Active Learning
An Explicit Treatment of Unreliable Parameters

Markus Becker

Doctor of Philosophy

Institute for Communicating and Collaborative Systems
School of Informatics
University of Edinburgh

2008

Abstract

Active learning reduces annotation costs for supervised learning by concentrating labelling efforts on the most informative data. Most active learning methods assume that the model structure is fixed in advance and focus upon improving parameters within that structure. However, this is not appropriate for natural language processing where the model structure and associated parameters are determined using labelled data. Applying traditional active learning methods to natural language processing can fail to produce expected reductions in annotation cost. We show that one of the reasons for this problem is that active learning can only select examples which are already covered by the model. In this thesis, we better tailor active learning to the need of natural language processing as follows. We formulate the *Unreliable Parameter Principle*:

> Active learning should explicitly and additionally address unreliably trained model parameters in order to optimally reduce classification error. In order to do so, we should target both missing events and infrequent events.

We demonstrate the effectiveness of such an approach for a range of natural language processing tasks: prepositional phrase attachment, sequence labelling, and syntactic parsing. For prepositional phrase attachment, the explicit selection of unknown prepositions significantly improves coverage and classification performance for all examined active learning methods. For sequence labelling, we introduce a novel active learning method which explicitly targets unreliable parameters by selecting sentences with many unknown words and a large number of unobserved transition probabilities. For parsing, targeting unparseable sentences significantly improves coverage and f-measure in active learning.

Acknowledgements

Miles Osborne has been a great supervisor over the years. I would like to thank him for his invaluable input, his guidance and unwavering support and for allowing me the freedom to pursue my own ideas.

The School of Informatics at the University of Edinburgh has been an immensely inspiring environment, and I am very grateful for the many encounters and discussions I had with people here. Among them I would particularly like to mention Beatrice Alex, Amittai Axelrod, Samuel Brody, Chris Callison-Burch, Stephen Clarke, Trevor Cohn, Ben Hachey, James Henderson, Yuval Krymolowski, Mirella Lapata, Andrew Smith and David Talbot.

I am endebted to my examiners Frank Keller and Saturnino Luz for their feedback on my thesis.

Finally, I would like to thank my family - Inge and Anja Becker - for their support and understanding and Soyeon Kim for her love and patience.

I gratefully acknowledge financial support in the form of an EPSRC Doctoral Training Award provided by the School of Informatics.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Markus Becker*)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Active learning refers to a class of methods which reduce the amount of manually annotated data necessary for the supervised training of classifiers to reach a given performance level. The standard training of supervised classifiers assumes randomly sampled training data. This risks the inclusion of redundant or irrelevant data points, thereby wasting human annotation effort. By contrast, active learning directs human annotation effort towards useful data points. It has been applied in the field of machine learning, such as boundary recognition problems (Cohn et al., 1994), benchmark classification tasks (Melville and Mooney, 2004), and speech recognition (Hakkani-Tür et al., 2006). It has also been applied widely within the field of natural language processing, for instance text categorisation (McCallum and Nigam, 1998), part-of-speech tagging (Argamon-Engelson and Dagan, 1999), and parsing (Hwa,).

We can define active learning as an incremental labelling and retraining process with a human annotator in the loop as in Figure 1. First, a model C is trained on a training set of labelled examples L. Then, n new data points are chosen from a pool of unlabelled examples U, according to the current model and a selection function f. These selected data points are submitted to the annotator for labelling. Labelled data points are added to training set L and removed from the pool U. The process is repeated until either the model converges, the pool is exhausted or the human stops.

There are a wide variety of approaches to active learning, and they can generally be characterised by their choice of selection function f. Analytical solutions to the problem of finding the minimal training set are, in general, intractable except for a very few kinds of problems, for instance learning robot arm control (Cohn et al., 1996). Similarly, methods which attempt to minimise the classification error directly are also computationally impractical for most situations (Roy and McCallum, 2001).

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

Figure 1.1: Generic active learning algorithm, pseudo-code due (Hwa, 2000).

