USING ZIPF FREQUENCIES AS A REPRESENTATIVENESS MEASURE IN STATISTICAL ACTIVE LEARNING OF NATURAL LANGUAGE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ONUR ÇOBANOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2008

Approval of the thesis:

USING ZIPF FREQUENCIES AS A REPRESENTATIVENESS MEASURE IN STATISTICAL ACTIVE LEARNING OF NATURAL LANGUAGE

submitted by **ONUR ÇOBANOĞLU** in partial fullfillment of the requirements for the degree of **Master of Science in Computer Engineering**, **Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Volkan Atalay _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Hüseyin Cem Bozşahin _____
Supervisor, **Department of Computer Engineering, METU**

Examining Committee Members:

Assist. Prof. Dr. Bilge Say _____
Department of Cognitive Sciences, METU

Assoc. Prof. Dr. Hüseyin Cem Bozşahin _____
Department of Computer Engineering, METU

Dr. Ayşenur Birtürk _____
Department of Computer Engineering, METU

Dr. Onur Tolga Şehitoğlu _____
Department of Computer Engineering, METU

Assist. Prof. Dr. Didem Gökçay _____
Department of Medical Informatics, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Onur Çobanoğlu

Signature :

ABSTRACT

USING ZIPF FREQUENCIES AS A REPRESENTATIVENESS MEASURE IN STATISTICAL ACTIVE LEARNING OF NATURAL LANGUAGE

Çobanoğlu, Onur

M.S., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. Hüseyin Cem Bozşahin

June 2008, 72 pages

Active learning has proven to be a successful strategy in quick development of corpora to be used in statistical induction of natural language. A vast majority of studies in this field has concentrated on finding and testing various informativeness measures for samples; however, representativeness measures for samples have not been thoroughly studied. In this thesis, we introduce a novel representativeness measure which is, being based on Zipf's law, model-independent and validated both theoretically and empirically. Experiments conducted on WSJ corpus with a wide-coverage parser show that our representativeness measure leads to better performance than previously introduced representativeness measures when used with most of the known informativeness measures.

Keywords: Active Learning, Grammar Induction, Natural Language Processing.

ÖZ

DOĞAL DİLLERİN İSTATİSTİKSEL ETKİN ÖĞRENMESİNDE ZİPF SIKLIKLARININ BİR TEMSİLİYET ÖLÇÜSÜ OLARAK KULLANILMASI

Çobanoğlu, Onur

Yüksek Lisans, Bilgisayar Mühendisligi Bölümü

Tez Yöneticisi: Doç. Dr. Hüseyin Cem Bozşahin

Haziran 2008, 72 sayfa

Etkin öğrenme, doğal dillerin istatistiksel edinimi amacıyla kullanılan külliyatların hızlı derlenmesinde başarılı bir strateji olduğunu ispatlamıştır. Bugüne kadar bu alandaki çalışmaların büyük bir çoğunluğu, örnekler için çeşitli bilgilendiricilik ölçüleri bulma ve bunları sınamaya odaklanmıştır; fakat örnekler için temsiliyet ölçüleri etraflıca araştırılmamıştır. Bu tezde, Zipf yasasına dayandığından ötürü modelden bağımsız ve hem kuramsal hem de deneysel olarak geçerli yeni bir temsiliyet ölçüsünü ortaya koymaktayız. Geniş kapsamlı bir ayrıştırıcı ile WSJ külliyatı üzerinde yapılan deneyler, ortaya koyduğumuz temsiliyet ölçüsünün, bilinen bilgilendiricilik ölçülerinin çoğuyla kullanıldığında daha önce ortaya atılmış temsiliyet ölçülerinden daha iyi sonuç verdiğini göstermektedir.

Anahtar Kelimeler: Etkin Öğrenme, Gramer Edinimi, Doğal Dil İşleme.

ACKNOWLEDGMENTS

First of all, I thank Cem Bozşahin for his helpful and supportive supervision.

I thank Jason Baldridge, Rebecca Hwa and Miles Osborne for their precious comments, discussions and encouragements.

I thank Volkan Atalay, Faruk Polat, İsmail Hakkı Toroslu and Fatoş Tünay Yarman Vural for guiding and encouraging young researchers. I also thank Meltem Turhan Yöndem for her guiding and support.

I thank TÜBİTAK-BİDEB for partially funding my M.S. study in the scope of 2210 National Graduate Scholarship Programme for M.Sc. students.

I thank members of the Laboratory for the Computational Studies of Language (especially Deniz Zeyrek and Ayışığı Başak Sevdik Çallı) for their efforts in getting CCGBank from Linguistic Data Consortium.

I thank my fellow graduate students, including Samet Akpınar, Hande Çelikkanat, Mehmet Remzi Doğar and Cüneyt Mertayak (and all others that I cannot remember right now) for their comments and encouragements, my office-mate Bahar Pamuk, our dedicated system administrators Can Eroğul, Gökdeniz Karadağ and Özgür Kaya and cluster administrators Ahmet Ketenci and Çelebi Kocair for their technical support, and all administrative and support staff of our department.

I thank NAR, the computer cluster of the Department of Computer Engineering in METU, for silently running my never ending experiments.

Special thanks to Ruken Çakıcı for her thesis-saving helps and advices. I also thank James Curran for his support on C&C Parser usage and modification.

Most of all, I thank my parents and Mehlika Merve Topçuoğlu for their invaluable love, support and understanding. Your great support made my scientific career (hence existence of this thesis) possible.

To my family and Merve...

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF ABBREVIATIONS

CCG	Combinatory Categorial Grammar
CKY	Cocke-Kasami-Younger
C&C	Clark & Curran
DUR	Data Utilization Ratio
GGP	Generalized Grouping Property
ILP	Inductive Logic Programming
LBP	Lowest Best Probability
LBS	Length Balanced Sampling
NLP	Natural Language Processing
OCR	Optical Character Recognition
PAC	Probably Approximately Correct
PAS	Predicate-Argument Structure
PCFG	Probabilistic Context Free Grammar
PER	Percentage Error Reduction
PLTIG	Probabilistic Lexicalized Tree Insertion Grammar
POS	Part-of-speech
PRUD	Percentage Reduction in Utilized Data
SUBG	Stochastic Unification Based Grammar
VC	Vapnik-Chervonenkis
WSJ	Wall Street Journal

CHAPTER 1

INTRODUCTION

Existence of computational models for natural languages is potentially a valuable source for many NLP tasks such as question-answering, information extraction and machine translation. Creation of such models by human computational linguists, however, remains to be a challenging problem due to the inherent complexity of human languages. The difficulty of this problem led many researchers to investigate the option of computers' inducing such models from corpus. Induction of statistical language models has been an active research field after Gold's negative learnability results for exact models (Gold, 1967) and Horning's probably approximately correct (PAC) learnability results for probabilistic context free grammars (Horning, 1969). Today, wide coverage statistical parsers have reached satisfactory accuracy levels in the existence of treebanks, including enormous amount of annotated data (parse trees, dependencies, etc.) prepared by human experts (see (Collins, 1997; Charniak, 1997; Charniak, 2000)). However, building a treebank is a "Herculean task" (Charniak, 1997). For instance, creating the first version of Penn Treebank, which included about 4.8 million words, all of which were part-of-speech (POS) tagged and about an half of which had skeletal syntactic trees (Marcus, Marcinkiewicz, and Santorini, 1993) required 12 person-years to be completed. Today there are only a few languages for which a treebank sufficient for wide-coverage parsing is built, and existing treebanks may fail to satisfy some needs due to absence of necessary kind of linguistic data.

In order to avoid the heavy burden of huge amount of human annotation, many alternatives of fully supervised learning have been tried. One of the most successful

approaches among them is active supervised learning (or active learning, referring to its most widely used name),[1] which is a relatively new stream in machine learning.[2] In active learning, contrary to classical supervised learning in which training samples are sampled randomly according to an input distribution, learner has control over training data stream. The motivation behind the discovery of active learning was the observation that random samples could (and in general did) contain redundant data. If a statistical learner had the ability of sampling the training data itself, it could be used to eliminate redundancy by sampling only informative data (or data which is expected to be informative in training).

Such an approach, though proven to be successful (see Chapter 2), has one important drawback: Since there is no longer randomness in sampling, we do not have the assurance that samples reflect the underlying statistical characteristics of the population from which the samples are drawn. Consequences of this problem have been seen in practice as well: Some researchers in the field of active learning of natural language have reported that since sentences with rare linguistic phenomena are found to be most informative by an active learner, active learning process tends to select outliers (see (Becker, 2003; Tang, Luo, and Roukos, 2002)). Becker (2008) gives the exaggerated example that a French sentence would be perceived to be very informative for an English parser, since training data of that parser would not possibly have included such a sentence before.

This is a general problem in active learning (not specific to statistical grammar induction domain). The problem could be solved optimally if the learner could know the exact probability distribution from which the samples are drawn, no estimate of which has been found in any domain of real-life active learning problems. Nevertheless, active learning researchers have exploited any information they could find about how

[1] We emphasized the term "active supervised learning" since active learning term is also used for active reinforcement learning methods based on exploration-exploitation. Such methods are designed in the spirit of agent-based AI and have almost nothing in common with active learning methods used in pattern recognition problems.

[2] Actually it is not new, for decades it has been considered as an option in machine learning literature. However, this consideration had been theoretical in general and emergence of its application to real-life machine learning tasks is relatively recent.

representative a sample is in the underlying probability distribution.

Of course, such information is often domain-specific; and some representativeness measures derived from such information have been proposed in natural language domain previously. In our work, we contribute to this line of research with a novel representativeness measure, based on a modification of Zipf's *Principle of Least Effort* (Zipf, 1949). We found this method especially compelling, since Zipf's laws – in general, hence including Principle of Least Effort – have been empirically the most successful sensible theories explaining the statistical characteristics of a language.

Since *representativeness* is orthogonal to *informativeness* conceptually, we conjecture that the comparison of various representativeness measures are meaningful only when they are compared with the same informativeness measure. Experiments conducted with C&C parser – a wide coverage CCG parser – on WSJ corpus have shown that, when used with most of the known informativeness measures in the literature, our measure leads to better performance than alternative representativeness measures. With some informativeness measures, our measure brings insignificant improvement; however, its performance never falls behind the performance of alternative representativeness measures.

1.1 Contributions of the Thesis

Novel contributions introduced by this thesis are as follows:

A Novel Computational Method for Calculating Tree Entropy In Both Generative and Discriminative Models

Tree entropy, introduced by Hwa (2000), has been perhaps the most widely used informativeness measure in the literature of active learning of probabilistic grammars. To the best of our knowledge, there is not a single work in the literature of active learning of natural language which is not using tree entropy as a comparison benchmark. Hwa (2000) introduces a CKY-based dynamic programming algorithm to calculate tree entropy over an exponential number of parses produced by a PCFG in polynomial time. We introduce an alternative algorithm, again based on CKY, which can calculate the tree entropy of a sentence having an exponential number of parses in polynomial time.

Our algorithm has advantages and disadvantages compared to Hwa's method, in the form of a time-space trade-off, which are described in Appendix A.

Testing and Comparison of Single-Learner Active Learning Methods In C&C Parser Domain

As we will review in detail in Section 2.2, there are several informativeness measures proposed for active learning in statistical grammar induction domain, and some of them are tested in various domains. However, to the best of our knowledge, no experimental results are evaluated for combinatory categorial grammar domain. In this work, we test the performance of known single-learner based informativeness measures and random sampling on WSJ corpus with C&C parser, which is a wide-coverage statistical parser based on CCG and conditional log-linear models. Results are presented in terms of labeled and unlabeled precision, recall and F_1 score of the recovery of predicate-argument dependencies from CCGbank (Hockenmaier and Steedman, 2005).

A Novel Representativeness Measure for Active Learning of Natural Language

As implied by the title of this thesis, this is the main contribution of our work. We propose to use a known frequency estimator for English sentences, which is based on a revision of Zipf's Principle of Least Effort (Zipf, 1949), as a representativeness measure for active learning of natural language. This estimator has the following desirable properties:

- The estimator has a theoretical basis (Zipf, 1935; Sigurd, Eeg-Olofsson, and van Weijer, 2004). Moreover, experiments on Brown Corpus (Kučera and Francis, 1967) have shown that estimated frequencies exhibit a very good fit to the observed frequencies (see Chapter 3).

- Since estimations are done only using sentence length, the measure is invariant from the parser used, underlying statistical model or state of the model/parser at any stage of active learning. Besides preventing deficiencies of the temporal state (or essence) of the parser/model from skewing representativeness estimates, this property provides another advantage: Our representativeness measure can

be used in active learning of other NLP tasks such as POS-tagging, supertagging etc.

1.2 Structure of the Thesis

The remainder of this thesis is structured as follows:

- In Chapter 2, we present a literature review on using active learning in statistical grammar induction domain. Firstly, after giving a general definition of active learning, types of active learning and three main active learning measures (informativeness, representativeness and diversity), along with active learning evaluation criteria are described. In addition, known theoretical results on the performance of active learning is presented. Finally, active learning measures for statistical parser training domain proposed up to now and their recorded experimental performances are presented.

- In Chapter 3, we explain the novel representativeness measure we mentioned in Section 1.1 in detail. This chapter is a quick review of a part of the work of Sigurd, Eeg-Olofsson, and van Weijer (2004). Mathematical formulation of the estimator, theoretical motivation and experimental results on Brown Corpus are presented.

- In Chapter 4, CCGbank corpus and C&C parser, which are used in all experiments in this thesis, are presented. This chapter is a quick review of (Clark and Curran, 2004b) and (Clark and Curran, 2007).

- In Chapter 5, tested active learning methods, experimental setup and results are presented. Information on experimental setup includes parser/model settings, pool information, batch construction specifics, performance evaluation metrics and sample selection schemes combining the information coming from our representativeness measure and tested informativeness measures. Presented results are comparative performances of single-learner active learning methods and comparative performances of each informativeness measure applied with other representativeness measures and our representativeness measure.

- In Chapter 6, the results are discussed and some possible research directions based on this thesis are proposed.

CHAPTER 2

ACTIVE LEARNING OF STATISTICAL GRAMMARS: A REVIEW

2.1 Active Learning

2.1.1 Definition and Types of Active Learning

Considering the variety of active learning studies done so far, we think that the best definition of active learning is that given in (Tong, 2001). In this definition, *active learning* is a *supervised learning* setting in which the learner can ask *queries* to an oracle and receive answers to its queries. Oracles are human trainers in general, but do not have to be humans. Queries can be requests for any kind of information that the oracle may respond. It can be, for example, the value of a feature of a sample (Liu, Motoda, and Yu, 2004). A more sophisticated example is found in (Angluin, 1987): When trying to learn a DFA, the learner can propose a DFA which the learner expects to be the target DFA. If it is not, then the learner may get a counterexample from oracle, which is a string accepted by the target DFA and rejected by the proposed DFA. New queries can be formed and asked by the learner, based on the information from previous query answers.

The most common type of queries that have been used in the literature so far is *membership queries*, with which the learner asks the *label* (or class, or concept, alternatively) of a sample to the oracle. The sample may be synthetic – built by the learner from scratch; but more often the sample is selected from a relatively large pool

U is a set of unlabeled candidates
L is a set of labeled training examples
C is the current hypothesis

$C \leftarrow Train(L)$ {Initialize}
repeat
  $N \leftarrow Select(n, U, C, f_{select})$
  $U \leftarrow U - N$
  $L \leftarrow L \cup Label(N)$
  $C \leftarrow Train(L)$
until $C = C_{true}$ or $U = \emptyset$ or human stops

Figure 2.1: The algorithm for selective sampling

of unlabeled samples, an approach which has generally been called *selective sampling* in the literature. Usually, an initial *seed training data*, which is sampled randomly, is used to estimate an initial model for the learner, which is used in selection of new samples from pool later. As new samples are added to the training data, the model of the learner is updated accordingly and new model is used in reranking of remaining unlabeled samples for selection. Algorithm of the process can be found in Figure 2.1, which is taken from (Hwa, 2000) and slightly modified.

Selective sampling has dominated the active learning research up to now. Building synthetic samples has lost its popularity in active learning research generally since the study of Lang and Baum (1992), in which a learner trained in OCR domain was shown to build freakish samples resembling neither of the known numerical characters in 10-base. While there have been some works proposing smarter queries (as we mentioned above), designing queries other than those inspecting membership is still an open area of research.

2.1.2 Sample Selection Criteria

As can be guessed from Figure 2.1, the aim of research in selective sampling is finding good f_{select} and $Select()$ functions. Actually it is the aim of research in membership query learning in general, since there should be some criteria to distinguish queries with higher *expected* contribution to the current hypothesis. In (Dan, 2004), three distinct criteria for a sample to be used by an active learning are identified:

- **Informativeness:** The idea of active learning is meaningful as long as the learner may determine the samples including useful information to improve current hypothesis. The learner cannot know the informational contribution of an unlabeled sample exactly beforehand, so it selects a sample according to some expectation of informational contribution of the sample. These expectations are called *informativeness measures*, and a vast majority of active learning research is focused on finding useful informativeness measures. There are so many distinct informativeness measures developed so far that they may easily fill a textbook.

