a sentence by selecting one tree out of a set of possible parse trees, there are two major distinctions that differentiate it from typical classification problems. First, a classifier usually chooses from a fixed set of categories, but in our domain, every sentence has a different set of possible parse trees. Second, for most classification problems, the the number of the possible categories is relatively small, whereas the number of potential parse trees for a sentence is exponential with respect to the sentence length.

3 Grammar Induction

The degree of difficulty of the task of learning a grammar from data depends on the quantity and quality of the training supervision. When the training corpus consists of a large reservoir of fully annotated parse trees, it is possible to directly extract a grammar based on these parse trees. The success of recent high-quality parsers (Charniak, 1997; Collins, 1997) relies on the availability of such treebank corpora. To work with smaller training corpora, the learning system would require even more information about the examples than their syntactic parse trees. For instance, Hermjakob and Mooney (1997) have described a learning system that can build a deterministic shift-reduce parser from a small set of training examples with the aid of detailed morphological, syntactical, and semantic knowledge databases and step-by-step guidance from human experts.

The induction task becomes more challenging as the amount of supervision in the training data and background knowledge decreases. To compensate for the missing information, the learning process requires heuristic search to find locally optimal grammars. One form of partially supervised data might specify the phrasal boundaries without specifying their labels by bracketing each constituent unit with a pair of parentheses (McNaughton, 1967). For example, the parse tree for the sentence "*Several fund managers expect a rough market this morning before prices stablize.*" is labeled as "((Several fund managers) (expect ((a rough market) (this morning)) (before (prices stabilize))).)" As shown in Pereira and Schabes (1992), an essentially unsupervised learning algorithm such as the Inside-Outside re-estimation process (Baker, 1979; Lari and Young, 1990) can be modified to take advantage of these bracketing constraints.

For our sample selection experiment, we chose to work under the more stringent condition of partially supervised training data, as described above, because our ultimate goal is to minimize the amount of annotation done by humans in terms of both the number of sentences and the number of brackets within the sentences. Thus, the quality of our induced grammars should not be compared to those extracted from a fully annotated training corpus. The learning algorithm we use is a variant of the Inside-Outside algorithm that induces grammars expressed in the Probabilistic Lexicalized Tree Insertion Grammar representation (Schabes and Waters, 1993; Hwa, 1998). This formalism's context-free equivalence and its lexicalized representation make the training process efficient and computationally plausible.

4 Selective Sampling Evaluation Functions

In this paper, we propose two uncertainty-based evaluation functions for estimating the training utilities of the candidate sentences. The first is a simple heuristic that uses the length of a sentence to estimate uncertainties. The second function computes uncertainty in terms of the entropy of the parse trees that the hypothesis-grammar generated for the sentence.

4.1 Sentence Length

Let us first consider a simple evaluation function that estimates the training utility of a candidate without consulting the current hypothesis-grammar, G. The function $f_{len}(s, G)$ coarsely approximates the uncertainty of a candidate sentence s with its length:

$$f_{len}(s, G) = length(s).$$

The intuition behind this function is based on the general observation that longer sentences tend to have complex structures and introduce more opportunities for ambiguous parses. Since the scoring only depends on sentence lengths, this naive evaluation function orders the training pool deterministically regardless of either the current state of the grammar or the annotation of previous training sentences. This approach has one major advantage: it is easy to compute and takes negligible processing time.

4.2 Tree Entropy

Sentence length is not a very reliable indicator of uncertainty. To measure the uncertainty of a sentence more accurately, the evaluation function must base its estimation on the outcome of testing the sentence on the hypothesis-grammar. When a stochastic grammar parses a sentence, it generates a set of possible trees and associates a likelihood value with each. Typically, the most likely tree is taken to be the best parse for the sentence.

We propose an evaluation function that considers the probabilities of all parses. The set of probabilities of the possible parse trees for a sentence defines a distribution that indicates the grammar's uncertainty about the structure of the sentence. For example, a uniform distribution signifies that the grammar is at its highest uncertainty because all the parses are equally likely; whereas a distribution resembling an impulse function suggests that the grammar is very certain because it finds one parse much more likely than all others. To quantitatively characterize a distribution, we compute its entropy.