For this reason, most research in active learning has focused on computationally cheaper methods such as *uncertainty sampling* and *Query-by-Committee* (QBC). Uncertainty sampling selects examples where the currently trained classifier is uncertain with regard to the preferred labelling (Lewis and Catlett, 1994). QBC selects examples for which an ensemble of classifiers shows a high degree of disagreement with respect to the preferred labelling (Seung et al., 1992). Although both uncertainty sampling and QBC appear to yield excellent results, a closer examination shows that, typically, practitioners do not apply these techniques in a 'pure' setting. That is, each example selected for labelling may not be the most informative one (as predicted by active learning). As we show in this thesis, the reason for this is that active learning methods typically do not directly address the problems of dealing with sparse data.

Sparse data problems are ubiquitous in natural language processing due to the Zipfian nature of language. Zipf's law states that a small number of words (or event types in general) occur very often, whereas the majority of word (event types) occurs very rarely (Zipf, 1935). In a machine learning scenario, this means that some rare events will be missing completely from a training set, even when it is very large. Accordingly, parameter estimation for such rare events can be unreliable in two ways, depending on whether a rare event in question has been observed in the training set or not. We will refer to these two types of situations as *infrequent events* and *missing events*.

Parameter estimation for infrequent events suffers from a high degree of variance and can result in degraded classification performance. The same holds for situations

where an event is entirely missing from the training set, but we stipulate its presence. For instance, in the parameter estimation for Hidden Markov models, some probability mass is commonly reserved for stipulated, yet unobserved transition probabilities, in other words, probability estimates without corresponding n-grams in the training set, in order to avoid zero probability transitions. However, such smoothing methods merely alleviate variance problems, and do not solve them.

Missing events can also result in a loss of *coverage*, a situation when the structure of the probabilistic model is not rich enough to cover all types of input. For *out-of-coverage* examples, the classifier may not be able to predict any label at all or only output a generic label. Example for this are complete parse failures due to missing rules in the case of syntactic parsing, unknown prepositions in prepositional phrase attachment, and unknown words in part-of-speech tagging.

An important observation is that both uncertainty sampling and QBC have shortcomings with regard to unreliably estimated parameters. Neither method has a principled way to deal with missing events, because they can only refine probability estimates *within* a given model structure. We will demonstrate this repeatedly in later chapters, in a variety of applications. One example is that of uncertainty sampling applied to the learning of a state-of-the-art parser (Bikel, 2004a). As we will show in more detail in Section 6.2, a standardly smoothed parser fails to select examples which would provide novel and important parsing events. We show that entropy alone does not reliably identify such examples. Another illustrative case is that when applying QBC to the prepositional phrase attachment method of (Collins and Brooks, 1995) without further adjustments, QBC will assign minimal disagreement scores to examples with unknown prepositions and hence ignore them. We will show in Section 4.2 that this can result in important prepositions missing from the training set and in severe coverage problems.

Uncertainty sampling may not properly address infrequently observed events either. In fact, uncertainty sampling prefers examples with high entropy distributions regardless of the underlying empirical counts. In chapter Section 4.1, we show that uncertainty sampling applied to prepositional phrase attachment can lead to a situation where annotation effort is stubbornly directed towards a single preposition – and away from all other prepositions – for many iterations on end simply because this preposition happens to have a high entropy distribution. This is inappropriate because the remaining prepositions will continue to suffer from infrequently observed events. QBC can, in fact, address the problem of infrequently observed events but only indirectly. QBC is

typically based on methods that perturb the model distributions. Infrequently observed events will exhibit a higher degree of variance and thus contribute to disagreement within the ensemble. However, due to the stochastic nature of model perturbation, we are not guaranteed to select the appropriate examples.

1.1 Explicit Treatment of Unreliable Parameters

In view of these shortcomings of active learning, we state the following principle:

> **Unreliable Parameter Principle**:
>
> Active learning should explicitly and additionally address unreliably trained model parameters in order to optimally reduce classification error. In order to do so, we should target both missing events and infrequent events.

We demonstrate how this principle applies to a variety of problems, namely prepositional phrase attachment, part-of-speech tagging, named entity recognition (NER) and syntactic parsing. We have chosen these tasks as representative of typical natural language processing applications and for having various degrees of difficulty. Prepositional phrase attachment, the simplest of the tasks, only requires binary classification. Part-of-speech tagging and NER are representative of sequence labelling tasks and are therefore more difficult than mere classification. Syntactic parsing, which requires the assignment of structured labellings, presents an even higher degree of difficulty. Because of the different characteristics of these tasks, the actual implementation of the Unreliable Parameter Principle varies.