- **Representativeness:** As we mentioned in Chapter 1, while abandoning randomness in selecting samples we take the risk of selecting exceptional samples which do not exhibit statistical characteristics of the population from which samples are drawn. This risk grows when samples are built from scratch – in selective sampling, we at least select a subsample of a random sample. The learner (even the oracle) cannot know the exact underlying distribution, so the best a learner can do is using an expectation of representativeness of a sample (namely, a *representativeness measure*). The dominating approach in developing representativeness measures have been using density estimation methods (see (McCallum and Nigam, 1998; Tang, Luo, and Roukos, 2002; Dan, 2004)), which requires a metric of similarity between samples. Of course, for such an approach to work, the learner must have access to a random sample (either labeled or unlabeled). Success of such an approach heavily depends on the quality of the similarity metric.

- **Diversity:** Unless the learner uses an online learning algorithm (in which case this criterion does not apply), active learner must retrain its internal model with samples selected at the previous stage. Considering the running time of machine learning algorithms and realistic data set sizes, selecting samples one by one is infeasible. As a result, all offline active learners select a *batch* of samples at each iteration of the active learning algorithm. This scheme brings the problem of selecting similar samples for the batch at an iteration. For example, the current labeled training data may be lacking samples including an unfamiliar event. In such a situation, an active learner finds the samples including that event to be most informative. However, if the learner fills the batch with only such samples,

it will be inefficient since only several such examples would suffice for learner to get familiar with them. So, in offline active learning, a measure of *diversity* of samples may help the learner select different kinds of samples for the batch. Recently, some works have appeared considering diversity of samples in active learning (see (Tang, Luo, and Roukos, 2002; Brinker, 2003; Dan, 2004)).

2.1.3 Theory of Active Learning

There have been many studies on the theoretical contribution of active learning, but many of them are domain and algorithm specific. There are too few general theoretical results that apply to a class of domains and methods.

In his seminal work, Gold (1967) introduced the concept of "identifiability in the limit", which is a general framework for learning a formal language. In this work, active learning is considered as an option and proved to be useless in the sense that if a language cannot be identified in the limit, it cannot be identified in the limit by utilizing active learning either.

In VC theory setting (Vapnik, 1999), more positive results have been found. The most trivial example is locating a boundary on the unit line interval ($[0, 1]$). Assuming uniform distribution of samples along unit line, $O(\frac{1}{\epsilon} ln \frac{1}{\epsilon})$ random samples are required to detect the boundary with an expected convergence error ϵ, while $O(ln \frac{1}{\epsilon})$ membership queries are sufficient for the same task. Dasgupta (2004) showed that the worst case sample complexity of finding the target linear seperator for m samples distributed on the unit sphere in $\mathbb{R}^d$ still requires querying the label of all m samples; however, the average sample complexity drops to $O(log\, m)$. The most generic theoretical result was found in (Eisenberg, 1992), a result which is shown to be valid for all *dense-in-itself* concept classes. According to its definition, dense-in-itself concept classes include half-spaces of $\mathbb{R}^n$, rectangles in the plane and thresholding function. The result shows that, if the learner has no prior knowledge about the probability distribution from which the samples come, then the ability of using membership queries improves the lower bound on the samples needed to PAC-learn a dense-in-itself concept class only by a constant factor, whatever the underlying probability distribution is. Only when the

learner knows a priori that the probability distribution generating samples is *smooth*[1], then the upper bound on the number of samples to learn a half space in n-dimensional unit simplex becomes $n^2(log(s/\epsilon)+4)$ while the lower bound on the number of samples to learn the same concept is $(1/4\epsilon)ln(1/\delta)$ with only random sampling, where ϵ and δ are general PAC-learning parameters and s is a constant depending on the probability distribution of samples.

2.1.4 Evaluation Criteria for Active Learning

The ultimate aim of active learning is either to reduce the cost of labeled data to reach a fixed performance level or to reach a better performance level with the same cost of labeled data. Cost of labeling a data set **D** is defined as the human effort needed to label all data in **D**. This quantity is surely directly relational with the amount of data to be labeled (which is itself a nontrivial concept), but not necessarily directly proportional. So the first step in evaluating an active learning method must be determining a realistic cost function. Some works have utilized experiments on human annotators labeling a given data set to determine or empirically validate cost functions (Hachey, Alex, and Becker, 2005; Baldridge and Osborne, 2008).

There are two general metrics used to evaluate active learning systems in general: *Data utilization ratio* (DUR) is the ratio of the cost of the data that active learning uses to the cost of the data that random sampling uses in order to reach a specified accuracy (accuracy is a performance metric for the classifier). *Percentage error reduction* (PER) is the measure of error reduction that active learning provides over random sampling, keeping the annotation cost fixed. It can be measured at a specific cost level, or can be averaged over all cost values. The lower the DUR is, or the higher the PER is, the better an active learning method is said to perform.

In the active learning literature, evaluation figures are generally given in terms of percentage reduction in the cost of data sampled actively, compared to the cost of data sampled randomly, in order to reach a certain accuracy level. This value is 1 – DUR,

[1] Mathematical definition of a smooth probability distribution function is given in (Eisenberg, 1992). For our purposes, it is enough to say that in a smooth p.d.f., there is a limit on accumulation of probability density to a small region, so the p.d.f is not highly irregular.

but it's so widely used that we will give it a seperate name, *Percentage Reduction in Utilized Data* (PRUD), and will use this name in the rest of the paper.

2.2 Active Learning in Statistical Parser Training

To the best of our knowledge, all works of active learning in statistical parser training has been in the selective sampling form. Using queries other than membership are proposed, like queries about local parse decisions proposed in (Hwa, 2004) and by Jason Baldridge in personal communication, but none has been realized yet. Using alternative queries is still an open research problem in this field. Building synthetic samples have not been proposed or tried up to now.

In this review, we classify the previous works according to their contribution to the literature of basic three active learning measures:

2.2.1 Proposed Informativeness Measures

2.2.1.1 Tree Entropy

Tree entropy, first proposed by Hwa (2000), is the Shannon entropy defined over probabilities of parses assigned by a probabilistic grammar to a sentence. More formally, given probabilistic grammar G and sentence s, tree entropy $f_{te}(s, G)$ is defined as:

$$f_{te}(s, G) = -\sum_{v \in V} p(v|s, G) \log_2 p(v|s, G) \tag{2.1}$$

Where V is the set of possible parses that G generates for s and $p(v|s, G)$ is the probability of a parse v conditioned on s and G.

Obviously, tree entropy is a very natural information-theoretic metric indicating the uncertainty of a stochastic parser for a given sentence. The motivation behind *uncertainty-based methods* – like tree entropy – is the expectation that the more a parser is uncertain about parses it assigns to a sentence, the more the actual parse of the sentence will possibly reveal some useful information for the parser. Selecting sentences having high tree entropies is an instance of *Maximum Entropy Sampling* (Shewry and Wynn, 1987), which is a generic active learning algorithm for probabilistic classifiers, dictating the

selection (or construction) of samples having highest classification entropies. Computation of tree entropy is straightforward if the probabilistic grammar G generates a tractable number of parses, which is often not the situation. Fortunately, a CKY-variant algorithm is supplied for computing tree entropy, which can work with any CKY-parsable grammar. In Appendix A, we propose an alternative algorithm for calculating tree entropy, which is again a CKY-variant and works with any probabilistic parser based on CKY. Our algorithm has advantages and disadvantages, compared to Hwa's method, which are explained in detail in Appendix A.

Since longer sentences tend to have more possible parses, they tend to have higher tree entropies; so it is conjectured that considering only tree entropy may lead to an unfair treatment of sentences with different lengths. To prevent this, Hwa first proposed to normalize the tree entropy of a sentence with the length of the sentence, since maximal tree entropy of a sentence having length l is $O(l)$ bits (Hwa, 2000). In her later work, Hwa (2001) proposed to use the binary logarithm of the number of possible parses of the sentence (which is the maximum possible tree entropy of the sentence as well) as the normalization factor, which normalizes tree entropy to [0,1] interval.

Tree entropy is the most widely tested informativeness measure in the literature of active learning of probabilistic grammars (see Table 2.1). First results came from Hwa herself (Hwa, 2000): Experiments conducted on WSJ corpus with Probabilistic Lexicalized Tree Insertion Grammar (PLTIG) (Schabes and Waters, 1993; Hwa, 1998) showed that tree entropy based active learning provides a PRUD varying between 27% and 36% according to the size of unlabeled sample pool (smaller pools lead to smaller PRUD). Used cost function is the total number of brackets (constituents) in the training data and consistent bracketing metric is used (the percentage of brackets in the proposed parse not crossing brackets of the true parse (Pereira and Schabes, 1992)) to measure parsing accuracy. Sentence length is used as the normalization factor in these experiments.

Table 2.1: Reported performances of tree entropy based selective sampling

Reported In	**Domain**	**Accuracy Metric**	**Cost Metric**	**Normalization**	**PRUD**
Hwa2000	PLTIG-WSJ	Consistent Bracketing	# of brackets	sentence length	27%-36%
Hwa2001	PLTIG-WSJ	Consistent Bracketing	# of brackets	$log_2(\#\,of\,parses)$	33%
Hwa2001	Collins 2-WSJ	Labeled F_1	# of constituents	$log_2(\#\,of\,parses)$	23%
Tang2002	CLASSER-DARPA	Exact Match (%)	# of sentences	sentence length	67%
Tang2002	CLASSER-DARPA	Exact Match (%)	# of sentences	None	67%
Baldridge2003	Redwoods-HPSG-ERG	True Selection (%)	# of sentences	sentence length	51.7%
Osborne2004	Redwoods-HPSG-ERG	True Selection (%)	# of discriminants	None	58.4%
Hwa2004	PLTIG-WSJ	Consistent Bracketing	# of brackets	$log_2(\#\,of\,parses)$	50%
Hwa2004	Collins 2-WSJ	Labeled F_1	# of constituents	$log_2(\#\,of\,parses)$	27%
Becker2005	Collins 2-WSJ	F_1	# of constituents	None (LBS)	-5.7%
Baldridge2008	Redwoods-HPSG-ERG	Exact Match (%)	# of discriminants	None	24% (avg.)

Hwa (2001) conducted experiments on WSJ corpus again with PLTIG and Model 2 Collins parser (Collins, 1997). Experiments showed that using tree entropy based active learning provided 23% PRUD to achieve 88.7% labeled F_1 score with Collins 2 parser and 33% PRUD to achieve the best performance achieved by random sampling with PLTIG parser (cost function was the total number of constituents). When samples selected by a Collins 2 parser trained simultaneously are used to train the PLTIG parser, 15% PRUD is observed to achieve the best performance achieved by random sampling. In this work, binary logarithm of the number of possible parses of the sentence is used as the normalization factor.

In (Tang, Luo, and Roukos, 2002), both non-normalized tree entropy and tree entropy normalized with sentence length are tested, both alone and along with some representativeness and informativeness measures (see Section 2.2.2 and 2.2.3), with shallow semantic parser CLASSER (Davies et al., 1999) on the DARPA Communicator domain. They used the percentage of exact matches between parses assigned by the human annotator and the learner as accuracy measure and the number of sentences as the cost measure. Performance of tree entropy varied with whether it is used with representativeness and diversity metrics or whether it is normalized, but on average approximately 67% PRUD is achieved. Normalized tree entropy performed worse than non-normalized tree entropy, justification of which given by the authors is that the normalized tree entropy is unnaturaly high in short sentences (contrary to non-normalized version). Their experiments show that when diversity is considered as well, normalized and non-normalized tree entropy perform almost equally, since longer sentences are considered as well in this setting even if they have lower normalized tree entropy scores. We conjecture that the tendency of normalized tree entropy to be unnaturally high in shorter sentences is not surprising, considering that when sentence length is used as the normalization factor, longer sentences are unfairly penalized since the actual number of parses may be far less than the maximum possible number of parses in longer sentences. We conjecture that using binary logarithm of actual number of parses as the normalization factor will not penalize any sentence unfairly because of its length.

Baldridge and Osborne (2003) applied active learning to *parse disambiguation* (or *parse selection*, equivalently) domain, rather than grammar induction domain. Aim

of the work was extending Redwoods treebank (Oepen et al., 2002), which was a corpus including automatically created parses of sentences according to English Resource Grammar (ERG) (Flickinger, 2000), which was a hand-crafted broad coverage HPSG grammar of English, along with the true (or preferred) parse(s) of each sentence among all alternatives created by ERG. Motivation behind the creation of the Redwoods was supplying annotated material for training of statistical models used in disambiguation of HPSG parses created by ERG. As the performance measure, number of sentences with exactly true parse selections is taken (when sentence is ambiguous and there are m true parses, if model selects one of the true parses score of this is counted as $1/m$). Cost function is the number of sentences. Experiments conducted with conditional log-linear models on Redwoods showed that tree entropy (normalized with sentence length) provided 51.7% PRUD at the performance level achieved by random sampling with the largest training data used in experiments. Combining tree entropy with an active ensemble learning method (preferred parse disagreement) provide 60% PRUD at the same performance level. Consistent with the reported results in (Hwa, 2001), training a model with samples selected by another simultaneously trained model gave lower PRUD values, though still introducing improvement over random sampling performance.

In (Osborne and Baldridge, 2004), tree entropy is tested again (apparently in non-normalized way) on the same Redwoods-HPSG-ERG domain, but this time ensemble models along with a novel and more realistic cost metric named *discriminant cost*, details of which can be found in the paper, are utilized. Experiments showed that with ensemble models, tree entropy based sampling provided up to 58.4% PRUD, compared to the random sampling counterparts using same models.

In (Hwa, 2004), tree entropy is tested with PLTIG and Model 2 Collins parser on WSJ corpus again. PLTIG annotations included phrasal boundaries of the POS-tagged sentence. Cost function was the total number of labeled consituents for Collins parser and the total number of labeled brackets for PLTIG parser. Accuracy metric was consistent bracketing metric for PLTIG parser and labeled F_1 score for Collins parser. Tree learning provided 50% PRUD in PLTIG induction and 27% PRUD in training Collins parser. Author explains this difference with more lexicalized nature of the Collins parser.

In (Becker and Osborne, 2005), tree entropy is tested with nearly the same experimental setting, accuracy and cost functions as in (Hwa, 2004); however, it is reported that purely tree entropy based sample selection leads to -5.7% PRUD (that is, it performs worse than random sampling). In these experiments tree entropy is not normalized, however, it is used along with *length balanced sampling* (LBS) (see Section 2.2.2 below).

In (Baldridge and Osborne, 2008), tree entropy is tested again in Redwoods 5-HPSG-ERG domain, with various feature sets and their ensembles. Performance of tree entropy varied dramatically with the used feature set, on average 24% PRUD is reported.

2.2.1.2 Unparsed

Unparsed method gives priority to sentences that cannot be parsed by the current state of the parser while selecting samples. If sentences are selected one by one it is trivial; but if a batch of sentence is selected at a step, then there should be a secondary selection criterion in case the sentences that cannot be parsed do not suffice and more sentences are required to fill the batch.

Though proposed as early as 1999 for an ILP-based natural language parser (Thompson, Califf, and Mooney, 1999), *unparsed* had not been tried in statistical parser training domain until 2005 (Becker and Osborne, 2005). In (Becker and Osborne, 2005), two schemes of filling the rest of the batch (in case the number of unparsed sentences falls below the batch size) are introduced: One is filling the rest by random sampling (a method which authors called *unparsed*, but we will call it *unparsed/random* for clarity) and the other is filling the rest with sentences having highest tree entropies (a method which authors called *unparsed/entropy*). These methods were tested with Collins 2 parser on WSJ corpus, with F_1 score as the accuracy metric and number of constituents as the cost metric. Length balanced sampling was used as the representativeness measure (see Section 2.2.2). In these experimental setup, 29.4% PRUD was achieved by unparsed and 30.6% PRUD was achieved by unparsed/entropy to reach 80% F_1 score (results are summarized in Table 2.2)

Table 2.2: Reported performances of unparsed based selective sampling

Reported In	Domain	Accuracy Metric	Cost Metric	Secondary Selection	PRUD
Becker2005	Collins 2-WSJ	F_1	# of constituents	random	29.4%
Becker2005	Collins 2-WSJ	F_1	# of constituents	tree entropy	30.6%

2.2.1.3 Lowest Best Probability (LBP)

First proposed in (Osborne and Baldridge, 2004), *lowest best probability* method selects samples having the lowest probability values for their most probable parses. More formally, given a sentence s and probabilistic grammar G, the lowest best probability metric $f_{lbp}(s, G)$ is defined as:

$$f_{lbp}(s, G) = max_{v \in V} P(v|s, G) \tag{2.2}$$

Where V is the set of parses of s found by G. Authors conjecture that f_{lbp} is a good measure of uncertainty, like tree entropy (hence it is an uncertainty-based method).

Summary of test results are listed in Table 2.3. First test reports of lowest best probability appeared on the paper in which this metric is proposed for the first time (Osborne and Baldridge, 2004). Experiments are conducted in Redwoods-HPSG-ERG domain, with number of discriminants as the cost metric and percentage of exactly true parse selections as the accuracy metric. It is reported that LBP based selective sampling provided up to 55.9% PRUD, compared to the random sampling counterparts using the same model.