Entropy measures the uncertainty of assigning a value to a random variable over a distribution. Informally speaking, it is the expected number of bits needed to encode the assignment. A higher entropy value signifies a higher degree of uncertainty. At the highest uncertainty, the random variable is assigned one of n values over a uniform distribution, and the outcome would require $log_2(n)$ bits to encode.

More formally, let V be a discrete random variable that can take any possible outcome in set $\mathcal{V}$. Let $p(v)$ be the density function $p(v) = Pr(V = v), v \in \mathcal{V}$. The entropy $H(V)$ is the expected negative log likelihood of random variable V:

$$\begin{aligned} H(V) &= -EX(\log_2(p(V))). \\ &= -\sum_{v \in \mathcal{V}} p(v) \log_2(p(v)). \end{aligned}$$

Further details about the properties of entropy can be found in textbooks on information theory (Cover and Thomas, 1991).

Determining the parse tree for a sentence from a set of possible parses can be viewed as assigning a value to a random variable. Thus, a direct application of the entropy definition to the probability distribution of the parses for sentence s in grammar G computes its *tree entropy*, $TE(s, G)$, the expected number of bits needed to encode the distribution of possible parses for s. Note that we cannot compare sentences of different lengths by their entropy. For two sentences of unequal lengths, both with uniform distributions, the entropy of the longer one is higher. To normalize for sentence length, we define an evaluation function that computes the similarity between the actual probability distribution and the uniform distribution for a sentence of that length. For a sentence s of length l, there can be at most $O(2^l)$ equally likely parse trees and its maximal entropy is $O(l)$ bits (Cover and Thomas, 1991). Therefore, we define the evaluation function, $f_{te}(s, G)$ to be the tree entropy divided by the sentence length.

$$f_{te}(s, G) = \frac{TE(s, G)}{length(s)}.$$

We now derive the expression for $TE(s, G)$. Suppose that a sentence s can be generated by a grammar G with some non-zero probability, $Pr(s \mid G)$. Let $\mathcal{V}$ be the set of possible parses that G generated for s. Then the probability that sentence s is generated by G is the sum of the probabilities of its parses. That is:

$$Pr(s \mid G) = \sum_{v \in \mathcal{V}} Pr(v \mid G).$$

Note that $Pr(v \mid G)$ reflects the probability of one particular parse tree, v, in the grammar out of all possible parse trees for all possible sentences that G accepts. But in order to apply the entropy definition from above, we need to specify a distribution of probabilities for the parses of sentence s such that

$$\sum_{v \in \mathcal{V}} Pr(v \mid s, G) = 1.$$

$Pr(v \mid s, G)$ indicates the likelihood that v is the correct parse tree out of a set of possible parses for s according to grammar G. It is also the density function, $p(v)$, for the distribution (i.e., the probability of assigning v to a random variable V). Using Bayes Rule and noting that $Pr(v, s \mid G) = Pr(v \mid G)$ (because the existence of tree v implies the existence of sentence s), we get:

$$p(v) = Pr(v \mid s, G) = \frac{Pr(v, s \mid G)}{Pr(s \mid G)} = \frac{Pr(v \mid G)}{Pr(s \mid G)}.$$

Replacing the generic density function term in the entropy definition, we derive the expression for $TE(s, G)$, the tree entropy of s:

$$\begin{aligned} TE(s, G) &= H(V) \\ &= -\sum_{v \in \mathcal{V}} p(v) \log_2 p(v) \\ &= -\sum_{v \in \mathcal{V}} \frac{Pr(v \mid G)}{Pr(s \mid G)} \log_2\left(\frac{Pr(v \mid G)}{Pr(s \mid G)}\right) \\ &= -\sum_{v \in \mathcal{V}} \frac{Pr(v \mid G)}{Pr(s \mid G)} \log_2 Pr(v \mid G) \\ &\quad + \sum_{v \in \mathcal{V}} \frac{Pr(v \mid G)}{Pr(s \mid G)} \log_2 Pr(s \mid G) \end{aligned}$$