Prepositional Phrase Attachment Completely unknown prepositions are a particularly severe manifestation of missing events for this task. We show that the preferred selection of unknown prepositions results in improved classification accuracy. This can be achieved within the framework of standard methods such as uncertainty sampling QBC by using appropriate backoff probabilities during sample selection to flag such instances. This approach addresses the *missing events* aspect of the Unreliable Parameter Principle.

Sequence Labelling We introduce a novel active learning method suitable for sequence labelling tasks. This method explicitly targets unreliable model parameters by selecting sequences with many unknown words and a large expected number of unobserved transition probabilities. This method, too, addresses the *missing events* aspect of the Unreliable Parameter Principle.

Syntactic Parsing We demonstrate that the preferred selection of unparsable sentences is a surprisingly strong active learning method. Interestingly, this method requires that the parser runs with a lesser degree of smoothing. By targeting missing parsing rules, we again address the *missing event* aspect of the Unreliable Parameter Principle. We also present a novel two-stage method which selects unparsable sentences according to a parser which was trained on a perturbed training set. This method implements both the *missing event* and the *infrequent event* aspect of the Unreliable Parameter Principle.

1.2 Contributions of the Thesis

The main contributions of this thesis fall into the following categories:

Comparison between Uncertainty Sampling and QBC

Given their popularity, we consider uncertainty sampling and QBC as points of departure and as important benchmarks throughout this thesis. We provide an extensive comparison of these methods in application to three different natural language processing tasks of varying degrees of difficulty. Within QBC, we explore i) alternative methods of creating diverse ensembles, ii) different divergence metrics, and iii) influence of ensemble size. Interesting results from this part of the thesis include:

- Generally, both uncertainty sampling and QBC outperform random sampling, as expected. In most cases, we find that QBC beats uncertainty sampling. However, in named entity recognition uncertainty sampling outperforms QBC.
- For QBC, there is no particular setting of ensemble creation method or divergence metric which is always guaranteed to be best. Rather, appropriate settings vary from application to application.

In general, these findings suggest caution when choosing active learning methods for novel applications. Unfortunately, the explorable alternative active learning methods are restricted due to a shortage of training material in exactly those situations when the use of active learning is desirable.

To the best of our knowledge, a comprehensive comparison between uncertainty sampling and QBC across different tasks has not been presented in previous literature. Similarly, the exploration of the experimental parameter space for QBC is novel.

Explicitly Targeting Unreliable Parameters via Smoothing/Backing-Off

Choosing appropriate smoothing or backoff settings in active learning can be vital to allow for the targeted selection of out-of-coverage examples. We demonstrate this for prepositional phrase attachment and for parsing.

- For prepositional phrase attachment, assigning an appropriate backoff probability helps to select unknown prepositions and substantially improves coverage and accuracy for both uncertainty sampling and QBC.
- For parsing, switching off *constraint relaxation* (an effective smoothing method) helps to select out-of-coverage examples and substantially improves coverage and f-measure. Again, this method combines gracefully with uncertainty sampling and QBC and results in improved performance in each situation.

This goes against the perceived wisdom in the literature that the same model should be used in the sample selection phase and in deployment, for example (Baldridge and Osborne, 2004). Even for standard methods such as uncertainty sampling and QBC, using an off-the-shelf classifier may not be the best option. On the contrary, we find that the best settings for sample selection may not be appropriate for deployment. In summary, we should consider settings which promote the selection of out-of-coverage examples. The successful application of these techniques supports our thesis that active learning should explicitly address unreliably trained parameters.

Explicitly Targeting Unreliable Parameters via Other Methodologies

Beyond adjusted smoothing, we introduce two novel methods in this thesis which explicitly target either out-of-coverage examples or variance in parameter estimates.

- For sequence labelling tasks, we introduce a novel method which targets unsupported model parameters. For part-of-speech tagging, this method in isolation performs as well as, or better than, uncertainty sampling. In combination with uncertainty sampling, we match QBC at its optimal setting.
- For parsing, we present a novel two-stage method which selects unparsable sentences according to a parser which was trained on a perturbed training set. Again, this method works as well as the best QBC setting for parsing.