In (Hwa, 2004), lowest best probability is tested (in the paper, *error-driven evaluation* is used instead of the term LBP, but essence is the same) on WSJ corpus with PLTIG and Collins 2 parsers. PLTIG annotations included phrasal boundaries of the POS-tagged sentence. Cost function was total number of labeled consituents for Collins parser and total number of labeled brackets for PLTIG parser. Accuracy metric was consistent bracketing metric for PLTIG parser and labeled F_1 score for Collins parser. LBP provided 45% PRUD to reach 80% accuracy with PLTIG parser and 17% PRUD to reach 88% accuracy with Collins 2 parser.

Table 2.3: Reported performances of lowest best probability based selective sampling

Reported In	Domain	Accuracy Metric	Cost Metric	PRUD
Osborne2004	Redwoods-HPSG-ERG	True Selection (%)	# of discriminants	55.9%
Hwa2004	PLTIG-WSJ	Consistent Bracketing	# of brackets	45%
Hwa2004	Collins 2-WSJ	Labeled F_1	# of constituents	17%
Baldridge2008	Redwoods-HPSG-ERG	Exact Match (%)	# of discriminants	28% (avg.)

In (Baldridge and Osborne, 2008), LBP is tested again in Redwoods 5-HPSG-ERG domain, with various feature sets and their ensembles. Performance of LBP varied dramatically with the used feature set, on average 28% PRUD is reported.

2.2.1.4 Two-Stage Active Learning

Conjecturing that tree entropy based uncertainty sampling fails to catch informative samples when selection decision requires statistics about low frequency events, Becker and Osborne (2005) proposes a two-stage approach in order to incorporate samples including low frequency events into selective sampling process. In the first stage, available samples are first parsed by a *bagged* parser (Breiman, 1996) and sentences that cannot be parsed by bagged parser are selected for sampling by the learner. If unparsable sentences fail to fill the batch, as a second stage remaining of the batch is filled by remaining sentences having highest tree entropies. Since bagged training set is expected to be relatively free of low frequency events, sentences that cannot be parsed by bagged parser are likely to include low frequency events, which are conjectured to be more informative than samples selected according to uncertainty.

Two-stage active learning is tested with Collins 2 parser on WSJ corpus, as reported in (Becker and Osborne, 2005), with F_1 score as the accuracy metric and number of constituents as the cost metric. Length balanced sampling was used as the representativeness measure (see Section 2.2.2). In this setting, two-stage active learning provides 32.6% PRUD to reach 85.5 F score.

Table 2.4: Reported performances of sentence length based selective sampling

Reported In	Domain	Accuracy Metric	Cost Metric	PRUD
Hwa2000	PLTIG-WSJ	Consistent Bracketing	# of brackets	9%
Baldridge2003	Redwoods-HPSG-ERG	True Selection (%)	# of sentences	12%
Osborne2004	Redwoods-HPSG-ERG	True Selection (%)	# of discriminants	-55.4%
Hwa2004	PLTIG-WSJ	Consistent Bracketing	# of brackets	26%
Hwa2004	Collins 2-WSJ	Labeled F_1	# of constituents	10%
Baldridge2008	Redwoods-HPSG-ERG	Exact Match (%)	# of discriminants	-315% (avg.)

2.2.1.5 Sentence Length

Sentence length has been considered as an informativeness measure with the assumption that longer sentences tend to be more complex, hence tending to include more information. One appealing feature of this metric is the ease of its computation.

Sentence length based sample selection is tested first in (Hwa, 2000) (for a summary of all results, see Table 2.4), with the same setting with which tree entropy was tested in the same paper. Sentence length provided 9% PRUD in this setting.

In (Baldridge and Osborne, 2003), sentence length is tested (again, experimental setup is the same as described above). 12% PRUD is observed in this setting. However, the same experimental setup led to -55.4% PRUD later in (Osborne and Baldridge, 2004). Authors conjecture that this difference stems from that number of sentences is used in the first work, and number of discriminants is used in the second work.

In (Hwa, 2004), sentence length is tested on WSJ corpus with PLTIG and Collins 2 parsers (experimental setup is the same as described above for the same paper). Experiments show that sentence length provides 26% reduction to reach 80% accuracy level with PLTIG parser and approximately 10% reduction to reach 88% F score level with Collins 2 parser.

In (Baldridge and Osborne, 2008), sentence length is tested (with the same experimental setup as described above for the same paper) to see -315% PRUD on average

(as claimed for (Osborne and Baldridge, 2004), the difference in results despite the usage of the same experimental setup is claimed to stem from the difference in used feature sets and machine learning algorithm).

2.2.1.6 Change Of Entropy

Change of entropy method was proposed in (Tang, Luo, and Roukos, 2002). In this approach, the informativeness of a sample is measured by how much it changes the entropy of the model. In decision tree models used in (Tang, Luo, and Roukos, 2002), this is calculated with the expected change in total weighted entropy at the leaves of decision trees, normalized with the number of possible parse actions. Authors claim that this is a measure of how much the new sample surprises the model.

Change of entropy metric is only tested in (Tang, Luo, and Roukos, 2002). Experimental setup and domain of this work is described above. On average, a PRUD of 67% is provided.

2.2.1.7 Word Co-occurance Statistics

Using word co-occurance statistics in active learning is first proposed in (Hwa, 2004). According to these method, sentences including previously unseen word co-occurances are prioritized in sample selection. Such samples are prioritized according to the function $f_{lex}(\mathbf{w}, G)$, which is:

$$f_{lex}(\mathbf{w}, G) = \frac{\sum_{w_i, w_j \in \mathbf{w}} new(w_i, w_j) \times coocc(w_i, w_j)}{length(\mathbf{w})} \tag{2.3}$$

Where $\mathbf{w}$ is the sentence, G is the current model, $new(w_i, w_j)$ becomes 1 if w_i and w_j have not been seen together before and 0 otherwise, $coocc(w_i, w_j)$ is the number of times w_i and w_j co-occurs in the candidate pool. In this scheme, sentences having highest f_{lex} values are selected first.

In experiments conducted on Collins 2 parser (see experimental setup specifics above),

it is seen that this method performs slightly worse than random sampling (an exact figure is not provided in the paper).

2.2.1.8 A Comparison of Single-Learner Active Learning Methods

We compiled the comparative performances of some single-learner active learning methods in Table 2.5. Note that we included results of only tree entropy, lowest best probability, unparsed, unparsed/entropy and two-stage active learning, because these will be the only informativeness measures that will be covered in Chapter 5, so it may provide ease to the reader in comparing reported and novel experimental results. Along with PRUD values of two different informativeness measures tested on the same domain, relative PRUD values of these measures (i.e. the PRUD that the informativeness measure in the "Inf. Meas. 1" column provides over the informativeness measure in the "Inf. Meas. 2" column) are presented for comparison purposes.

Table 2.5: Comparative performances of tree entropy, unparsed, unparsed/entropy, LBP and two-stage active learning

Inf. Meas. 1	**PRUD 1**	**Inf. Meas. 2**	**PRUD 2**	**RelPRUD**	**Reported In**	**Domain**
Unparsed	29.4%	Tree entropy	-5.7%	33.2%	Becker2005	Collins 2-WSJ
Unparsed/entropy	30.6%	Tree entropy	-5.7%	34.4%	Becker2005	Collins 2-WSJ
Two-stage	32.6%	Tree entropy	-5.7%	36.2%	Becker2005	Collins 2-WSJ
Tree entropy	58.4%	LBP	55.9%	5.7%	Osborne2004	Redwoods-ERG-HPSG
Tree entropy	50%	LBP	45%	9.1%	Hwa2004	PLTIG-WSJ
Tree entropy	27%	LBP	17%	12.0%	Hwa2004	Collins 2-WSJ
LBP	28%	Tree entropy	24%	5.3%	Baldridge2008	Redwoods-HPSG-ERG
Unparsed/entropy	30.6%	Unparsed	29.4%	1.7%	Becker2005	Collins 2-WSJ
Two-stage	32.6%	Unparsed/entropy	30.6%	2.9%	Becker2005	Collins 2-WSJ
Two-stage	32.6%	Unparsed	29.4%	4.5%	Becker2005	Collins 2-WSJ

2.2.1.9 Multi-Learner and Ensemble Active Learning

Active learning methods incorporating multiple models have a rich literature. Some of the most popular ones used in statistical parser training domain is as follows:

- **Preferred Parse Disagreement:** Sentences for which two different models get two different most likely parses are prioritized in sample selection.

- **Query By Committee:** First proposed by Argamon-Engelson and Dagan (1999), query by committee is used in statistical parser training first time by Osborne and Baldridge (2004). The idea is selecting samples on which a *committee* of learners disagree most. One measure of disagreement is *vote entropy*, which is defined as:

$$f^{ve}_{qbc}(s,\tau) = -\frac{1}{\log min(n,|\tau|)} \sum_{t\in\tau} \frac{V(t,s)}{n} \log \frac{V(t,s)}{n} \tag{2.4}$$

  Where n is the number of committee members, τ is the set of parses proposed by committee members and *V(t, s)* is the number of committee members proposing the parse t for sentence s. Another measure of disagreement is *Kullback-Leibler Divergence to the Mean* (abbreviated as *kl-div*), which is defined as:

$$f^{kl-div}_{\mu}(s,\tau) = \frac{1}{|\mu|} \sum_{M\in\mu} D(P_M, P_{avg}) \tag{2.5}$$

  Where "μ denotes the set of ensemble models, P_{avg} is the mean distribution over ensemble members in μ, $P_{avg} = \sum_M P_M(t|s)/|\mu|$, and $D(.,.)$ is KL-divergence" (Becker and Osborne, 2005).

- An alternative to multi-learner setting is constructing an *ensemble* from multiple models, which behaves like a single-learner during the active learning process. There are different ways of creating and ensemble like directly combining feature sets of multiple models into a single *monolithic* feature set (Baldridge and Osborne, 2008), *product-of-experts* formulation (Osborne and Baldridge, 2004) and *logarithmic opinion pool* formulation (Baldridge and Osborne, 2008).

We will not review the literature on multiple model active learning thoroughly, since our experiments do not cover multiple model methods (see Chapter 5)

2.2.2 Proposed Representativeness Measures

2.2.2.1 Length Balanced Sampling (LBS)

Length Balanced Sampling (LBS) was proposed in (Becker and Osborne, 2005). LBS can be used in any selective sampling setting and works in the following way:

- Before beginning active learning, b samples are selected randomly from the pool of unlabeled sentences, where b is the batch size to be used during active learning.
- Sentence length statistics of these b samples are recorded in a histogram H, where:

$$H = (e_1, e_2, \ldots, e_{|H|}) \tag{2.6}$$

  Here each e_i denotes the number of occurences of i words long sentences in the samples. Histogram size (i.e. number of entries in the histogram) may be set arbitrarily (as long as it is not less than the maximum sentence length appearing in the random sample), but sums of values recorded in the histogram must sum up to b
- At any step of active learning, the current sample pool is partitioned into $|H|$ equivalence classes according to the sentence length and from each equivalence class including i words long sentences, the most informative e_i sentences are selected for batch (whatever the used informativeness measure is).

It is conjectured that since the samples for the histogram are drawn randomly from the pool to be used in active learning, this random sample reflects the statistical characteristics of the pool with respect to sentence length, hence any batch constructed according to the histogram will represent the statistical characteristic of the pool with respect to sentence length.

LBS is tested in (Becker and Osborne, 2005) with tree entropy, two-stage active learning, unparsed and unparsed/entropy informativeness measures. Histogram size is set as 40 in this work. However, LBS is not compared with any other representativeness measure or a random baseline (i.e. the scheme of selecting only the most informative samples everytime) in (Becker and Osborne, 2005). Experimental results of (Becker and Osborne, 2005) can be found in the "Proposed Informativeness Measures" section.

2.2.2.2 Sample Density

Sample density is a density estimation based representativeness measure, first proposed in (Tang, Luo, and Roukos, 2002). Given a set of sentences $\mathbf{S} = \{S_1, \ldots, S_N\}$, density of a sample sentence S_i is defined as:

$$\rho(S_i) = \frac{N-1}{\sum_{j \neq i} d_M(S_i, S_j)} \tag{2.7}$$

Where d_M is the *distance metric* or *similarity measure* defined according to the model M. $\rho(S_i)$ is the multiplicative inverse of the average distance to S_i, and effectively corresponds to the notion of *density* of a sample. The d_M used in (Tang, Luo, and Roukos, 2002) was the edit distance between the most likely parses of two sentences. Since the edit distance function they defined was model-specific, d_M is not generic. Any successor study that would like to employ sample density should implement its own d_M. Besides, since d_M depends on M, it should be recalculated at each iteration of active learning. Note that the success of this method critically depends on the modeling strength of M. If the model is a weak one or it has not sufficient data to estimate the underlying sample distribution, this method will give false predictions of the densities of samples.

This representative method is tested in conjunction with tree entropy (normalized with sentence length) in (Tang, Luo, and Roukos, 2002), with the same experimental setup and domain described above. Authors measured PER instead of DUR, and they report 22.3% PER compared to using only normalized tree entropy (and no representativeness measures) after 1000 sentences.

2.2.3 Proposed Diversity Measures

2.2.3.1 K-Means Clustering

First tried in statistical parser training domain by Tang, Luo, and Roukos (2002), k-means clustering is a generic diversity incorporation method used in active learning. The main idea is that after clustering, similar samples will tend to gather in the same cluster; so by selecting samples from different clusters, the learner can get diverse samples for the batch. Of course, the success of such an approach vitally depends on the distance metric used in clustering process.

In (Tang, Luo, and Roukos, 2002), d_M is used as the distance metric (see "Sample Density" topic in Section 2.2.2) and number of clusters is equal to the batch size. At each iteration of active learning, clusters are recalculated (since d_M is updated at each iteration) and one sample is selected from each cluster. Experiments show that change of entropy metric is not affected from whether k-means clustering is employed or not. Non-normalized tree entropy does a little bit worse; however, tree entropy normalized with sentence length shows a clear PER of 20% after 1000 sentences when clustering is employed.

CHAPTER 3

PROPOSED REPRESENTATIVENESS MEASURE

Principle of Least Effort, as formulated by Zipf (1949), roughly says that people try to express themselves in the most economic way possible. In (Zipf, 1935), Zipf says "In view of the evidence of the stream of speech we may say that the length of a word tends to bear an inverse relationship to its relative frequency". In a footnote, he adds: "Not necessarily proportionate; possibly some non-linear mathematical function". There is no reason for this logic not to be valid for sentences as well, and indeed, empirical observation on Brown corpus (Kučera and Francis, 1967) shows that this claim is true *down to a certain length*. In the corpus, empirical frequencies of the words begin to fall after three letters as predicted by Zipf; however, frequencies of one-letter and two-letter words are less than the frequency of three-letter words. In a similar way, empirical frequencies of the sentences begin to fall after 16 words while they rise until 16 words.

Sigurd, Eeg-Olofsson, and van Weijer (2004) explained this predictive failure of Principle of Least Effort for short words by conjecturing that two conflicting tendencies govern the frequencies of words according to their lengths: As more letters are available, more different words may be constructed by these letters; hence frequencies of longer words tend to rise. On the other hand, long words are uneconomic, hence frequencies of longer words tend to fall. Authors model rising tendency originating from expressivity as a polynomial function of word length, and falling tendency as an exponential decay function of word length. So they give the approximate frequency function

as:

$$f(L) = a \times L^b \times c^L \tag{3.1}$$

Where $f(L)$ is the relative frequency of words of L letters, a, b and c are real constants where $0 < c < 1$. This formula actually gives a Gamma distribution and successfully models the empirical phenomena: Since for little L values, polynomial term dominates in determining the rate of change (in an increasing way); however, as L gets larger exponential decay term eventually prevails in determining the rate of change (in a decreasing way). When they fit the parameters a, b and c, they saw that model and data gave the correlation coefficient 0.978 for English. In addition, data shows that sentence length seems to obey the same formula. Authors argue that it's not surprising, since the same theoretical reasoning is applicable for sentences as well (when letters are substituted with words). When they fit the parameters for sentence length, the model fits the data with the correlation coefficient 0.992 for English (you can see the quality of the fit in Figure 3.1, which is taken from (Sigurd, Eeg-Olofsson, and van Weijer, 2004)). The fitted model (which we will call $f_{zipf-eng}$ from now on) is reported as:

$$f_{zipf-eng}(s) = 1.1 \times L^1 \times 0.90^L \tag{3.2}$$

Gamma distribution based on length seems to exhibit characteristics of many linguistic structures. Empirical fits for syllables in German and phonemes in Swedish are reported in (Sigurd, Eeg-Olofsson, and van Weijer, 2004), while morpheme lengths are reported to fit good in Finnish (Creutz, 2003).

Using $f_{zipf-eng}$ as a representativeness measure in active learning is especially appealing because of the following reasons:

1. $f_{zipf-eng}$ is model-independent, so its decisions are not affected from the deficiencies of the current model, unlike density estimation based representativeness measures like sample density. This property brings another advantage: First, it can be used in active learning of other NLP tasks such as POS-tagging, supertagging etc. and second, it can be used on synthetic samples (hence it can be used in settings other than selective sampling).