$$
\begin{aligned}
&= -\frac{\sum_{v\in\mathcal{V}} Pr(v \mid G)\log_2 Pr(v \mid G)}{Pr(s \mid G)} \\
&\quad + \log_2 Pr(s \mid G)\frac{\sum_{v\in\mathcal{V}} Pr(v \mid G)}{Pr(s \mid G)} \\
&= -\frac{\sum_{v\in\mathcal{V}} Pr(v \mid G)\log_2 Pr(v \mid G)}{Pr(s \mid G)} \\
&\quad + \log_2 Pr(s \mid G)
\end{aligned}
$$

Using the bottom-up, dynamic programming technique of computing Inside Probabilities (Lari and Young, 1990), we can efficiently compute the probability of the sentence, $Pr(s \mid G)$. Similarly, the algorithm can be modified to compute the quantity $\sum_{v\in\mathcal{V}} Pr(v \mid G)\log_2(Pr(v \mid G))$ (see Appendix A).

5 Experimental Setup

To determine the effectiveness of selecting training examples with the two proposed evaluation functions, we compare them against a baseline of random selection ($f_{rand}(s, G) = rand()$). The task is to induce grammars from selected sentences in the Wall Street Journal (WSJ) corpus, and to parse unseen test sentences with the trained grammars. Because the vocabulary size (and the grammar size by extension) is very large, we have substituted the words with their part-of-speech tags to avoid additional computational complexity in training the grammar. After replacing the words with part-of-speech tags, the vocabulary size of the corpus is reduced to 47 tags.

We repeat the study for two different candidate-pool sizes. For the first experiment, we assume that there exists an abundant supply of unlabeled data. Based on empirical observations (as will be shown in Section 6), for the task we are considering, the induction algorithm typically reaches its asymptotic limit after training with 2600 sentences; therefore, it is sufficient to allow for a candidate-pool size of $U = 3500$ unlabeled WSJ sentences. In the second experiment, we restrict the size of the candidate-pool such that U contains only 900 unlabeled sentences. This experiment studies how the paucity of training data affects the evaluation functions.

For both experiments, each of the three evaluation functions: f_{rand}, f_{len}, and f_{te}, is applied to the sample selection learning algorithm shown in Figure 1, where concept C is the current hypothesis-grammar G, and L, the set of labeled training data, initially consists of 100 sentences. In every iteration, $n = 100$ new sentences are picked from U to be added to L, and a new C is induced from the updated L. After the hypothesis-grammar is updated, it is tested. The quality of the induced grammar is judged by its ability to generate correct parses for unseen test sentences. We use the consistent bracketing metric (i.e., the percentage of brackets in the proposed parse not crossing brackets of the true parse) to measure parsing accuracy[1]. To ensure the statistical significance of the results, we report the average of ten trials for each experiment[2].