The success of both of these techniques supports once again our thesis regarding unreliable parameters.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows.

- In Chapter 2, we first provide the background for the different tasks to which we apply our active learning methods: prepositional phrase attachment, sequence labelling, and syntactic parsing. Next, we cover relevant literature for different active learning methods with a special focus on uncertainty sampling and Query-by-Committee.
- In Chapter 3, we set out the experimental conditions for all the experiments in subsequent chapters, including evaluation metrics and statistical significance tests.
- In Chapter 4, we present results for active learning as applied to prepositional phrase attachment.
- In Chapter 5, we present results for sequence labelling. In particular, we apply active learning to part-of-speech tagging and named entity recognition.
- In Chapter 6, we present results using active learning for syntactic parsing.
- Chapter 7 discusses the implications of the experimental results and suggests future work.

Chapter 2

Literature Review

This chapter describes the background literature relevant to the remainder of this thesis. In the first three sections, we give a survey of those natural language processing tasks to which we apply our active learning methods. In Section 2.1, we discuss prepositional phrase attachment as a suitable testing ground for active learning. In Section 2.2, we review sequencing models with applications to part-of-speech tagging and named entity recognition. In Section 2.3, we cover the relevant literature on syntactic parsing. We give a more detailed overview of one representative model for each of the tasks as we use it in the corresponding experimental sections in later chapters and discuss relevant applications and available data sets. In Section 2.4, we examine relevant literature in the field of active learning with a special focus on uncertainty sampling and Query-by-Committee. We conclude in Section 2.5.

2.1 Prepositional Phrase Attachment

Attachment choices present a common source of ambiguity in the syntactic analysis of sentences. In prepositional phrase attachment (PPA), one predicts whether a prepositional phrase is attached to the verb in the matrix clause or to the preceding noun phrase. This is useful for determining the argument structure of a sentence and its semantic interpretation. Suppose we would like to analyse the following sentence:[1]

He bought a car with a steering wheel.

The sentence receives different interpretations, depending on whether the prepositional phrase 'with a steering wheel' is attached to the verb 'bought' or to the noun phrase 'a car' in the analysis of the verbal phrase.

[1] This example is adapted from (Brill and Resnik,).

NP-attached: (bought ((a car) (with a steering wheel)))
VP-attached: ((bought (a car)) (with a steering wheel))

World knowledge helps us to decide that the prepositional phrase should be noun-attached which yields the intended interpretation of 'the purchase of a car which is equipped with a steering wheel'.

2.1.1 Approaches

Altmann and Steedman, 1988 argued that, at least in some cases, we may need to resort to a discourse model for proper PPA disambiguation. However, subsequent work showed that we can disambiguate successfully based only on local lexical information in the majority of cases. Seminal work by Hindle and Rooth, 1993 demonstrated the feasibility of a corpus-based approach. Setting the scene for most later research, they showed that attachment can be predicted fairly accurately based only on lexical head information for the involved verb, noun, and prepositional phrases. They applied unsupervised learning, using co-occurrence information from a corpus of 13 million words of AP news. Later approaches showed that even larger corpora can further improve estimation, for example by tapping into the web as a corpus (Volk, 2000).

Not surprisingly, supervised approaches based on annotated tuples extracted from the Penn treebank yield better results. Successful approaches include maximum entropy modelling (Ratnaparkhi et al., 1994) and transformation-based learning (Brill and Resnik,).

2.1.2 An Exemplary PPA Approach – Collins & Brooks, 1995

Collins and Brooks, 1995 present a supervised training approach involving maximum likelihood estimation (MLE) and backing-off for sparse data problems. They use a training set which was extracted from the Penn treebank by (Ratnaparkhi et al., 1994). Each training instance is a quadruple of the form $\langle v, n, p, n_2 \rangle$ together with the correct attachment information, where v is the verb, n the first head noun, p the preposition, and n_2 the second head noun, for example:

$\langle joined, board, as, director \rangle \rightarrow \mathbf{v}$
$\langle bought, car, with, wheel \rangle \rightarrow \mathbf{n}$