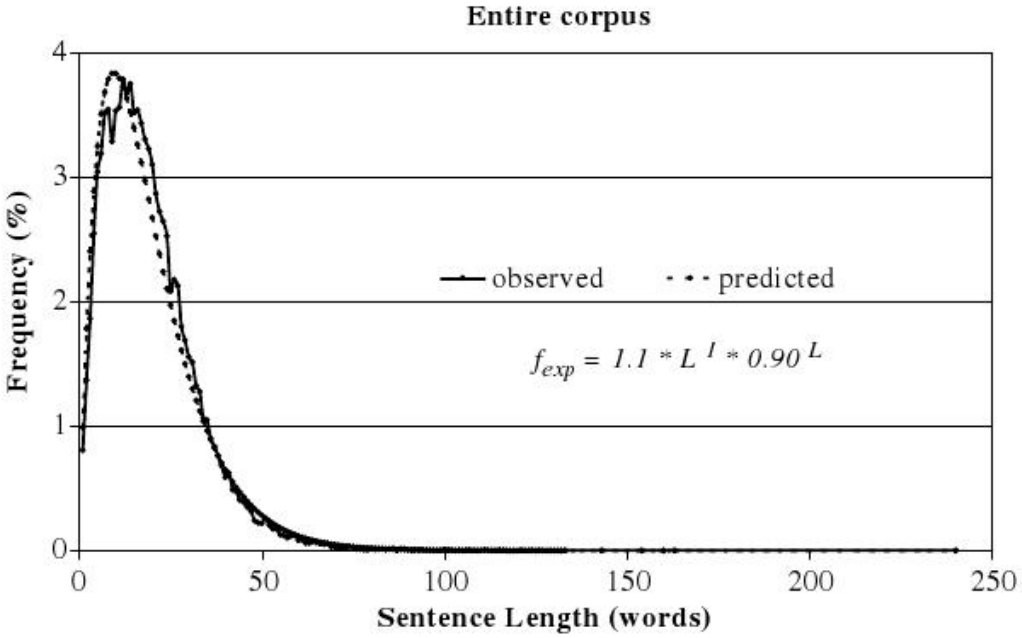

Figure 3.1: Observed and predicted sentence frequencies on the basis of sentence length in words (L) in the entire Brown corpus

2. $f_{zipf-eng}$ is based on a sound theory which is empirically validated. Predictions on a large corpus have been very successful (again, see Figure 3.1).

3. Since it is a numerical value, it can be combined with quantitative informativeness measures to give quantitative sample selection metrics (a few such combination functions are introduced in Chapter 5). The ability of defining alternative combination functions provides a flexible framework in active learning, bringing the opportunity to determine and understand the contributions of informativeness and representativeness terms mathematically.

Of course, since $f_{zipf-eng}$ is determined by only sentence length, it can be criticized for not being a *fine-grained* representativeness measure, since it does not consider any syntactic or semantic feature of the sentence. As a simple example, it will assign the same probability value to two sentences having the same length; but one of them may include linguistically more rare events. Although we accept that this is a drawback of our method, to the best of our knowledge there is not a successful fine-grained frequency estimator for sentences proposed up to now.

CHAPTER 4

TEST DOMAIN

In this chapter we describe our experimental domain. Since the $f_{zipf-eng}$ formula is valid for English and–to the best of our knowledge–there is not empirical fitting studies for languages other than English for now, we will test Zipfian sampling on an English corpus and parser. For purposes described later in Section 4.3, we chose Clark and Curran's combinatory categorial grammar parser (abbreviated as "C&C Parser" (Clark and Curran, 2004b; Clark and Curran, 2007)) as a wide coverage parser to be used in tests. Since we work with a CCG parser, we naturally use CCG Bank (Hockenmaier and Steedman, 2005) as the annotated English corpus.

4.1 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) (Ades and Steedman, 1982; Steedman, 2000), which is an extension of classical categorial grammars (Bar-Hillel, 1953), is a radically-lexicalized type-driven theory of grammar. Unlike top-down generative models (like context-free grammars) in which derivations are controlled mainly by rules and there are relatively many rules for that, categorial grammars are characterized by that there are few rules and derivations are controlled mainly by types of lexical entries. Types are in the form of *functors* or *arguments*; but functor types include type and directionality of their arguments as well. At the beginning of a derivation, each word is assigned a type (or *category*, as this is the common name used in categorial grammar literature), either a functor or an argument, and these categories *combine* according to some *combinatory rules* to give new categories, wither functor or argument. Combinatory rules,

along with directionality and categorial constraints in the functor categories, enforce the word order constraints.

In CCG, argument types are considered as *basic categories*. Examples of basic categories are S (for sentences), N (for nouns), NP (for noun phrases) and PP (for prepositional phrases). In realistic grammars, basic categories are rich types including syntactic features like gender, number, case, tense etc. Functor categories are considered as *complex categories*. For example, in English transitive verbs are *assigned* the category:

(1) bought := $(S\backslash NP)/NP$

Which means that the transitive verb (here, *bought*) is assigned a functor (complex) category, which takes an NP argument to the right, takes an NP argument to the left (exactly in this order) and results in an S category. Combinatory rules used in combining functor and arguments (or transforming a category to another, which are called *unary rules*). Two most basic combinatory rules used in classical categorial grammars are *function application* rules, which are:

(2) $(X/Y) \quad Y \quad \Rightarrow \quad X \quad (>)$

(3) $Y \quad (X\backslash Y) \quad \Rightarrow \quad X \quad (<)$

Where X and Y denotes any kind of categories. Forward application rule ($>$) says that at a stage of parsing, when a sequence of words (or tokens, in more general sense) having the category Y follows immediately to the right of another sequence of words having the category (X/Y), these two word sequences combine to a single sequence having the category X. Backward application rule ($<$) is similar, but this time the word sequence having the argument category must follow immediately to the left of the sequence having the functor category, since directionality of the slash is reversed this time.

Since classical categorial grammars use only the application rules, their expressive power is rather limited (actually their class is shown to be weakly equivalent to the class of context-free grammars (Bar-Hillel, Gaifman, and Shamir, 1960)). For this reason, Steedman introduced (2000) additional rules stemming from combinatory logic.

(4) $(X/Y) \quad (Y/Z) \quad \Rightarrow \quad (X/Z) \quad (>\mathbf{B})$

(5) $(Y\backslash Z) \quad (X\backslash Y) \quad \Rightarrow \quad (X\backslash Z) \quad (<\mathbf{B})$

(6) $(X/Y) \quad (Y\backslash Z) \quad \Rightarrow \quad (X\backslash Z) \quad (>\mathbf{B}_\times)$

(7) $(Y/Z) \quad (X\backslash Y) \quad \Rightarrow \quad (X/Z) \quad (<\mathbf{B}_\times)$

(8) $((X/Y)/Z) \quad (Y/Z) \quad \Rightarrow \quad (X/Z) \quad (>\mathbf{S})$

(9) $(Y\backslash Z) \quad ((X\backslash Y)\backslash Z) \quad \Rightarrow \quad (X\backslash Z) \quad (<\mathbf{S})$

(10) $((X/Y)\backslash \mathrm{Z}) \quad (Y\backslash Z) \quad \Rightarrow \quad (X\backslash Z) \quad (>\mathbf{S}_\times)$

(11) $(Y/Z) \quad ((X\backslash Y)/Z) \quad \Rightarrow \quad (X/Z) \quad (<\mathbf{S}_\times)$

(12) $X \quad \Rightarrow \quad T/(T\backslash X) \quad (>\mathbf{T})$

(13) $X \quad \Rightarrow \quad T\backslash(T/X) \quad (<\mathbf{T})$

In literature, rules ($>\mathbf{B}$) and ($<\mathbf{B}$) are called as forward and backward *composition rules* respectively, rules ($>\mathbf{B}_\times$) and ($<\mathbf{B}_\times$) are called as forward and backward *crossed composition* rules respectively, rules ($>\mathbf{S}$) and ($<\mathbf{S}$) are called as forward and backward *substitution* rules respectively, rules ($>\mathbf{S}_\times$) and ($<\mathbf{S}_\times$) are called as forward and backward *crossed substitution* rules respectively and rules ($>\mathbf{T}$) and ($<\mathbf{T}$) are called as forward and backward *type-raising* rules respectively. A sample derivation using some of these rules can be seen in Figure 4.1. In time new rules have been derived (Hoyt and Baldridge, 2008), or existing ones have been modified (Baldridge, 2002), as long as the proposed rules comply a set of *principles* (Steedman, 2000). These principles limit the expressive power of the formalism; however it is still more expressive than context-free grammars. (Vijay-Shanker and Weir, 1994) proved that combinatory categorial grammar is mildly context-sensitive (Joshi, 1985).

As noted in (Clark and Curran, 2007), CCG has several features making itself appealing to be used in wide-coverage statistical natural language parsing:

- Being mildly context-sensitive, it enables the capturing of limited kinds of, but unboundedly many cross-serial dependencies, while still being parsable in polynomial time.

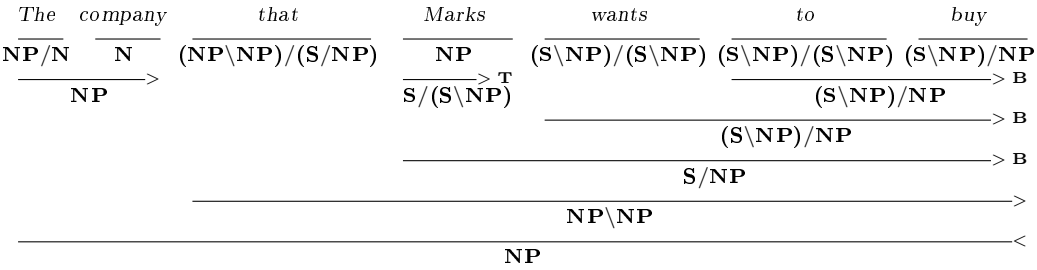

Figure 4.1: A sample CCG derivation

- It provides a very natural interpretation of many linguistic phenomena common in general natural language texts, such as long-range dependencies, non-constituent coordination, gapping, scrambling etc. These phenomena are predicted by CCG derivations naturally, i.e. without using any extra assumptions, movements and transformations etc.

- Since *Principle of Categorial Type Transparency* (one of the principles setting constraints on possible lexical categories in a grammar instance) dictates that the syntactic category of a lexical item must be a reflection of the semantic type of that lexical item, syntax-semantics interface is completely transparent and syntactic derivation itself is the process of recovery of the underlying predicate-argument structure in one sense.

4.2 CCGbank

CCGbank (Hockenmaier and Steedman, 2005) is the collection CCG analyses of WSJ corpus sentences, which are semi-automatically derived from Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993). Penn Treebank phrase-structure trees are translated to CCG analyses with an algorithm (Hockenmaier, 2003), with some manual corrections.

Sentence categories (S) in CCGbank carry features indicating the sentence/clause type (declarative, interrogative, 'for' clause etc.) or verb phrase properties (bare infinitive, 'to' infinitive, past participle in passive mode etc.).

In addition to standard CCG rules for English, CCGbank introduces a number of

unary rules (or *type changing rules*). A complete list of unary rules is available in (Hockenmaier, 2003). In general, these rules change a verb phrase into a modifier to handle situations like relative pronoun drop etc. Here are some examples of these rules (examples are taken from (Clark and Curran, 2007), type-changing rule is applied to the bracketed expression in each example):

- $S_{[pss]}\backslash NP \Rightarrow NP\backslash NP$ (*pss* corresponds to past participle in passive mode)
  workers [exposed to it]
- $S_{[adj]}\backslash NP \Rightarrow NP\backslash NP$ (*adj* corresponds to adjectival phrase)
  a forum [likely to bring attention to the problem]
- $S_{[ng]}\backslash NP \Rightarrow NP\backslash NP$ (*ng* corresponds to -ing form)
  signboards [advertising imported cigarettes]
- $S_{[ng]}\backslash NP \Rightarrow (S\backslash NP)\backslash(S\backslash NP)$
  became chairman [succeeding Ian Butler]
- $S_{[dcl]}\backslash NP \Rightarrow NP\backslash NP$ (*dcl* corresponds to declarative)
  the millions of dollars [it generates]
- $N \Rightarrow NP$ *transfer [money]*

Some binary rules which are not in CCG's standard rule set are employed to handle punctuation, for example:

- , $S_{[dcl]} \Rightarrow S_{[dcl]}$
- , $X \Rightarrow X\backslash X$

Associativity of combination operators causes *spurious ambiguity* in CCG, because of which the same result can be obtained by different analyses. An example of spurious ambiguity is as follows:

(14) a.

Mary	*saw*	*John*
NP	**(S\NP)/NP**	**NP**
	S\NP (>)	
S (<)		

b.
$$\dfrac{\dfrac{\dfrac{\textit{Mary}}{\mathbf{NP}} \quad \dfrac{\textit{saw}}{\mathbf{(S\backslash NP)/NP}} \quad \dfrac{\textit{John}}{\mathbf{NP}}}{\mathbf{S/(S\backslash NP)}}\,{>}\mathbf{T}}{\mathbf{S/NP}}\,{>}\mathbf{B}}{\mathbf{S}}\,{>}$$

Due to spurious ambiguity, a sentence can have more than one correct analyses. In order to prevent confusion, all analyses in the CCGbank are in a kind of *normal form*, such that a CCGbank derivation includes type-raising and composition only when it is necessary.

Besides the CCG derivations, CCGbank presents the list of predicate-argument dependencies for each sentence. This is especially important for parser evaluation and training parsers relying on dependency modeling.

Finally, we should note that CCGbank is not one-to-one translation of Penn Treebank. Coverage of the conversion algorithm is reported as 99.44% and many corrections are made on the original Penn Treebank data before conversion (mostly POS-tag assignments).

4.3 Clark and Curran Parser

C&C Parser (Clark and Curran, 2004b; Clark and Curran, 2007) was a project which began as a part of the Edinburgh wide-coverage CCG parsing project (2000-2004) and developed later independently by Stephen Clark and James Curran. The motivation of the project was developing a wide-coverage parser having both high coverage of linguistic constructions that could be found in CCGbank and high efficiency for large-scale NLP tasks. In almost all design decisions for the parser, this trade-off is considered.

4.3.1 The CCG Implementation

C&C Parser does not use multi-modal extension of CCG (Baldridge, 2002), hence it follows Steedman's classic analysis of English (Steedman, 2000). In this analysis application ($<$ and $>$), composition ($<\mathbf{B}$ and $>\mathbf{B}$), backward cross composition ($<\mathbf{B}_{\times}$) and type raising ($<\mathbf{T}$ and $>\mathbf{T}$) are allowed. Forward cross composition ($>\mathbf{B}_{\times}$) is disal-

lowed complying to Steedman's analysis and all substitution (**S**) rules are ignored since their contribution to the coverage of the parser did not worth the reduction in speed. Besides these rules, all the rules that are manually added to CCGbank derivations (see Section 4.2) are included in the parser as well.

C&C parser set restrictions on categories to which CCG rules can be applied. For example, again following Steedman, there is a restriction on backward cross composition rule ($<\mathbf{B}_{\times}$) that Y in (7) cannot be N or NP. Type raising is applied only to the categories NP, PP and $S_{[adj]}\backslash NP$.

Optionally, in order to increase efficiency of the parser (hence success of the training), the parser may employ two types of constraints for rule applications: One is allowing two categories to combine only if they are seen to be combined in the training data, and second is Eisner's normal-form constraint (Eisner, 1996). Eisner's normal-form constraint dictates that a category produced by a forward composition rule instance cannot serve as the primary (left) functor in a forward composition or application rule instance and similarly, a category produced by a backward composition rule instance cannot serve as the primary (right) functor in a backward composition or application rule instance. This constraints cannot ensure normal-form derivation in a grammar using type-raising, nonetheless, they are reported to prune the parse space dramatically (Clark and Curran, 2007).

4.3.2 Modeling Predicate-Argument Dependencies and Head-Lexicalized Derivations

As we noted, lexical items in a realistic CCG include features. In C&C parser, in addition to the categorial features introduced in CCGbank, lexical items include head-word and predicate-argument dependency information encoded in their syntactic categories. An example category is:

(15) bought := $(S[dcl]_{bought}\backslash NP_1)/NP_2$

Where dependency *slots* are enumerated from left to right. Clark and Curran use a kind of unification based grammar, hence two categories can combine only if they are *unifiable*. Using co-indexing with unification, a kind of *head passing* mechanism can be

employed. Head passing is useful for determining multiple heads and extraction of non-local dependencies. For example, consider the following category for the object-control verb *persuade*:

(16) persuade := $((S[dcl]_{persuade}\backslash NP_1)/(S[to]_2\backslash NP_X))/NP_{X,3}$

Via variable X, unification *passes* the head of the object to the subject of the infinitival, so the head of the infinitival complement's subject can be determined by the head of the object. In a similar way, long-range object-extraction from a relative clause can be identified using the following category for a relative pronoun:

(17) who := $((NP_X\backslash NP_{X,1})/(S[dcl]_2\backslash NP_X))$

The head-driven and unification-based version of the CCG derivation given in Figure 4.1 can be found in Figure 4.2 (taken from (Clark, Hockenmaier, and Steedman, 2001), features on sentence categories are discarded to save space and head-words are reduced to the first letter). Whenever an argument slot is filled during the derivation, a new *dependency* is created by the parser. Formally, a predicate-argument dependency is a five-tuple $\langle h_f, f, s, h_a, l\rangle$, where h_f is the head word of the functor, f is the functor category, s is the number of the filled argument slot, h_a is the head-word of the argument and l encodes whether the dependency is local or non-local. For example, consider the following derivation:

(18)
$$\dfrac{\dfrac{\overline{\textit{Mary}}}{\mathbf{NP}_{\text{Mary}}} \quad \dfrac{\dfrac{\overline{\textit{saw}}}{(\mathbf{S}[dcl]_{\text{saw}}\backslash\mathbf{NP}_1)/\mathbf{NP}_2} \quad \dfrac{\overline{\textit{John}}}{\mathbf{NP}_{\text{John}}}}{\mathbf{S}[dcl]_{\text{saw}}\backslash\mathbf{NP}_1}>}{\mathbf{S}[dcl]_{\text{saw}}}<$$

In the first step of derivation in (18), the following dependency is created by the parser:

(19) $\langle saw_2, (S[dcl]_{saw}\backslash NP_1)/NP_2, 2, John_3, -\rangle$

Subscripts on the lexical items indicate sentence position and the final field ($-$) indicates that the dependency is a local one. The set of dependencies created during a derivation constitute the predicate-argument structure (PAS) of the sentence and it is one of the outputs of the C&C Parser.