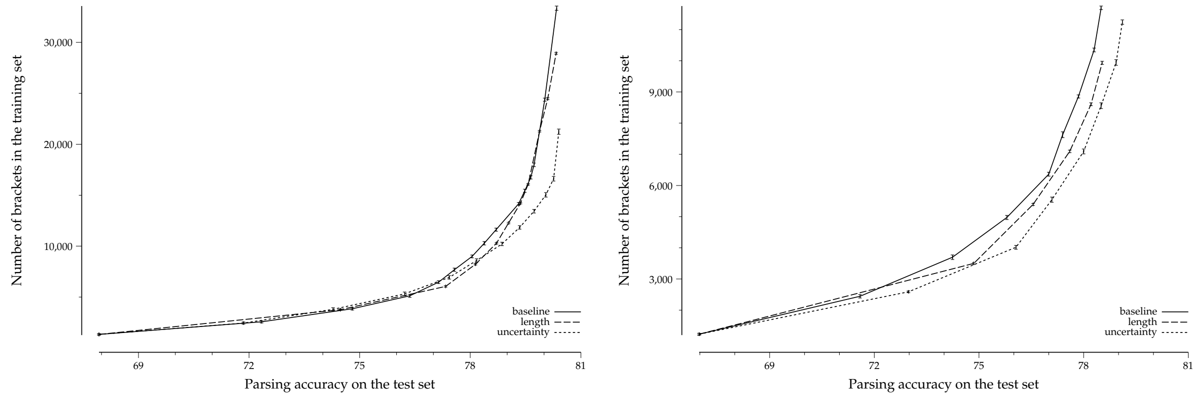
6 Results

The results of the two experiments are graphically depicted in Figure 2. We plot learning rates of the induction processes using training sentences selected by the three evaluation functions. The learning rate relates the quality of the induced grammars to the amount of supervised training data available. In order for the induced grammar to parse test sentences with higher accuracy (x-axis), more supervision (y-axis) is needed. The amount of supervision is measured in terms of the number of *brackets* rather than sentences because it more accurately quantifies the effort spent by the human annotator. Longer sentences tend to require more brackets than short ones, and thus take more time to analyze. We deem one evaluation function more effective than another if the smallest set of sentences it selected can train a grammar that performs at least as well as the grammar trained under the other function and if the selected data contains considerably fewer brackets than that of the other function.

Figure 2(a) presents the outcomes of the first experiment, in which the evaluation functions select training examples out of a large candidate-pool. We see that overall, sample selection has a positive effect on the learning

[1]The unsupervised induction algorithm induces grammars that generate binary branching trees so that the number of proposed brackets in a sentence is always one fewer than the length of the sentence. The WSJ corpus, on the other hand, favors a more flattened tree structure with considerably fewer brackets per sentence. The consistent bracketing metric does not unfairly penalize a proposed parse tree for being binary branching.

[2]We generate different candidate-pools by moving a fixed-size window across WSJ sections 02 through 05, advancing 400 sentences for each trial. Section 23 is always used for testing.



(a) (b)

Figure 2: The learning rates of the induction processes using examples selected by the three evaluation functions for (a) when the candidate-pool is large, and (b) when the candidate-pool is small.

grammar set	avg. training brackets	t-test on bracket avg.	avg. score	t-test on score avg.
baseline-26	33355	N/A	80.3	N/A
length-17	30288	better	80.3	not sig. worse
tree entropy-14	21236	better	80.4	not sig. worse

Table 1: Summary of pair-wise t-test with 95% confidence comparing the best set of grammars induced with the baseline (after 26 selection iterations) to the sets of grammars induced under the proposed evaluation functions (f_{len} after 17 iterations, f_{te} after 14 iterations).

rate of the induction process. For the baseline case, the induction process uses f_{rand}, in which training sentences are randomly selected. The resulting grammars achieves an average parsing accuracy of 80.3% on the test sentences after seeing an average of 33355 brackets in the training data. The learning rate of the tree entropy evaluation function, f_{te}, progresses much faster than the baseline. To induce a grammar that reaches the same 80.3% parsing accuracy with the examples selected by f_{te}, the learner requires, on average, 21236 training brackets, reducing the amount of annotation by 36% comparing to the baseline. While the simplistic sentence length evaluation function, f_{len}, is less helpful, its learning rate still improves slightly faster than the baseline. A grammar of comparable quality can be induced from a set of training examples selected by f_{len} containing an average of 30288 brackets. This provides a small reduction of 9% from the baseline [3]. We consider a set of grammars to be comparable to the baseline if its mean test score is at least as high as that of the baseline and if the difference of the means is not statistically significant (using pair-wise t-test at 95% confidence). Table 1 summarizes the statistical significance of comparing the best set of baseline grammars with those of of f_{len} and f_{te}.

Figure 2(b) presents the results of the second experiment, in which the evaluation functions only have access to a small candidate pool. Similar to the previous experiment, grammars induced from training examples selected by f_{te} require significantly less annotations than the baseline. Under the baseline, f_{rand}, to train grammars with 78.5% parsing accuracy on test data, an average of 11699 brackets (in 900 sentences) is required. In contrast, f_{te} can induce a comparable grammar with an average of 8559 brackets (in 600 sentences), providing a saving of 27% in the number of training brackets. The simpler evaluation function f_{len} out-performs the baseline as well; the 600 sentences it selected have an average of 9935 brackets. Table 2 shows the statistical significance of these comparisons.