The naïve application of MLE suggests the following frequency-based estimation for the probability of a quadruple to be noun-attached:

$$
\begin{array}{lll}
f_1 = \langle v, n, p, n_2 \rangle & & \\
f_2^1 = \langle v, p, n_2 \rangle, & f_2^2 = \langle n, p, n_2 \rangle, & f_2^3 = \langle v, n, p \rangle \\
f_3^1 = \langle v, p \rangle, & f_3^2 = \langle n, p \rangle, & f_3^3 = \langle p, n_2 \rangle \\
f_4 = \langle p \rangle & &
\end{array}
$$

Table 2.1: Feature sets for prepositional phrase attachment at different backoff levels

$$P(\mathbf{n}|\langle v, n, p, n_2 \rangle) = \frac{C_{\mathbf{n}}(\langle v, n, p, n_2 \rangle)}{C(\langle v, n, p, n_2 \rangle)} \tag{2.1}$$

where $C(\cdot)$ counts all occurrences of a tuple, and $C_{\mathbf{n}}(\cdot)$ counts all noun-attached occurrences. The problem with this approach is that the feature set is too specific in most cases so that the estimate will be undefined for test instances which are not in the training set.

To counter this problem, Collins and Brooks use a backoff scheme based on work in speech recognition (Katz, 1987). Their backoff scheme is defined as follows:

$$
\begin{array}{rcl}
P_1(\mathbf{n}|u) & = & \dfrac{C_{\mathbf{n}}(f_1(u))}{C(f_1(u))} \\
P_2(\mathbf{n}|u) & = & \dfrac{C_{\mathbf{n}}(f_2^1(u)) + C_{\mathbf{n}}(f_2^2(u)) + C_{\mathbf{n}}(f_2^3(u))}{C(f_2^1(u)) + C(f_2^2(u)) + C(f_2^3(u))} \\
P_3(\mathbf{n}|u) & = & \dfrac{C_{\mathbf{n}}(f_3^1(u)) + C_{\mathbf{n}}(f_3^2(u)) + C_{\mathbf{n}}(f_3^3(u))}{C(f_3^1(u)) + C(f_3^2(u)) + C(f_3^3(u))} \\
P_4(\mathbf{n}|u) & = & \dfrac{C_{\mathbf{n}}(f_4(u))}{C(f_4(u))} \\
P_5(\mathbf{n}|u) & = & 1.0
\end{array} \tag{2.2}
$$

where feature tuples f are as defined in Table 2.1. As we can see, feature tuples are increasingly general for later backing-off stages while always retaining the preposition.

The probability $P(\mathbf{n}|u)$ is taken to be $P_i(\mathbf{n}|u)$ at the most specific backoff level i (lowest i) for which the value is defined. This is the case where the denominator is non-zero. An instance is classified as noun-attached if $P(\mathbf{n}|u) > 0.5$. If none of the feature tuples match on the first four levels, $P(\mathbf{n}|u)$ is set to 1.0 and the instance will be (deterministically) classified as noun-attached.

To give an example from (Collins and Brooks, 1995), assume that we would like to classify the instance $\langle joined, board, as, director \rangle$ and that none of these features have been observed on the first three backoff levels. However, there are 4 instances which

match on the fourth level ($f_4 = \langle as \rangle$), 3 out of which are labelled as noun-attached. This instance would be classified as noun-attached.

$$P_4(\mathbf{n}|u) = \frac{C_{\mathbf{n}}(\langle as \rangle)}{C(\langle as \rangle)} = \frac{3}{4} \tag{2.3}$$

Collins and Brooks achieve 84.1% classification accuracy. Applying preprocessing, for instance by replacing four-digit numbers with the string 'YEAR', yields a further increase in performance up to 84.5%. Interestingly, they get best results by retaining all low count events.

A baseline method for PPA which always votes for noun-attachment yields 59% accuracy. By contrast, assigning the most likely attachment for each preposition increases accuracy to 72.2%. For practical purposes, one can reliably guess the most likely attachment by labelling just a few instances for each preposition. This insight motivates the idea that active learning methods in application to PPA should pursue coverage so that every preposition has been seen at least a few times. We explore this idea in Chapter 4.