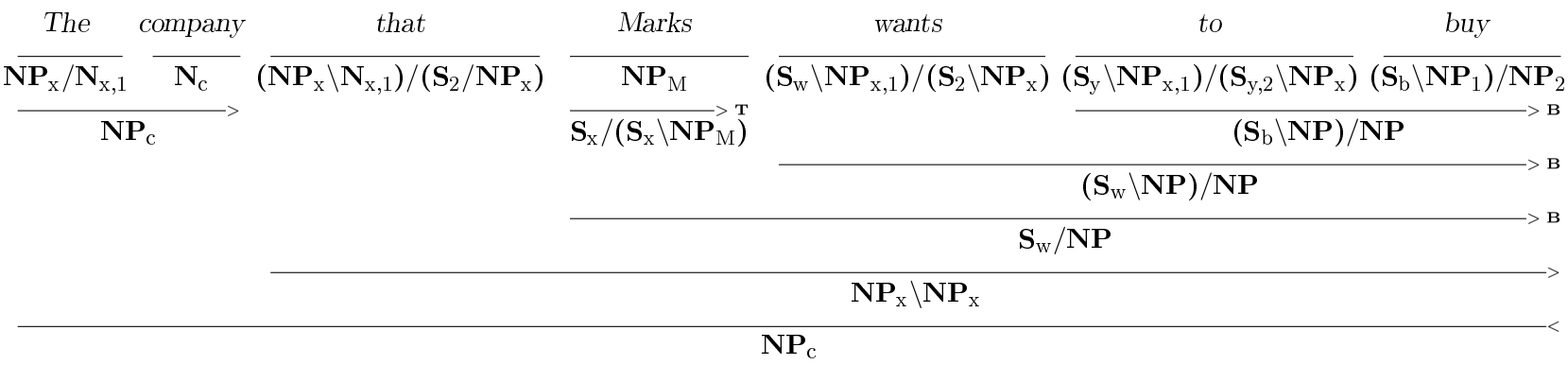

Figure 4.2: A sample head-driven and unification-based CCG derivation

4.3.3 Statistical Model, Parsing and Training

The underlying CCG parser used is a variant of CKY algorithm described in (Steedman, 2000). Statistical model and training method are an instance of Stochastic Unification Based Grammar (SUBG) framework proposed in (Johnson et al., 1999). The parser assigns probabilities to each parse that it finds according to the following conditional log-linear model:

$$P(\omega|S) = \frac{e^{\lambda \cdot \mathbf{f}(\omega)}}{Z(S)} \tag{4.1}$$

Where ω is a parse, $\mathbf{f}(\omega)$ is the vector including counts of features associated with the parse, λ is the vector including weights of features and $Z(S)$ is a normalizing constant ensuring that $P(\omega|S)$ is a probability distribution:

$$Z(S) = \sum_{\omega' \in \rho(S)} e^{\lambda \cdot \mathbf{f}(\omega')} \tag{4.2}$$

Where $\rho(S)$ is the set of possible parses of S. The notion of 'parse' depends on the model used by the parser: In *derivation* model, a parse is a (derivation,PAS) pair where the specified derivation leads to the specified PAS. In *normal-form* model, a parse is simply a head-lexicalized derivation. The features that are allowed to appear in a parse vary with the model as well. Since we do use only normal-form model in our experiments (because of the reasons we will explain in Section 4.3.5), we will not review any details associated with dependency model from now on. The features associated with the normal-form model are as follows:

- Lexical category assignments to words and POS-tags.
- Final resulting category in bare form, with its head word and with POS-tag of the head.
- A rule instantiation in its bare form, with its dependent word, with POS-tag of the dependant, with both head and dependant words/POS tags.
- Distance of occuring dependencies with their rule instantiation, dependant words and POS-tags of the dependant.

Included instances of the above are those observed in the gold-standard derivations in the CCGbank at least with a pre-determined frequency. This lower bound on the

frequency of a feature is called *frequency cutoff* for that feature.

Estimating the log-linear normal-form model parameters is done by maximizing the objective function $L'(\Lambda)$, which is:

$$L'(\Lambda) = L(\Lambda) - G(\Lambda) \tag{4.3}$$

$$= log \prod_{j=1}^{m} P_\Lambda(d_j|S_j) - \sum_{i=1}^{n} \frac{\lambda_i^2}{2\sigma_i^2} \tag{4.4}$$

$$\tag{4.5}$$

Where $L(\Lambda)$ is the log-likelihood of the model Λ, $G(\Lambda)$ is the regularization term (alternatively called Gaussian prior term) used to prevent overfitting, n is the number of features, m is the number of sentences in the training data, d_j is the gold standard derivation for the sentence S_j, λ_i is the weight of the ith feature and σ_i is the smoothing parameter (also called Gaussian prior) for the ith feature. In practice, for all i $\sigma_i = \sigma$. Optimization is done by moving in the feature space with a *steepest-ascent* strategy and `L-BFGS` algorithm (Nocedal and Wright, 1999) is used for this purpose. `L-BFGS` is an efficient search algorithm since it calculates only first partial derivatives of the objective function with respect to feature weights and estimates second partial derivatives according to the changes in the first partial derivatives during a fixed number of previous steps, hence evading calculating the Hessian, which is computationally very expensive. Calculating first partial derivatives requires calculating empirical feature counts (i.e. the number of times a feature appear in the training data) and expected empirical feature counts (i.e. the *expected* number of times the feature may appear according to the current model Λ). Calculating expected empirical feature counts require parsing all sentences in the training data, and reparsing the whole training set in each iteration can be computationally expensive. To overcome this problem, charts of all training sentences are kept in memory. Since it may require a huge amount of RAM in practice, Beowulf clusters are employed in order to supply the necessary memory.

Parsing is done using *packed charts*. Packed charts are common in parsing, in which equivalent *individual entries* are collected into *equivalence classes*. Two individual entries are said to be equivalent when they lead to the generation of the same subsequent parse structure (i.e. they behave exactly same in subsequent parse actions). In C&C

parser, two individual entries are equivalent if they have the same span, same category, identical head and the same list of unfilled dependencies. In this way, equivalent entries lead to the same derivation and PAS. Individual entries are either leaves of CCG derivations (lexical item/category pairs) or obtained by combining the canonical representatives of two equivalence classes via a binary rule or applying a unary rule to the canonical representative of an equivalence class. For conceptual ease, Clark and Curran use the abstraction of *feature forests*, which is a graph of *conjunctive* and *disjunctive* nodes. The notion of conjunctive node corresponds to individual entries and the notion of disjunctive node corresponds to equivalence classes. Equivalence classes at the root of CCG derivation trees are called *root disjunctive nodes*.

Packed charts and binary nature of CCG derivations make CKY algorithm applicable for parsing. Via CKY, inside and outside scores can be calculated (and tree entropy as well, see Appendix A). In normal-form model it also enables employing Viterbi decoder to find the most probable parse. Actually since spurious ambiguity cannot be eliminated completely (even with Eisner's constraints), it is not guaranteed that Viterbi decoder will return the most probable parse since non-normal forms of a parse can arise in the chart and 'steal' some of the probability belonging to the normal form of the parse[1]. However, since the model is trained *discriminatively* and the probability of the gold-derivation (which is always in normal form) of each sentence is maximized against the alternative parses, authors conjecture that the parser will typically assign low probabilities to non-normal form derivations in general.

4.3.4 POS-Tagger and Supertagger

For generalization purposes, a maximum entropy POS-tagger described in (Curran and Clark, 2003) is used for POS-tagging. One of the most important components of the parser is the *supertagger* (Clark and Curran, 2004a). Supertagger works identical to the POS-tagger described in (Ratnaparkhi, 1996), but it assigns CCG categories to lexical items instead of POS tags. The probability of the assignment of a category to a lexical item is determined according to a log-linear model, features of which are

[1]Same problem affects the calculation of tree entropy in such a way that monolithic probabilities belonging to an equivalence class of parses can be scattered among normal and non-normal form derivations of the class, boosting the calculated value of tree entropy above its actual value.

the words and POS tags in the local 5-word window and the two previously assigned categories immediately to the left.

Since supertagger is too restrictive, authors implemented a *multitagger*, which is able to assign multiple CCG categories to a word. Flexibility of the multitagger in assigning categories is determined by an ambiguity parameter β: Only categories whose probabilities are within β of the highest probability category are assigned to the lexical item. Besides β, the supertagger has an additional parameter limiting the categories that can be assigned to a lexical item: The dictionary cutoff level, which is set to some integer k. For words seen at least k times in the training data, the supertagger can only assign the categories seen with the word in the training data. For words seen less than k times in the training data, the POS tag of the word is used in such a way that the supertagger can only assign the categories seen with the POS tag of the word in the training data. β and dictionary cutoff level effectively determines the number of categories that can be assigned to a lexical item.

The motivation for using supertagger is the fact that using all possible lexical item/category assignments as leaves of CCG derivations is infeasible due to memory (hence time) constraints. Supertagger functions as a filter on categories that can be assigned to words and 'permeability' of the filter can be changed via playing with β and dictionary cutoff level. In practice, dictionary cutoff level is not changed frequently, it is the β determining the restrictivity: Higher β values are more restrictive and lead to assignment of less categories. Supertagger operates adaptively with the parser in the following way: In the first parsing attempt, β value is set to a relatively high value (hence it's most restrictive). If the parser cannot find any derivation leading to a root, β value is reduced and attempt is repeated, and so on (of course, number of attempts are limited). In this way, the parser operates at a very high speed without sacrificing accuracy. The supertagger is vital in the training process as well, since memory requirements for training are infeasible without a supertagger even using a cluster. In the training process, a reverse strategy is followed in tuning ambiguity: The parser begins with low β values and β is increased if the chart gets too large. Low β values are more desirable in training process, since high β values lead to a less number of parses and training algorithm cannot find enough incorrect derivations to discriminate against. Consequently,

accurracy deteriorates with training done with high β values.

The whole process can be summarized as follows: POS tagger takes raw input words as input and outputs (POS tag,word) pairs. Supertagger takes the output of the POS tagger as input and outputs several (category,POS tag,word) triples for each (POS tag,word) pair. Finally, parser takes the output of the supertagger and outputs a (derivation,PAS) pair.

4.3.5 Why Do We Use C&C Parser?

C&C Parser is a wide-coverage parser with high performance on WSJ corpus. We wanted a realistic setting for our experiments, and high performance of the passive learner is paramount for the performance of active learner counterpart (see (Baldridge and Osborne, 2004)). The parser has an open source implementation, which provides ease for us. Besides these properties which can be found in other parsers (like Collins parser (Collins, 1997)), one important advantage of C&C Parser distinguishing it from the alternatives is its speed: The parser is able to parse approximately 30 sentences per second on average on the hardware we use. In this way active learning experiments could be completed very fast.

Among the two models C&C Parser implemented, we chose normal-form model over dependency model for similar reasons: Training of the normal-form model is faster and uses less memory. Besides, Viterbi decoding cannot be employed to find the most probable PAS in the dependency model (instead authors use a method finding the PAS with maximum expected recall), so we cannot employ some active learning algorithms like LBP with dependency model.

CHAPTER 5

EXPERIMENTS

5.1 Scope of Experiments

Since we introduce a novel representativeness measure, conducted experiments are expected to test the performances of the novel method and previously proposed representativeness measures comparatively, possibly with various informativeness and diversity measures. We do this; however, because of the following reasons, we could not cover all measures presented in Section 2.2:

- We did not cover any multi-learner or ensemble style active learning methods, although they are shown to perform much better than single-learner methods *if used models are diverse enough*, since introducing diverse models in C&C Parser domain is not trivial. In (Becker and Osborne, 2005), multiple models needed for KL-div/unparsed algorithm was created by multiple bagging; however, we believe that it does not create sufficiently diverse models (a view which is backed by the observation that KL-div/unparsed did not do better than the single-learner method two-stage active learning). Using the two different models implemented in C&C parser (maybe even the third one, the "hybrid dependency model") could be a solution, but even these two models have a significantly large base common set of features. Although we believe that multi-model methods are too important to ignore, we believe that specifying multiple methods in C&C parser domain is a broad study and it should be handled in a separate work. Hence we leave multi-model methods out of the scope of experiments.

- Another untrivial task is specifying similarity measures like those described in Section 2.2.2 and 2.2.3 in C&C Parser domain. As pointed out in (Becker, 2008), the best approach to density estimation is not clear. To make the approach of applying edit distance variation on parses (as proposed in (Tang, Luo, and Roukos, 2002)) work empirically, determining useful edit distance parameters that can be derived from C&C parser outputs is necessary; hence we again think that this should be the topic of a separate study and density estimation methods are ignored in our experiments.

- Some informativeness measures either are too model-specific to be applied (like *change of entropy* proposed in (Tang, Luo, and Roukos, 2002)) or introduce insignificant or negative improvement reportedly (like sentence length and word co-occurence statistics proposed in (Hwa, 2004)), so we do not cover them in our experiments.

Although these constraints prune many methods, a vast majority of the most classical methods are still available for testing. Tested informativeness measures are tree entropy, unparsed/entropy, two-stage active learning and lowest best probability.

Since all diversity measures proposed up to now are based on density estimation methods, none of the known diversity measures could be incorporated in our experiments. The same constraint also rules out sample density based representativeness measures introduced in Section 2.2.2, leaving only length balanced sampling as a representativeness measure, which is itself based on sentence length as well, for comparison. It can be unsatisfactory to compare our novel method with only one alternative; however, it should be noted that the literature on representativeness already is quite poor.

In addition, we make random sampling experiments in order to measure PRUD values of active learning methods, as all studies in the field do.

5.2 Experimental Setup

5.2.1 Parser Settings

We used default settings of the C&C parser set in the distribution for fair evaluation. Maximum number of allowed chart entries is limited to 300,000. Maximum number of words allowed in a sentence to be parsed is 250. Supertagger is employed with five ambiguity levels in which β levels are 0.075, 0.03, 0.01, 0.005, 0.001, respectively and dictionary cutoff levels 20, 20, 20, 20, 150, respectively. Gaussian prior σ used in regularization of likelihood function is set to 1.3. Only rules that are seen in the training data are applied and Eisner normal-form constraints are employed during the parsing process. Question rules are not employed during the experiments. These settings are claimed to improve accuracy without degrading parsing speed significantly whenever possible, or to be a reasonable compromise between speed and accuracy otherwise (see C&C Tools website). Used POS-tagger settings are default as well, details of which can be found in C&C Tools website. As we noted in Section 4.3.3, we use normal-form model of the C&C parser in our experiments.

Training parameters are default as well. Supertagger is trained with BFGS algorithm and employed during the training with β levels 0.01, 0.05 and 0.1 and dictionary cutoff level 20. Eisner normal-form constraints are employed while parsing the training data. Frequency cutoff for features in the training is set to 1, Gaussian prior σ is set to 1.3 and frequency cutoff for lexical categories is set to 10 (resulting in 425 categories in total).

5.2.2 Training Data and Active Learning Settings

We use CCGbank annotations WSJ Sections 02-21 as the global training data pool (which is the source of initial seed training set as well), including a total of 39604 sentences. To ensure statistical significance of the results, we conducted 5 experiments for each sample selection scheme (see Section 5.2.4 for sample selection schemes). In each experiment, 500 randomly sampled (with replacement) sentences are used as the seed training set and in each iteration of active learning 100 new sentences selected from the remaining of the global pool and added to the the training data (with replacement), until training set size becomes 2000.

At each step of active learning, the remaining of the data pool is parsed with the current state of the parser (to update the values of the informativeness metric for sentences in the rest of the pool). Remember that parser consists of three components: POS-tagger (using raw data as input), supertagger (using word/POS-tag pairs as input) and CCG parser (using word/POS-tag/category triples as input). At a step of active learning, supertagger and CCG parser components are trained with the training data coming from the previous iteration; however, POS-tagger component is always trained with the initial global pool (that is, with the all of the initial 39604 sentences). The experiments would be more realistic if we trained the POS-tagger component with the same data used by supertagger and CCG parser components; but amounts of training data are not sufficient to train the POS-tagger in such a setting.

As done in (Clark and Curran, 2007), we did not include sentences, during parsing of whose the maximum chart size is exceeded even at the largest β value of the supertagger.

For length-balanced sampling, a random sample of 100 sentences is needed to construct the histogram (sampling for histogram is done from the initial global pool without replacement). For statistical significance of results, we drew 5 such random samples and for each sample selection scheme based on LBS, each experiment used a histogram constructed with a different sample.