A somewhat surprising outcome of the second study is that the grammars induced from

[3] In terms of the number of sentences, the baseline f_{rand} used 2600 randomly chosen training sentences; f_{len} selected the 1700 longest sentences as training data; and f_{te} selected 1400 sentences.

grammar set	avg. training brackets	t-test on bracket avg.	avg. score	t-test on score avg.
baseline-9	11699	N/A	78.5	N/A
length-6	9936	better	78.5	not sig. worse
tree entropy-6	8559	better	78.5	not sig. worse
tree entropy-8	11242	better	79.1	better

Table 2: Summary of pair-wise t-test with 95% confidence comparing the best set of grammars induced with the baseline (after 9 selection iterations) to the sets of grammars induced under the proposed evaluation functions (f_{len} after 6 iterations, f_{te} after 6 and 8 iterations).

the three methods did not parse with the same accuracy when all the sentences from the unlabeled pool have been added to the training set. Presenting the training examples in different orders changes the search path of the induction process. Trained on data selected by f_{te}, the induced grammar parses the test sentences with 79.1% accuracy, a small but statistically significant improvement over the baseline. This suggests that, when faced with a dearth of training candidates, f_{te} can make good use of the available data to induce grammars that are comparable to those directly induced from more data.

7 Conclusion and Future Work

This empirical study indicates that sample selection can significantly reduce the human effort in parsing sentences for inducing grammars. Our proposed evaluation function using tree entropy selects helpful training examples. Choosing from a large pool of unlabeled candidates, it significantly reduces the amount of training annotations needed (by 36% in the experiment). Although the reduction is less dramatic when the pool of candidates is small (by 27% in the experiment), the training examples it selected helped to induce slightly better grammars.

The current work suggests many potential research directions on selective sampling for grammar induction. First, since the ideas behind the proposed evaluation functions are general and independent of formalisms, we would like to empirically determine their effect on other parsers. Next, we shall explore alternative formulations of evaluation functions for the single-learner system. The current approach uses uncertainty-based evaluation functions; we hope to consider other factors such as confidence about the parameters of the grammars and domain knowledge. We also plan to focus on the constituent units within a sentence as training examples. Thus, the evaluation functions could estimate the training utilities of constituent units rather than full sentences. Another area of interest is to experiment with committee-based sample selection using multiple learners. Finally, we are interested in applying sample selection to other natural language learning algorithms that have been limited by the sparsity of annotated data.

References

James K. Baker. 1979. Trainable grammars for speech recognition. In *Proceedings of the Spring Conference of the Acoustical Society of America*, pages 547–550, Boston, MA, June.

Eugene Charniak. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the AAAI*, pages 598–603, Providence, RI. AAAI Press/MIT Press.

David Cohn, Les Atlas, and Richard Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, pages 16–23, Madrid, Spain.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley.

Sean P. Engelson and Ido Dagan. 1996. Minimizing manual annotation cost in supervised training from copora. In *Proceedings of the 34th Annual Meeting of the ACL*, pages 319–326.

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.

Atsushi Fujii, Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. 1998. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–598, December.

Ulf Hermjakob and Raymond J. Mooney. 1997. Learning parse and translation decisions from examples with rich context. In *Proceedings of the Association for Computational Linguistics*, pages 482–489.

Rebecca Hwa. 1998. An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *Proceedings of COLING-ACL*, volume 1, pages 557–563.

Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of 37th Annual Meeting of the ACL*, pages 73–79, June.

Rebecca Hwa. 2000. *Learning Probabilistic Lexicalized Grammars for Natural Language Processing*. Ph.D. thesis, Harvard University. Forthcoming.

K. Lari and S.J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Robert McNaughton. 1967. Parenthesis grammars. *Journal of the ACM*, 2(3):490–500.

Fernando Pereira and Yves Schabes. 1992. Inside-Outside reestimation from partially bracketed corpora. In *Proceedings of the 30th Annual Meeting of the ACL*, pages 128–135, Newark, Delaware.