2.1.3 Summary

We have outlined the prepositional phrase attachment task and discussed relevant literature. Particular attention was given to the approach of (Collins and Brooks, 1995) which we will use for our experiments in Chapter 4.

2.2 Sequence Labelling

Sequencing tasks imply the assignment of labels to tokens in a sequence. While this task superficially looks like a classification task, labelling decisions for individual tokens will typically inform each other and best results are achieved by aiming at a globally optimal decision. Prominent examples for sequencing tasks are part-of-speech tagging and named entity recognition.

Part-of-speech tagging

Part-of-speech tagging is the assignment of syntactical part-of-speech tags to the words in a sentence based on their context. For instance, the sentence "This is not a trivial issue ." will be labelled as follows:

This/DT is/VBZ not/RB a/DT trivial/JJ issue/NN ./.

Here, ‘DT’ stands for determiner, ‘VBZ’ for an inflected verb, and so on. A challenge in part-of-speech tagging is the accurate labelling of ambiguous words, for example the word ‘issue’ which can either be a noun or a verb. A part-of-speech tagger can reliably decide for the noun reading in this case given the determiner and adjective in the preceding context.

Named entity recognition

Named entity recognition (NER) is the task of identifying and classifying (non-embedded) phrases in a text as belonging to a set of predefined entities such as person and company names, locations, and time and date expressions. In the following example, we identify ‘Karim Alami’ as a person name and ‘Morocco’ as a location.

He meets [PER Karim Alami] of [LOC Morocco] .

NER was first introduced in the Sixth Message Understanding Conference in 1995 (MUC-6) as a task separate from information extraction, (Grishman and Sundheim, 1996). In a strict sense, NER is a structural labelling task, but, using a suitable token-wise representation for brackets, sequence labelling has been successfully applied to this task. In fact, the majority of recent work on NER addresses the problem as a sequence labelling problem as is evidenced for example by the entries for the shared NER tasks of the Conference on Computational Natural Language Learning (CoNLL) in 2002 and 2003 (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003). Using the IOB labelling scheme of Ramshaw and Marcus, 1995, tokens within an entity are marked by their entity type, prefixed with ‘B’ for tokens at the beginning of an entity and with ‘I’ for all other tokens in an entity; non-entities are marked ‘O’. Thus, the bracketing above is represented in the IOB scheme as follows:

He/O meets/O Karim/B-PER Alami/I-PER of/O Morocco/I-LOC ./O

2.2.1 Applications

Sequencing applications have a variety of applications in natural language processing. Part-of-speech taggers and NER systems can be used as components in information extraction systems as the first essential processing stage where relevant entities are identified before relations between entities are established. Part-of-speech taggers are also

commonly used to provide input for parsers as a method for reducing the search space. This has been shown to increase both speed and parsing accuracy (Charniak, 1996; Prins and van Noord, 2003). In text-to-speech systems, taggers are used to provide more information for the correct pronunciation. Typical examples include words which are ambiguous in their written form. For instance, the word 'object' is stressed on the first syllable as a noun, and on the second as a verb. Lastly, automatic part-of-speech tagging in combination with subsequent human correction is commonly used in corpus annotation and is significantly faster than human annotation from scratch (Skut et al., 1997). This is often done incrementally so that previously annotated sentences are added to the training set in order to improve performance and better guide humans in their annotation of later sentences.

2.2.2 Approaches

Manual writing of rule-based systems has been shown to be effective, for example the ENGTWOL system for English part-of-speech tagging (Voutilainen, 1995). This tagger has been shown to outperform automatically trained taggers (Samuelsson and Voutilainen, 1997). The majority of modern systems, however, are probabilistic taggers (Garside et al., 1987; Church, 1988; DeRose, 1988; Brants, 2000b). These have the advantage that they can be automatically trained on annotated corpora.

2.2.3 An Exemplary Sequencing Approach – Brants, 2000

In the following, we describe the Trigrams'n'Tags tagger (TnT) as an exemplary statistical Markov model tagger (Brants, 2000b).[2] A Markov model describes a stochastic process which generates sequences of symbols. The structure of a Markov model is given by a set of states and transitions between states. States emit symbols with an *emission probability* and transitions between states are taken with a *transition probability*. In many NLP applications, states represent tags and emissions represent words. Markov models have the *Markov property*: transition probabilities only condition on a path history of a fixed length. TnT uses a second-order Markov model, so that transition probabilities condition only on the previous two states.