5.2.3 Evaluation

With initial seed set and after each iteration of active learning, the resulting model is tested on WSJ Section 23, consisting of 2407 sentences. As the cost metric, we used the number of *brackets* appearing in the training data. In CCGbank, a bracket corresponds to either a word-category assignment, an unary rule application or a binary rule application. Since CCGbank is constructed semi-automatically (and in large part, automatically) from Penn Treebank (Marcus, Marcinkiewicz, and Santorini, 1993), we do not have any experimental study on the annotation times of sentences in CCGbank style; however, we conjecture that the number of brackets is a realistic cost metric, since it covers all possible CCG derivation decisions. Complying (Clark and Curran,

2007), we use the labeled and unlabeled F score, precision and recall of recovery of predicate-argument dependencies as possible parsing accuracy metrics.

5.2.4 Sample Selection Schemes

5.2.4.1 Length Balanced Sampling

In LBS, we did not set a constraint on histogram sizes and considered the whole pool in random sampling. Effective histogram size is 63 on average, and maximum 84 (in 5 samples).

LBS is applied with unparsed/entropy (Becker and Osborne, 2005), tree entropy and LBP informativeness metrics in a trivial way. Unlike Becker and Osborne (2005), we normalized tree entropies in these experiments with the binary logarithm of the number of parses. We did it for comparison purposes, since tree entropies are normalized in Zipfian sampling experiments as well. Application of two-stage active learning slightly differs from (Becker and Osborne, 2005), in that among sentences that cannot be parsed by the bagged parser, sentences that can be parsed by the full parser are prioritized in sample selection (since Becker and Osborne (2005) showed that tree entropy is especially useless in such samples). As in (Becker and Osborne, 2005), the rest of the batch is filled according to non-normalized tree entropy.

5.2.4.2 Zipfian Sampling

Employing $f_{zipf-eng}$ (see Chapter 3) is straightforward with tree entropy, as it is done by selecting the sentences having highest $f_{zipf-entropy}$ values such that:

$$f_{zipf-entropy}(s) = f_{zipf-eng}(s)\frac{f_{te}(s,G)}{log_2|V|} \tag{5.1}$$

Where G is the current model and V is the set of parses of s found by G.

LBP can be combined with $f_{zipf-eng}$ in a similar way. However, lower LBP values are more desired (i.e. expected to be informative), hence we cannot directly multiply f_{lbp} with $f_{zipf-eng}$. We should use a function of f_{lbp}, which will give higher values for more desired samples. There can be many ways to do it; in this study we will comply to the method introduced in (Hwa, 2004) which uses $1-f_{lbp}$. So in the combined method,

sentences having highest $f_{zipf-lbp}$ values are selected such that:

$$f_{zipf-lbp}(s) = f_{zipf-eng}(s)(1 - f_{lbp}(s, G)) \tag{5.2}$$

Where G is the current model.

In unparsed/entropy, unparsed sentences having the highest $f_{zipf-eng}$ values are selected first. If the batch cannot be filled with such sentences, the rest of the batch is filled with the sentences having highest $f_{zipf-entropy}$ values.

In two-stage active learning, batch is filled according to the following priority scheme:

1. First priority is given to sentences that cannot be parsed by the bagged parser and can be parsed by the full parser. Such sentences are prioritized again according to their $f_{zipf-eng}$ values (higher values bring higher priority).
2. Second priority is given to sentences that cannot be parsed by both the bagged and full parsers. Such sentences are prioritized again according to their $f_{zipf-eng}$ values (higher values bring higher priority).
3. Third priority is given to sentences having highest $f_{zipf-entropy}$ values.

5.3 Experimental Results

Complying to the convention of previous studies, we report performance as PRUD values.

5.3.1 Comparison of Informativeness Measures

In the first run of experiments, we compared the performances of informativeness measures, keeping used representativeness measure fixed. The learning curves of random sampling and informativeness measures using Length Balanced Sampling can be seen in Figure 5.1 (although we give the results in terms of unlabeled F score, using other accuracy metrics do not significantly alter the results). As can be seen from the figure, all active learning methods perform better than the random baseline. Two stage active

learning performs worse among all active learning methods and entropy seems to bring the highest PRUD. Performances of unparsed/entropy and LBP are nearly identical. Performance comparison and DUR values with Length Balanced Sampling for unlabeled F score levels near 84.9% is given in Table 5.1.

Similarly, the learning curves of random sampling and informativeness measures using Zipfian sampling can be seen in Figure 5.2. In this figure, horizontal gaps between topmost points of learning curves are more distinctive. Ordering is almost the same and more definite, except that LBP performs visibly better than unparsed/entropy. Performance comparison and DUR values with Zipfian Sampling for unlabeled F score levels near 84.68% is given in Table 5.2.

Table 5.1: Performances of informativeness measures used with Length Balanced Sampling near 84.9% unlabeled F score values

Inf. Meas.	**UF(%)**	**UP(%)**	**UR(%)**	**LF(%)**	**LP(%)**	**LR(%)**	**# of brackets**	**PRUD(%)**
Random	84.87	85.77	83.99	74.38	75.14	73.57	46277	N/A
Two-stage	84.99	85.90	84.10	74.36	75.16	73.58	41169	11.04
Unparsed/entropy	84.98	85.86	84.13	74.52	75.28	73.76	38515	16.77
Tree entropy	84.92	85.79	84.08	74.44	75.20	73.70	37371	19.24
LBP	84.90	85.76	84.06	74.51	75.26	73.78	38352	17.13

Table 5.2: Performances of informativeness measures used with Zipfian Sampling near 84.68% unlabeled F score values

Inf. Meas.	**UF(%)**	**UP(%)**	**UR(%)**	**LF(%)**	**LP(%)**	**LR(%)**	**# of brackets**	**PRUD(%)**
Random	84.65	85.58	83.74	73.94	74.75	73.14	44051	N/A
Two-stage	84.69	85.68	83.72	74.05	74.91	73.20	38159	13.38
Unparsed/entropy	84.74	85.74	83.77	74.17	75.05	73.32	34459	21.77
Tree entropy	84.76	85.60	83.95	74.20	74.93	73.49	31356	28.82
LBP	84.63	85.54	83.74	74.14	74.94	73.36	32807	25.52

Table 5.3: Performances of representativeness measures used with two-stage active learning near 84.68% unlabeled F score values

Rep. Meas.	**UF(%)**	**LF(%)**	**Brackets**	**PRUD(%)**	**RelPRUD(%)**
Random	84.65	73.94	44051	N/A	N/A
LBS	84.74	74.09	39150	11.13	N/A
Zipf	84.69	74.05	38159	**13.38**	**2.53**

5.3.2 Comparison of Representativeness Measures

These are the ultimate experiments in the sense that, the performance of our novel measure is compared to its alternative, Length Balanced Sampling.

Again, PRUD values are compared at a certain unlabeled F score level. Comparative results of representativeness measures with two-stage active learning is shown in Figure 5.3. Although Zipfian sampling strictly dominates LBS, improvement is not significant. Exact figures and relative PRUD introduced by Zipfian sampling over LBS for unlabeled F score levels near 84.68% is presented in Table 5.3. In this table (and other tables in this section), precision and recall values are omitted for the sake of place. Comparative results with unparsed/entropy is shown in Figure 5.4. Here it can clearly be seen that our method dominates, providing a relative PRUD of 8.65% over LBS near 84.86% F score level (see Table 5.4). Domination persists with entropy and LBP (see Figure 5.5 and Figure 5.6), as our method provides relative PRUDs 14.88% and 12.27% over LBS with entropy and LBP, respectively, near 84.95% F score level (see Table 5.5 and Table 5.6).

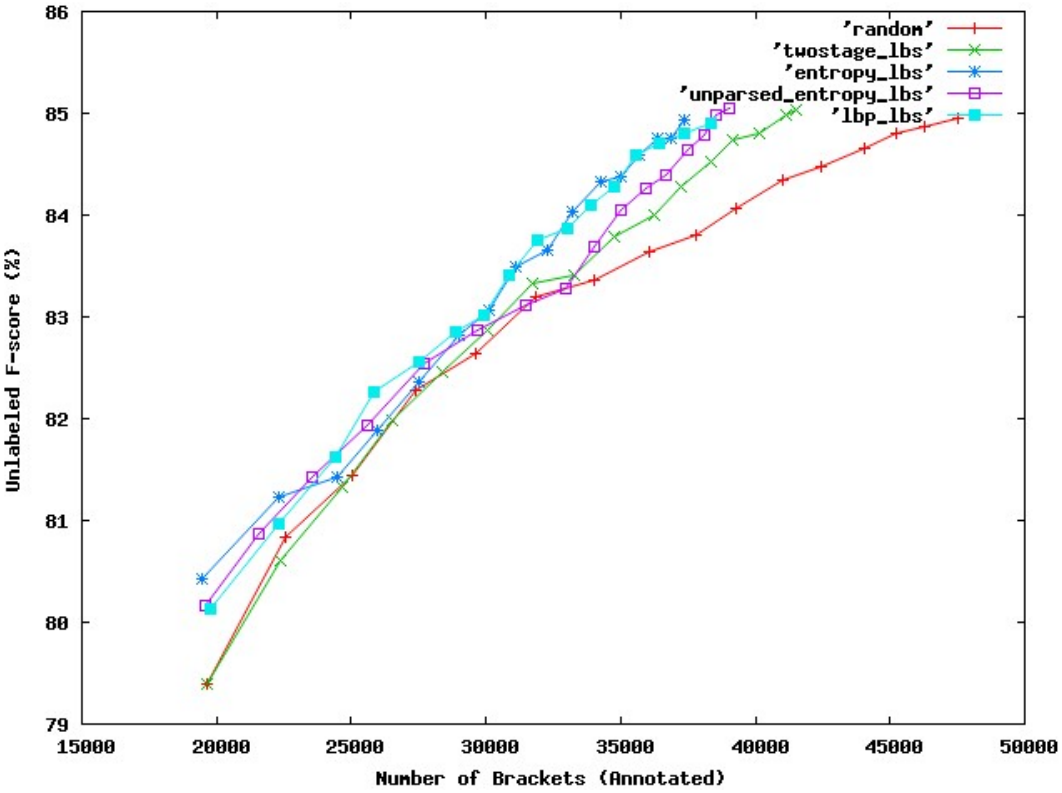

Figure 5.1: Comparative performances of informativeness measures used with Length Balanced Sampling

Table 5.4: Performances of representativeness measures used with unparsed/entropy near 84.86% unlabeled F score values

Rep. Meas.	**UF(%)**	**LF(%)**	**Brackets**	**PRUD(%)**	**RelPRUD(%)**
Random	84.87	74.38	46277	N/A	N/A
LBS	84.79	74.20	38129	17.61	N/A
Zipf	84.86	74.33	34859	**24.67**	**8.65**

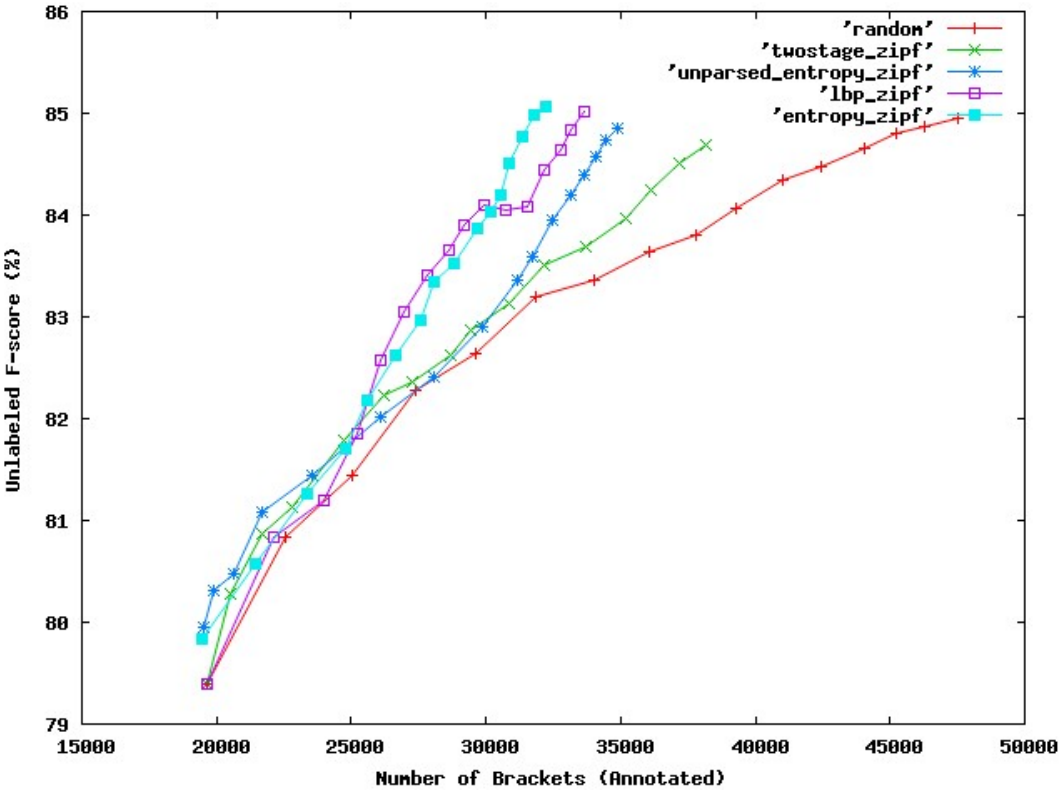

Figure 5.2: Comparative performances of informativeness measures used with Zipfian Sampling

Table 5.5: Performances of representativeness measures used with entropy near 84.95% unlabeled F score values

Rep. Meas.	**UF(%)**	**LF(%)**	**Brackets**	**PRUD(%)**	**RelPRUD(%)**
Random	84.95	74.50	47495	N/A	N/A
LBS	84.92	74.44	37371	19.24	N/A
Zipf	84.99	74.46	31811	**33.02**	**14.88**

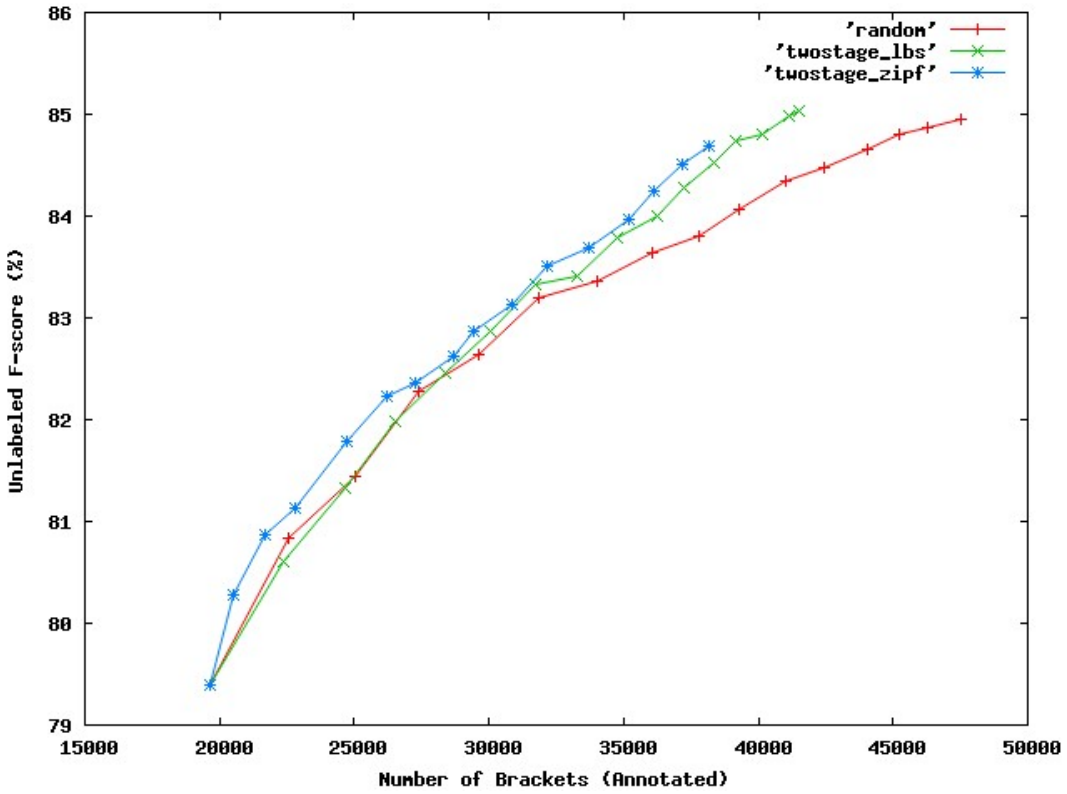

Figure 5.3: Comparative performances of representativeness measures used with Two-Stage Active Learning

Table 5.6: Performances of representativeness measures used with LBP near 84.95% unlabeled F score values

Rep. Meas.	**UF(%)**	**LF(%)**	**Brackets**	**PRUD(%)**	**RelPRUD(%)**
Random	84.95	74.50	47495	N/A	N/A
LBS	84.90	74.51	38352	19.25	N/A
Zipf	85.01	74.67	33646	**29.16**	**12.27**