Yves Schabes and Richard Waters. 1993. Stochastic lexicalized context-free grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 257–266.

Cynthia A. Thompson, Mary Elaine Califf, and Raymond J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of ICML-99*, pages 406–414, Bled, Slovenia.

A Efficient Computation of Tree Entropy

The tree entropy of a sentence depends on the quantity $\sum_{v\in\mathcal{V}} Pr(v \mid G)\log_2(Pr(v \mid G))$ described in Section 4.2, a sum of an exponential number of parses. Fortunately, through a dynamic programming algorithm similar to the computation of the Inside Probabilities, this quantity can be efficiently computed. The basic idea is to compose the tree entropy of the entire sentence from the tree entropy of the subtrees.

For illustrative purposes, we describe the computation process using a PCFG grammar expressed in Chomsky Normal Form, in which each rule can have two forms: $X \to YZ$ or $X \to a$, where X, Y, Z are variables over non-terminal symbols and a is a variable over terminal symbols. Moreover, let the symbol S be the start symbol of the grammar G. Following the notation of Lari and Young, we denote the inside probability as $e(X,i,j)$, which represents the probability that a non-terminal $X \stackrel{*}{\Rightarrow} w_i \ldots w_j$. Similarly, we define a new function $h(X,i,j)$ to represent the corresponding entropy for the set of subtrees.

$$h(X,i,j) = -\sum_{v\in X \stackrel{*}{\Rightarrow} w_i\ldots w_j} Pr(v \mid G)\log_2(Pr(v \mid G)).$$

Therefore, $\sum_{v\in\mathcal{V}} Pr(v \mid G)\log_2 Pr(v \mid G)$ can be expressed as $h(S,1,n)$.

We compute all possible $h(X,i,j)$ recursively. The base case is $h(X,i,i) = -e(X,i,i)\log_2(e(X,i,i))$ since a non-terminal X can generate the symbol w_i in exactly one way. For the more general case, $h(X,i,j)$, we consider all the possible rules with X on the left hand side that might have contributed to build $X \stackrel{*}{\Rightarrow} w_i \ldots w_j$.

$$h(X,i,j) = \sum_{k=i}^{j-1} \sum_{(X\to YZ)} h_{Y,Z,k}(X,i,j).$$

The function $h_{Y,Z,k}(X,i,j)$ is a portion of $h(X,i,j)$ where $Y \stackrel{*}{\Rightarrow} w_i \ldots w_k$ and $Z \stackrel{*}{\Rightarrow} w_{k+1} \ldots w_j$. The non-terminals Y and Z may, in turn, generate their substrings with multiple parses. Let there be α parses for $Y \stackrel{*}{\Rightarrow} w_i \ldots w_k$ and β parses for $Z \stackrel{*}{\Rightarrow} w_{k+1} \ldots w_j$. Let x denote the event of $X \to YZ$; $y \in y_1, \ldots, y_\alpha$; and $z \in z_1, \ldots, z_\beta$. The probability of one of the $\alpha \times \beta$ possible parses is $Pr(x)Pr(y)Pr(z)$, and $h_{Y,Z,k}$ is computed by summing over all possible parses:

$$\begin{aligned}
&h_{Y,Z,k}(X,i,j)\\
&\quad = -\textstyle\sum_{y,z} Pr(x)Pr(y)Pr(z)\times\\
&\qquad \log_2(Pr(x)Pr(y)Pr(z))\\
&\quad = -\textstyle\sum_{y,z} Pr(x)Pr(y)Pr(z)\times\\
&\qquad [\log_2 Pr(x) + \log_2 Pr(y) + \log_2 Pr(z)]\\
&\quad = -Pr(x)\log_2 Pr(x)e(Y,i,k)e(Z,k{+}1,j)\\
&\qquad +Pr(x)h(Y,i,k)e(Z,k+1,j)\\
&\qquad +Pr(x)e(Y,i,k)h(Z,k+1,j).
\end{aligned}$$

These equations can be modified to compute the tree entropy of sentences using a Probabilistic Lexicalized Tree Insertion Grammar (Hwa, 2000).