Sequential decoding implies finding the most probable tag sequence $t_1 \dots t_T$ given the words $w_1 \dots w_T$ of a sentence:

[2] For our experiments in Chapter 5 we will use the freely available implementation of (Schröder, 2002) which is based on the TnT tagger.

$$t_1 \dots t_T = \underset{t_1 \dots t_T}{\operatorname{argmax}} P(t_1 \dots t_T | w_1 \dots w_T) \tag{2.4}$$

This expression can be simplified by assuming that words are generated conditionally independently of each other and only dependent on their individual tags, and by applying the Markov assumption:

$$\begin{aligned}
P(t_1 \dots t_T | w_1 \dots w_T) &= \frac{P(w_1 \dots w_T | t_1 \dots t_T) P(t_1 \dots t_T)}{P(w_1 \dots w_T)} && (2.5)\\
&\propto P(w_1 \dots w_T | t_1 \dots t_T) P(t_1 \dots t_T) && (2.6)\\
&= \left[\prod_{i=1}^{n} P(w_i | t_1 \dots t_T) \right] P(t_1 \dots t_T) && (2.7)\\
&= \left[\prod_{i=1}^{n} P(w_i | t_i) \right] P(t_1 \dots t_T) && (2.8)\\
&= \left[\prod_{i=1}^{n} P(w_i | t_i) \right] \times P(t_T | t_{T-1} \dots t_1) && (2.9)\\
&\quad \times P(t_{T-1} | t_{T-2} \dots t_1) \times \dots \times P(t_2 | t_1) \times P(t_1)\\
&= \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1}, t_{i-2}) && (2.10)
\end{aligned}$$

Thus, the tagging problem reduces to the following expression:

$$t_1 \dots t_T = \underset{t_1 \dots t_T}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i | t_i) P(t_i | t_{i-1}, t_{i-2}) \tag{2.11}$$

This problem can be solved efficiently using Viterbi decoding (Viterbi, 1967). This is a dynamic programming technique where intermediate results are stored in a trellis structure for later reuse.

Training such a model requires estimation of the transition probabilities $P(t_i | t_{i-1}, t_{i-2})$ and the emission probabilities $P(w_i | t_i)$. In Brants, 2000b, this is done using standard maximum likelihood estimation using relative frequencies from manually annotated data, and appropriate backoff/smoothing schemes for sparse data problems.

Transition Probabilities

The transition probability $P(t_i | t_{i-1}, t_{i-2})$ is also called *trigram* probability because it involves a sequence of three tags. Estimating trigram transition probabilities from relative frequencies only, either observed or unobserved, would suffer from many unseen event types due to data sparseness, and, therefore, some kind of smoothing is required.

TnT combines trigram transition probabilities with transition probabilities of shorter sequences in the conditioning history via *linear interpolation*. *Unigrams* $\hat{P}(t_3)$ have no history, *bigrams* $\hat{P}(t_3|t_2)$ have a transition history of length one:

$$\hat{P}(t_3) = \frac{f(t_3)}{N} \tag{2.12}$$

$$\hat{P}(t_3|t_2) = \frac{f(t_2,t_3)}{f(t_2)} \tag{2.13}$$

$$\hat{P}(t_3|t_1,t_2) = \frac{f(t_1,t_2,t_3)}{f(t_1,t_2)} \tag{2.14}$$

where $f(\cdot)$ are frequencies of observed events and N is the size of the training set.

The linear interpolation of n-grams uses appropriate λ_i values to arrive at an estimate for conditional trigram tag probabilities:

$$P(t_3|t_1,t_2) = \lambda_1\hat{P}(t_3) + \lambda_2\hat{P}(t_3|t_2) + \lambda_3\hat{P}(t_3|t_1,t_2) \tag{2.15}$$

such that $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Values for λ_i are estimated on the training set using *deleted interpolation* (Jelinek and Mercer, 1980). This technique successively deletes each trigram in turn and estimates λ values using the remaining n-grams in the corpus.