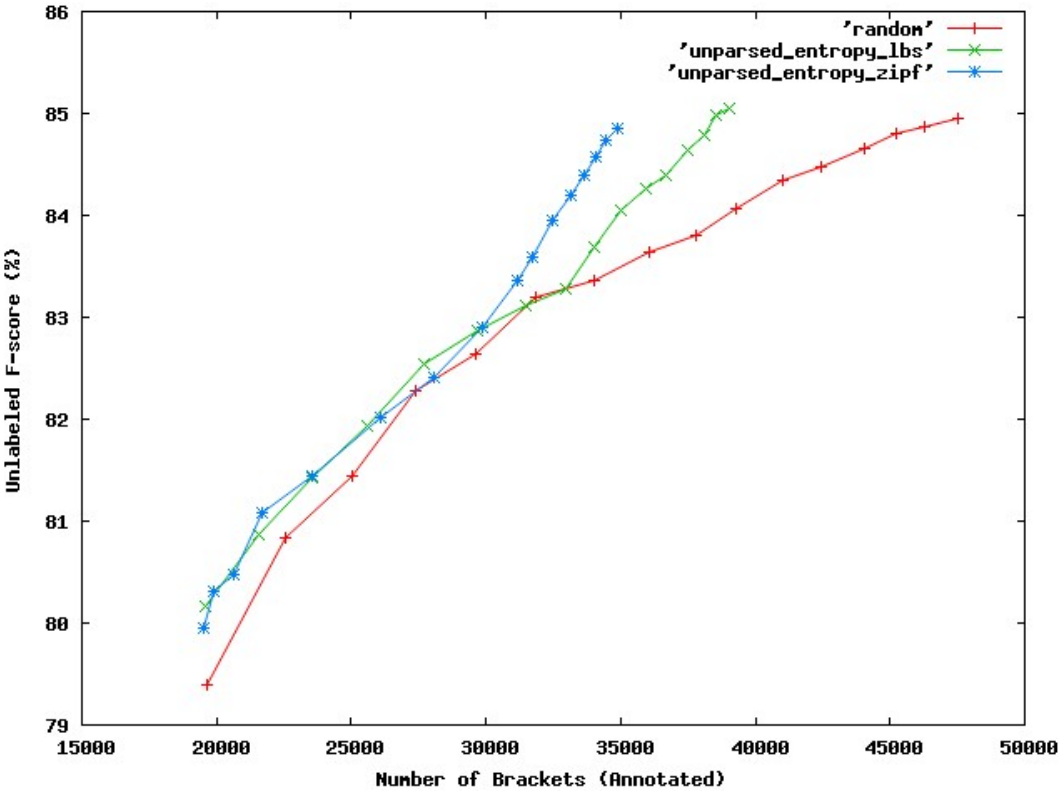

Figure 5.4: Comparative performances of representativeness measures used with unparsed/entropy

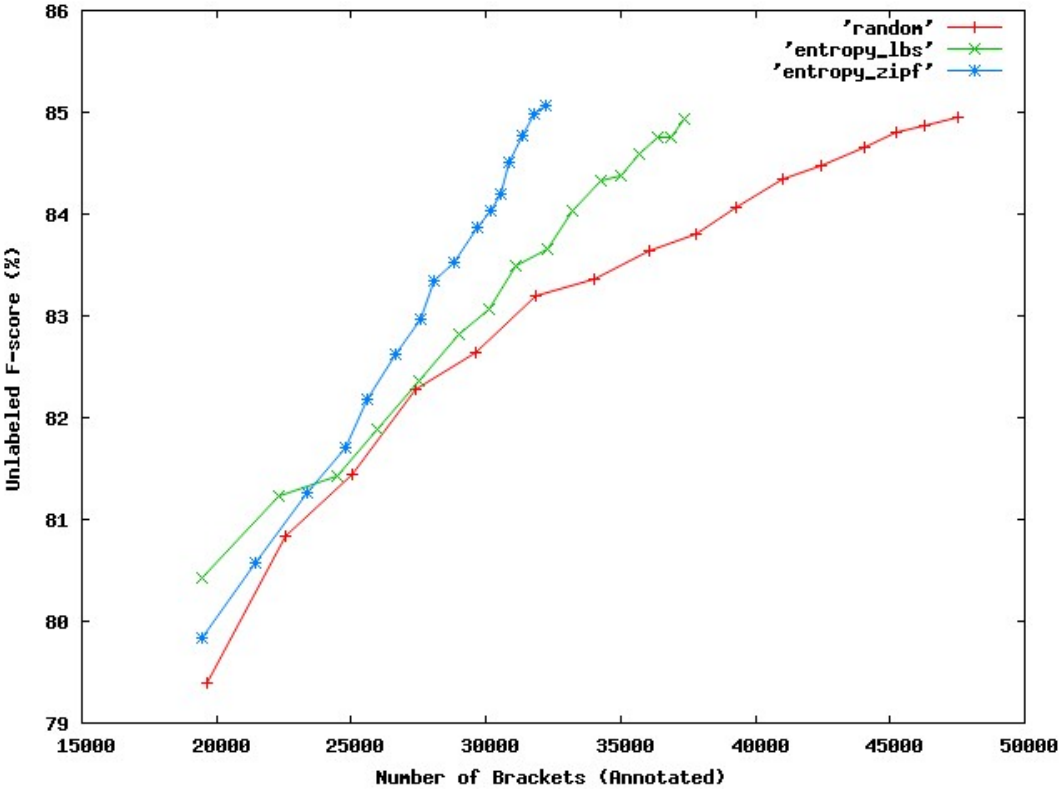

Figure 5.5: Comparative performances of representativeness measures used with entropy

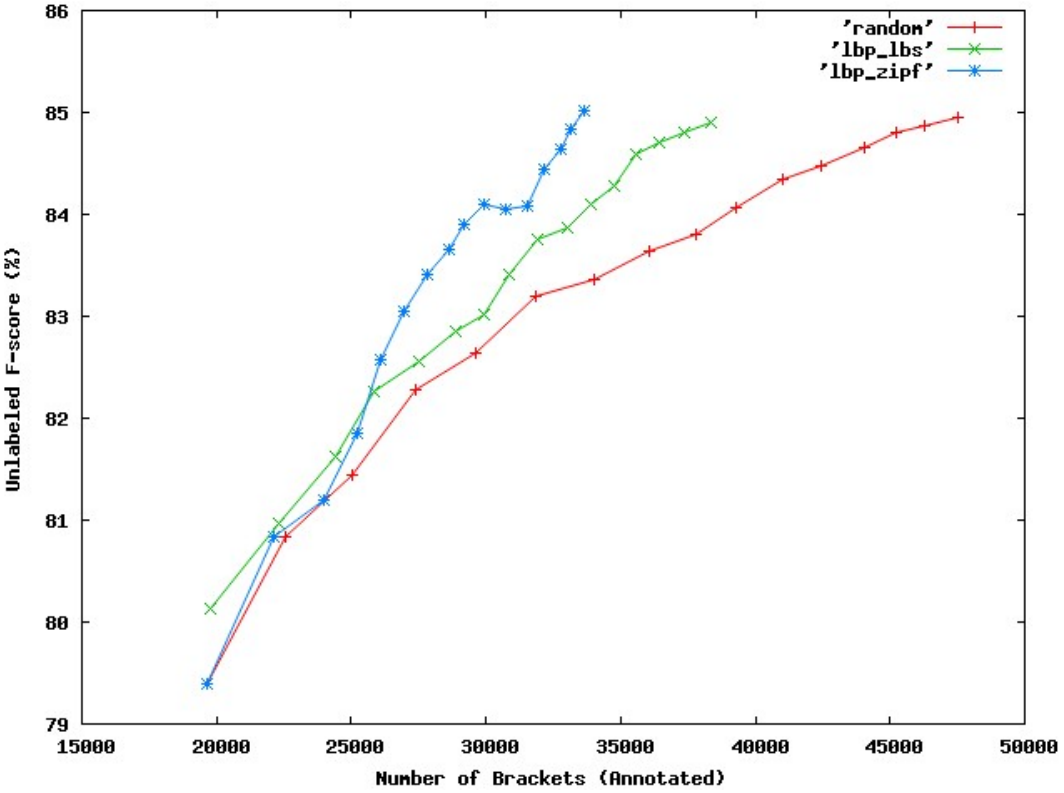

Figure 5.6: Comparative performances of representativeness measures used with LBP

CHAPTER 6

DISCUSSION AND FUTURE WORK

In this thesis, we introduced a novel representativeness measure for statistical active learning of natural language based on a revision of Zipf's Principle of Least Effort and successfully applied it to the CCGbank and C&C Parser domain. Experimental results show that our method outperforms the alternative representativeness measure (Length Balanced Sampling) consistently, significantly with most of the tried informativeness measures. We also tested the most widely-used informativeness measures for statistical active learning of natural language in C&C Parser/CCGbank domain and reported the results. Besides, we introduced an alternative method for calculating tree entropy metric, which is an important informativeness measure for statistical parser training tasks.

Experiments show that two-stage active learning methods performs worse than entropy or unparsed/entropy, contrary to the reported results. It is also observed that our method does not provide significant improvement with two-stage active learning method. The only plausible explanation we come up with to explain this phenomena is that using unparsed sentences in active learning decisions is unreliable, since supertagger failures can fool the active learner by directing it to selecting sentences which it could parse correctly if it had the true category assignment sequence. Selecting such examples can improve the supertagger accuracy but do not contribute much to the information about model parameters.

As we noted in Section 5.1, we had to exclude multi-learner active learning methods

and density estimation based measures from our experiments, for the reasons explained in the same section. In our opinion, testing and comparing Zipfian sampling with these methods and measures is the most urgent thing to do along this line of research.

We use English as the language of the experimental domain in this thesis since Zipfian frequency estimation has been done only for this language so far; however, annotated data resources are abundant in English and other languages need active learning much more than English does. Zipfian frequency estimation studies in other languages will enable applying our method in these languages. Such studies are easy to do in a short time.

REFERENCES

References

Ades, A. and M. Steedman. 1982. On the order of words. *Linguistics and Philosophy*, 4:517–558.

Angluin, D. 1987. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106.

Argamon-Engelson, S. and I. Dagan. 1999. Committee based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360.

Baldridge, J. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Baldridge, J. and M. Osborne. 2003. Active learning for HPSG parse selection. In *Proceedings of the 7th Conference on Natural Language Learning*.

Baldridge, J. and M. Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of EMNLP*.

Baldridge, J. and M. Osborne. 2008. Active learning and logarithmic opinion pools for HPSG parse selection. In *Natural Language Engineering*, volume 14. Cambridge, UK, pages 199–222.

Bar-Hillel, Y. 1953. A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Bar-Hillel, Y., C. Gaifman, and E. Shamir. 1960. On categorial and phrase structure grammars. *Bull. Res. Council Israel Sect. F*, 9F:1–16.

Becker, M. 2003. Active learning for treebank creation: A comparative study. Ph.D. Proposal for School of Informatics, University of Edinburgh, September.

Becker, M. 2008. *Active Learning: An Explicit Treatment of Unreliable Parameters*. Ph.D. thesis, University of Edinburgh.

Becker, M. and M. Osborne. 2005. A two-stage method for active learning of statistical grammars. In *Proccedings of IJCAI*, Edinburgh, UK.

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

Brinker, K. 2003. Incorporating diversity in active learning with support vector machines. In *The Twentieth International Conferenceon Machine Learning (ICML-2003)*, Washington D.C., USA, August.

Charniak, E. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI'97*, pages 598–603.

Charniak, E. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL'00*, pages 132–139.

Clark, S. and J.R. Curran. 2004a. The importance of supertagging for wide-coverage ccg parsing. In *Proceedings of COLING-04*, pages 8–88, Geneva, Switzerland.

Clark, S. and J.R. Curran. 2004b. Parsing the wsj using ccg and log-linear models. In *Proceedings of the 42nd Meeting of the ACL*, pages 104–111, Barcelona, Spain.

Clark, S. and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4).

Clark, S., J. Hockenmaier, and M. Steedman. 2001. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 327 – 334.

Cocke, J. and J.T. Schwartz. 1970. Programming languages and their compilers: Preliminary notes. Technical report, Courant Institute of Mathematical Sciences, New York University.

Collins, M.J. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics*, Madrid.

Creutz, M. 2003. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, volume 1, pages 280–287, Sapporo, Japan, July.

Curran, J.R. and S. Clark. 2003. Investigating GIS and smoothing for maximum entropy taggers. In *Proceedings of the 10th Meeting of the EACL*, pages 91–98, Budapest, Hungary.

Dan, S. 2004. Multi-criteria based active learning for named entity recognition. Master's thesis, National University of Singapore.

Dasgupta, S. 2004. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems 17*, Vancouver, Canada, December.

Davies, K., R. Donovan, M. Epstein, M. Franz, A. Ittycheriah, E. E. Jan, J. M. LeRoux, D. Lubensky, C. Neti, M. Padmanabhan, K. Papineni, S. Roukos, A. Sakrajda, J. S. Sorensen, B. Tydlitat, and T. Ward. 1999. The IBM conversational telephony system for financial applications. In *Sixth European Conference on Speech Communication and Technology*, volume 1, pages 275–278, Budapest, Hungary, September.

Eisenberg, B.B. 1992. On the sample complexity of pac-learning using random and chosen examples. Master's thesis, Massachusetts Institute of Technology, March.

Eisner, J. 1996. Efficient normal-form parsing for combinatory categorial grammar. In *Proceedings of the 34th meeting of the ACL*, pages 79–86, Santa Cruz, CA.

Flickinger, D. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Gold, E.M. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Hachey, B., B. Alex, and M. Becker. 2005. Investigating the effects of selective sampling on the annotation task. In *Proceedings of CoNLL 2005*, Ann Arbor, USA.

Hockenmaier, J. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Hockenmaier, J. and M. Steedman. 2005. *CCGbank*. Linguistic Data Consortium, Philadelphia.

Horning, J.J. 1969. *A Study of Grammatical Inference*. Ph.D. thesis, Stanford University.

Hoyt, F. and J. Baldridge. 2008. A logical basis for the D combinator and normal form constraints in combinatory categorial grammar. In *Proceedings of ACL/HLT-2008*, Columbus, OH. To appear.

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

Hwa, R. 2000. Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 45–52, October.

Hwa, R. 2001. On minimizing training corpus for parser acquisition. In *Proceedings of the Fifth Computational Natural Language Learning Workshop*, July.

Hwa, R. 2004. Sample selection for statistical parsing. *Computational Linguistics*, 30(3).

Johnson, M., S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proceedings of the 37th Meeting of the ACL*, pages 535–541, University of Maryland, MD.

Joshi, A.K., 1985. "Natural Language Parsing". chapter Tree adjoining grammars: How much context sensitivity is required ro provide reasonable structural descriptions?, pages 206–250. Cambridge University Press.

Kasami, T. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. Scientific report, Air Force Cambridge Research Lab, Bedford, MA.

Kučera, H. and W. Francis. 1967. *Computational Analysis of Present-day American English*. Providence: RI: Brown University Press.

Lang, J.K. and E.B. Baum. 1992. Query learning can work poorly when a human oracle is used. In *Proceedings of International Joint Conference on Neural Networks*, Beijing.

Liu, H., H. Motoda, and L. Yu. 2004. A selective sampling approach to active feature selection. *Artificial Intelligence*, 159(1-2):49–74.

Marcus, M.P., M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

McCallum, A.K. and K. Nigam. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning*.

Nocedal, J. and S. J. Wright. 1999. *Numerical Optimization*. New York, USA: Springer.

Oepen, S., K. Toutanova, S. Shieber, C. Manning, D. Flickinger, and T. Brants. 2002. The LinGO Redwoods Treebank: Motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan.

Osborne, M. and J. Baldridge. 2004. Ensemble-based active learning for parse selection. In *Proceedings of HLT-NAACL*.

Pereira, F. and Y. Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. *ACL*, 30:128–135.

Ratnaparkhi, A. 1996. A maximum entropy part-of-speech tagger. In *Proceedings of the EMNLP Conference*, pages 133–142, Philadelphia, PA.

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

Shewry, M.C. and H.P. Wynn. 1987. Maximum entropy sampling. *Journal of Applied Statistics*, 14(2):165–170.

Sigurd, B., M. Eeg-Olofsson, and J. van Weijer. 2004. Word length, sentence length and frequency - Zipf revisited. *Studia Linguistica*, 58(1):37, April.

Steedman, M. 2000. *The Syntactic Process*. MIT Press.

Tang, M., X. Luo, and S. Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 120–127, Philadelphia, Pennsylvania, USA, July.

Thompson, C. A., M. E. Califf, and R. J. Mooney. 1999. Active learning for natural language parsing and information extraction. In *Proceedings of 16th International Conference on Machine Learning*, pages 406–414, Bled, Slovenia.

Tong, S. 2001. *Active Learning: Theory and Applications.* Ph.D. thesis, Stanford University, August.

Vapnik, V. 1999. *The Nature of Statistical Learning Theory.* Springer-Verlag. ISBN 0-387-98780-0.

Vijay-Shanker, K. and D.J. Weir. 1994. The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

Younger, D.H. 1967. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

Zipf, G. 1935. *The psychobiology of language.* Cambridge, MA: MIT Press. Reprinted in 1965.

Zipf, G. 1949. *Human Behavior and the Principle of Least Effort.* Cambridge, MA: AddisonWesley.

APPENDIX A

COMPUTING TREE ENTROPY IN CLARK & CURRAN'S CCG PARSER

In (Hwa, 2000), a dynamic programming method for computing tree entropy of a sentence given a probabilistic context free grammar (PCFG) is given, based on a variant of CKY parsing algorithm (Cocke and Schwartz, 1970; Kasami, 1965; Younger, 1967). Although the method can be modified to work in any log-linear model, we developed an alternative method for computing tree entropy, based on CKY algorithm and the generalized grouping and additivity properties of Shannon entropy.

Before proceeding, the reader is encouraged to read the introduction part of Chapter 4, Sections 5.1 and 5.3 from (Clark and Curran, 2007).

Since a disjunctive node can be imagined to contain a group of equivalent sets of parses (where each such set corresponds to a distinct conjunctive daughter node), tree entropy H_d of a disjunctive node d is calculated as:

$$H_d = H(\frac{\phi_{c_1}}{\phi_d}, \frac{\phi_{c_2}}{\phi_d}, \ldots, \frac{\phi_{c_n}}{\phi_d}) + \sum_{i=1}^{n} \frac{\phi_{c_i}}{\phi_d} H_{c_i} \tag{A.1}$$

Where $c_1, c_2, \ldots, c_n \in \gamma(d)$, ϕ is the inside score function defined on nodes and H_{c_i} is the tree entropy of the conjunctive daughter node c_i.

A conjunctive node c corresponds to either a leaf node, or a node obtained by applying

a unary rule to a sub-parse, or a node obtained by applying a binary combination rule to two independent sub-parses (where sub-parses correspond to disjunctive daughter functions). If it is a leaf node, then there can be only a single parse leading to it, hence its tree entropy is 0. If it is a node derived using a unary or binary rule, its tree entropy H_c is:

$$H_c = \sum_{d \in \delta(c)} H_d \tag{A.2}$$

If there is a single disjunctive node, then there is no increase in the number of alternative parses rooted at c hence there is no increase in tree entropy: Tree entropy value is directly inherited from the disjunctive daughter. If there are two disjunctive daughters, then the number of alternative parses is the product of numbers of alternative sub-parses and the probability of each alternative parse is directly proportional to the product of probabilities of sub-parses leading to it. Hence, the tree entropy of the parse rooted at the conjunctive node is the sum of entropies of independent sub-parses (disjunctive daughters).

Since the set of all parses is partitioned into root disjunctive nodes, the tree entropy of a sentence S can be calculated as:

$$H_S = H(\frac{\phi_{r_1}}{Z(S)}, \frac{\phi_{r_2}}{Z(S)}, \ldots, \frac{\phi_{r_n}}{Z(S)}) + \sum_{i=1}^{n} \frac{\phi_{r_i}}{Z(S)} H_{r_i} \tag{A.3}$$

Where $r_1, r_2, \ldots, r_n \in R$ (R is the set of root disjunctive nodes), ϕ is the inside score function, H_{r_i} is the tree entropy of the root disjunctive node r_i and $Z(S)$ is the normalizing constant used in (Clark and Curran, 2007), which is basically the sum of scores of all parses of S.

We prove that tree entropy is calculated correctly by applying the equations A.1, A.2 and A.3. Before beginning the proof, we would like to remind the generalized grouping and additivity properties of the Shannon entropy:

Definition 1 (Generalized Grouping Property). *Let $P = \{p_1, ..., p_n\}$ be a set of real numbers in the interval [0,1] and $\sum_{i=1}^{n} p_i = 1$. Let $\{A_1, ..., A_k\}$ be a partition over P*

and let $A_i = \{p_{i_1}, ..., p_{i_m}\}$ *for all* i. *Let* $p(A_i)$ *be the sum of elements of* A_i. *Then generalized grouping property says that:*

$$H(p_1, ..., p_n) = H(p(A_1), ..., p(A_k)) + \sum_{i=1}^{k} p(A_i) * H(p_{i_1}/p(A_i), ..., p_{i_m}/p(A_i)) \quad \text{(A.4)}$$

Definition 2 (Additivity Property). *Let* $P_1 = \{p_{11}, ..., p_{1n}\}$ *and* $P_2 = \{p_{21}, ..., p_{2m}\}$ *be sets of real numers in the interval [0,1],* $\sum_{i=1}^{n} p_{1i} = 1$ *and* $\sum_{i=1}^{m} p_{2i} = 1$. *Then additivity property says that:*

$$H(p_{11}p_{21}, ..., p_{11}p_{2m}, p_{12}p_{21}, ..., p_{1n}p_{2m}) = H(p_{11}, ..., p_{1n}) + H(p_{21}, ..., p_{2m}) \quad \text{(A.5)}$$

In equations A.4, A.5 and throughout the rest of the proof, $H()$ function gives the Shannon entropy defined over the probability distribution denoted by the argument vector.

Proof:

Theorem 1. *Let* S *be a sentence,* $\rho(S)$ *is the set of derivations found for* S *in the feature forest of* S. *Let* $Z(S)$ *be the normalization constant defined in Section 4.3.3,* $\{r_1, \ldots, r_n\}$ *be the set of root disjunctive nodes of* $\rho(S)$, ϕ *be the inside score function,* ψ *be a function from subderivations (or derivations) to real numbers giving the total score accumulated in the subderivation (or derivation), that is,* $\psi_\omega = e^{\lambda f(\omega)}$ *and let* $\{\omega_{i1}, \ldots, \omega_{ik}\}$ *be the set of derivations collected in the root disjunctive node* r_i. *Then:*

$$-\sum_{\omega \in \rho(S)} \frac{\psi_\omega}{Z(S)} log(\frac{\psi_\omega}{Z(S)}) = H(\frac{\phi_{r_1}}{Z(S)}, \ldots, \frac{\phi_{r_n}}{Z(S)}) + \sum_{i=1}^{n} \frac{\phi_{r_i}}{Z(S)} H(\frac{\psi_{\omega_{i1}}/Z(S)}{\phi_{r_i}/Z(S)}, \ldots, \frac{\psi_{\omega_{ik}}/Z(S)}{\phi_{r_i}/Z(S)}) \quad \text{(A.6)}$$

This equation is a direct implication of the generalized grouping property (GGP, see equation A.4), since each $\psi_\omega/Z(S)$ is a real number between [0,1], they sum up to 1 and root disjunctive nodes actually create a partition over set of all ω s. Using this equation, we can calculate the tree entropy of the sentence by just calculating right hand side of the equation. We know that $Z(S)$ and ϕ_{r_i} for all i is calculable via CKY parsing; so if we can calculate $H(\frac{\psi_{\omega_{i1}}/Z(S)}{\phi_{r_i}/Z(S)}, \ldots, \frac{\psi_{\omega_{ik}}/Z(S)}{\phi_{r_i}/Z(S)})$ efficiently for all i, then we can calculate tree entropy efficiently (assuming a tractable number of root disjunctive nodes).

Now, let's define functions H_d and H_c such that:

$$H_d = H(\frac{\psi_{d^1}}{\phi_d}, \ldots, \frac{\psi_{d^k}}{\phi_d}) \quad \text{(A.7)}$$

$$H_c = H(\frac{\psi_{c^1}}{\phi_c}, \ldots, \frac{\psi_{c^k}}{\phi_c}) \quad \text{(A.8)}$$

Where d is a disjunctive node, $\{d^1, \ldots, d^k\}$ is the set of all subderivations leading to d, c is a conjunctive node and $\{c^1, \ldots, c^k\}$ is the set of all subderivations leading to c. Notice that $H(\frac{\psi_{\omega_{i1}}/Z(S)}{\phi_{r_i}/Z(S)}, \ldots, \frac{\psi_{\omega_{ik}}/Z(S)}{\phi_{r_i}/Z(S)})$ is actually H_{r_i} (when we get rid of $1/Z(S)$'s found in both numerators and denominators). So, if we can show that we can efficiently calculate H_d for all d, then we can say that we can calculate all H_{r_i} efficiently.

We claimed that H_d and H_c for any disjunctive node d and conjunctive node c can be calculated with equations A.1 and A.2, respectively. So, what we need to prove is that for all disjunctive nodes d and conjunctive nodes c the following equations hold:

$$H(\frac{\psi_{d^1}}{\phi_d}, \ldots, \frac{\psi_{d^k}}{\phi_d}) = H(\frac{\phi_{c_1}}{\phi_d}, \frac{\phi_{c_2}}{\phi_d}, \ldots, \frac{\phi_{c_n}}{\phi_d}) + \sum_{i=1}^{n} \frac{\phi_{c_i}}{\phi_d} H_{c_i} \quad \text{(A.9)}$$

$$H(\frac{\psi_{c^1}}{\phi_c}, \ldots, \frac{\psi_{c^k}}{\phi_c}) = \sum_{d \in \delta(c)} H_d \quad \text{(A.10)}$$

Where $\{c_1, \ldots, c_n\}$ are conjunctive daughters of d in equation A.9 and $\delta(c)$ is the set of disjunctive daughters of c in equation A.10.

We prove equation A.10 first: In a CKY variant parser, there are three possibilities for a conjunctive node c: It can be either (1) a leaf, (2) a conjunctive node with one disjunctive daughter and (3) a conjunctive node with two disjunctive daughters. If it's a leaf, then there will be only one subderivation leading to c, hence left hand side of the equation A.10 will be 0. Since it's a leaf, it cannot have any disjunctive daughters, so right hand side of the equation A.10 will be 0 as well.

In condition (2), a unary rule is applicable to all trees leading to daughter disjunctive node of c. So for each tree c^i leading to c, there is a sub-tree $sub(c^i)$ leading to daughter disjunctive node d_1 satisfying the following equation:

$$\psi_{c^i} = \varphi_c * \psi_{sub(c^i)} \quad \text{(A.11)}$$

Where $\varphi_c = e^{\lambda f(c)}$ and $f(c)$ is the vector of counts of occurences of features at conjunctive node c. Observe that $\{sub(c^1), ..., sub(c^k)\} = \{d_1^1, \ldots, d_1^k\}$. Moreover, $\phi_c = \varphi_c \phi_{d_1}$. So:

$$H(\frac{\psi_{c^1}}{\phi_c}, \ldots, \frac{\psi_{c^k}}{\phi_c}) = H(\frac{\varphi_c \psi_{sub(c^1)}}{\varphi_c \phi_{d_1}}, \ldots, \frac{\varphi_c \psi_{sub(c^k)}}{\varphi_c \phi_{d_1}}) \quad (A.12)$$

$$= H(\frac{\psi_{sub(c^1)}}{\phi_{d_1}}, \ldots, \frac{\psi_{sub(c^k)}}{\phi_{d_1}}) \quad (A.13)$$

$$= H(\frac{\psi_{d_1^1}}{\phi_{d_1}}, \ldots, \frac{\psi_{d_1^k}}{\phi_{d_1}}) \quad (A.14)$$

$$= H_{d_1} \quad (A.15)$$

So left hand side is equal to H_{d_1}. Right hand side is also equal to H_{d_1}, since there is a single daughter disjunctive node, which is d_1.

In condition (3), a binary rule is applicable to all tree pairs (c^{1i}, c^{2j}), where c^{1i} is a tree leading to left disjunctive daughter d_1 (stands for left sub-tree) and c^{2j} is a tree leading to right disjunctive daughter d_2 (stands for right sub-tree). Assume that there are m left sub-trees and n right sub-trees, then $\{c^{11}, \ldots, c^{1m}\} = \{d_1^1, \ldots, d_1^m\}$ and $\{c^{21}, \ldots, c^{2n}\} = \{d_2^1, \ldots, d_2^n\}$. In addition, for all i:

$$\psi_{c^i} = \varphi_c \psi_{c^{1j}} \psi_{c^{2k}} \quad (A.16)$$

Where c^{1j} is the left sub-tree of c^i and c^{2k} is the right sub-tree of c^i. Besides that, $\phi_c = \varphi_c \phi_{d_1} \phi_{d_2}$. So:

$$H(\frac{\psi_{c^1}}{\phi_c}, \ldots, \frac{\psi_{c^k}}{\phi_c}) = \frac{\tau_c^{11}}{\phi_c}, \ldots, \frac{\tau_c^{1n}}{\phi_c}, \frac{\tau_c^{21}}{\phi_c}, \ldots, \frac{\tau_c^{mn}}{\phi_c}) \quad (A.17)$$

$$= H(\frac{\tau_c^{11}}{\varphi_c \phi_{d_1} \phi_{d_2}}, \ldots, \frac{\tau_c^{1n}}{\varphi_c \phi_{d_1} \phi_{d_2}}, \frac{\tau_c^{21}}{\varphi_c \phi_{d_1} \phi_{d_2}}, \ldots, \frac{\tau_c^{mn}}{\varphi_c \phi_{d_1} \phi_{d_2}}) \quad (A.18)$$

$$= H(\frac{\psi_{c^{11}}}{\phi_{d_1}} \frac{\psi_{c^{21}}}{\phi_{d_2}}, \ldots, \frac{\psi_{c^{11}}}{\phi_{d_1}} \frac{\psi_{c^{2n}}}{\phi_{d_2}}, \frac{\psi_{c^{12}}}{\phi_{d_1}} \frac{\psi_{c^{21}}}{\phi_{d_2}}, \ldots, \frac{\psi_{c^{1m}}}{\phi_{d_1}} \frac{\psi_{c^{2n}}}{\phi_{d_2}}) \quad (A.19)$$

Where $\tau_c^{ij} = \varphi_c \psi_{c^{1i}} \psi_{c^{2j}}$. Now, observe that all $\frac{\psi_{c^{1j}}}{\phi_{d_1}}$s and $\frac{\psi_{c^{2k}}}{\phi_{d_2}}$s are real values in [0,1], $\sum_{j=1}^{m} \frac{\psi_{c^{1j}}}{\phi_{d_1}} = 1$ and $\sum_{k=1}^{n} \frac{\psi_{c^{2k}}}{\phi_{d_2}} = 1$; hence we can apply additivity property (equation A.5) to find that:

$$H(\frac{\psi_{c^{11}}}{\phi_{d_1}} \frac{\psi_{c^{21}}}{\phi_{d_2}}, \ldots, \frac{\psi_{c^{1m}}}{\phi_{d_1}} \frac{\psi_{c^{2n}}}{\phi_{d_2}}) = H(\frac{\psi_{c^{11}}}{\phi_{d_1}}, \ldots, \frac{\psi_{c^{1m}}}{\phi_{d_1}}) + H(\frac{\psi_{c^{21}}}{\phi_{d_2}}, \ldots, \frac{\psi_{c^{2n}}}{\phi_{d_2}}) \quad (A.20)$$

$$= H(\frac{\psi_{d_1^1}}{\phi_{d_1}}, \ldots, \frac{\psi_{d_1^m}}{\phi_{d_1}}) + H(\frac{\psi_{d_2^1}}{\phi_{d_2}}, \ldots, \frac{\psi_{d_2^n}}{\phi_{d_2}}) \quad \text{(A.21)}$$

$$= H_{d_1} + H_{d_2} \quad \text{(A.22)}$$

So, equation A.10 is satisfied for condition (3) as well.

For proof of equation A.9, observe that conjunctive daughters $c_1, \ldots, c_k$ of a disjunctive node d creates a partition over the set of parses $d^1, ..., d^n$ leading to d, such that any two parses d^i and d^j is in the same group if and only if they belong to the same conjunctive daughter. Assume that $\{c_i^1, \ldots, c_i^m\}$ is the set of parses leading to conjunctive daughter c_i and $\{c_i^1, \ldots, c_i^m\} = \{d^{i1}, ..., d^{im}\}$, then $\sum_{j=1}^{m} \psi_{d^{ij}} = \phi_{c_i}$. So GGP (equation A.4) says that:

$$H(\frac{\psi_{d^1}}{\phi_d}, \ldots, \frac{\psi_{d^k}}{\phi_d}) = H(\frac{\phi_{c_1}}{\phi_d}, \ldots, \frac{\phi_{c_k}}{\phi_d}) + \sum_{i=1}^{k} \frac{\phi_{c_i}}{\phi_d} H(\frac{\psi_{d^{i1}}/\phi_d}{\phi_{c_i}/\phi_d}, \ldots, \frac{\psi_{d^{im}}/\phi_d}{\phi_{c_i}/\phi_d}) \quad \text{(A.23)}$$

Dropping $\frac{1}{\phi_d}$s found in both numerators and denominators and substituting each d^{ij} with c_i^j, the last term becomes:

$$H(\frac{\psi_{c_i^1}}{\phi_{c_i}}, \ldots, \frac{\psi_{c_i^m}}{\phi_{c_i}}) \quad \text{(A.24)}$$

Which is exactly H_{c_i} according to equation A.8. So proof of equation A.9 is over.

Since H_d or H_c of a node is calculable using H function values of daughters, inside scores of daughters and itself, and since base case is the leaf conjunctive nodes whose H_c values are 0 , H_c and H_d of all nodes are calculable via CKY variants.

Our method has the same complexity with Hwa's method (2000), but it has pros and cons compared to Hwa's method. Unlike our method which does normalization at each node to obtain a valid entropy value, Hwa's method does normalization at the end of the algorithm. It introduces many additional division operators at each step in our method. However, our method guarantees that H_d and H_c are always positive. It is important since upper bound on tree entropy grows as Catalan numbers with respect to sentence length, and it may be necessary to use the logarithm of this value in the implementation. Since the H values in Hwa's method are not guaranteed to be positive, their logarithms can be undefined, which can boost memory requirements.