Emission Probabilities Emission probabilities for known words are maximum likelihood probability estimates based on relative frequencies:

$$\hat{P}(w|t) = \frac{f(w,t)}{f(t)} \tag{2.16}$$

Emission probabilities for unknown words are estimated using a suffix analysis in a process called *successive abstraction* (Samuelsson, 1993). For example, words ending in -able are most likely to be adjectives, for instance capable, recognisable, whereas words ending in -ion are most likely to be nouns (station, satisfaction, etc). The probability distribution for a particular suffix is derived from words in the training set that share the same suffix:

$$P(t|\text{-able}) = \frac{f(t,\text{-able})}{f(\text{-able})} \tag{2.17}$$

Probabilities are smoothed by combining the distributions of shorter suffixes:

$$P(t|l_{n-i+1},\ldots,l_n) = \frac{\hat{P}(t|l_{n-i+1},\ldots,l_n) + \theta_i P(t|l_{n-i},\ldots,l_n)}{1+\theta_i} \tag{2.18}$$

The value of θ is based on the standard deviation of unconditional MLE tag probabilities, see (Samuelsson, 1993) for details. $P(l_{\text{suff}}|t)$ is obtained by Bayesian inversion. The lookup procedure can be efficiently implemented using a suffix trie. The training set for parameter estimation is restricted to infrequent words since infrequently words have different tag output distributions from frequently observed ones (Dermatas and Kokkinakis, 1995; Baayen and Sproat, 1996).

2.2.4 Annotated Resources

Part-of-Speech Tagging

A pioneering effort in corpus linguistics was the compilation of the Brown University Standard Corpus of Present-Day American English (Brown Corpus) by Francis and Kučera in the 1960s (Francis and Kučera, 1964; Kučera and Francis, 1967). This corpus contains over a million words of modern American English from a variety of genres. The first editions of the corpus were just raw, unannotated text, but over time the entire corpus was part-of-speech tagged with a tagset of 87 tags, using a combination of automatic tagging and manual correction. A similar effort has been made for British English with the Lancaster-Oslo-Bergen corpus, also using the Brown tagset (Johansson et al., 1978).

The Penn treebank is best known as an annotated resource for syntactic tree structures (Marcus et al., 1993), and we will review it in more detail in Section 2.3. However, the Penn treebank is also annotated with part-of-speech tags. The Penn treebank tagset is a simplified version of the Brown tagset with only 45 tags (Santorini, 1990).

NER

NER was first introduced in the Sixth Message Understanding Conference in 1995 (MUC-6) as a task separate from information extraction, (Grishman and Sundheim, 1996). Here the target language was English. NER also featured in the shared tasks of the Conference on Computational Natural Language Learning (CoNLL). Dutch and Spanish were target language at CoNLL-2002, (Tjong Kim Sang, 2002), English and German at CoNLL-2003 (Tjong Kim Sang and De Meulder, 2003). These data sets are still commonly used for research purposes.

2.2.5 Summary

Despite the success of manually constructed taggers, the predominant paradigm is to use stochastically trained taggers, due to their very good performance and easy trainability. With the availability of large annotated corpora it may seem as if tagging is a solved problem. However, only a small fraction of the world's languages have such annotated corpora. With an interest of the research community to acquire annotated resources for these languages for a wide range of potential applications such as machine translation the annotation bottleneck continues to be a pressing problem. In Section 2.4, we will look at active learning methods which can alleviate the cost problem of annotation.

2.3 Parsing

Parsing is the process of recognising sentences as grammatical with respect to a grammar while simultaneously assigning syntactical structure to the input. Let us consider the syntactic analysis of the sentence "They fell into oblivion after the 1929 crash" in Figure 2.1. As in NER, we bracket associated words in the sentence, for example "the 1929 crash" is considered to be a noun phrase. In contrast to NER, brackets can be recursive like the noun phrases which are embedded in the prepositional phrases. Furthermore, analysis is usually complete in parsing, in the sense that all words in a sentence will be part of a phrase.

2.3.1 Applications

Parsing has been employed in a variety of applications, among others in grammar and style checking (Thurmair, 1990), language modelling (Charniak, 2001), and question answering (Harabagiu et al., 2001). Recently, the statistical machine translation community has focused a lot of attention on the possible benefits of syntax-based machine translation (Yamada and Knight, 2001). Finally, as with tagging, parsers have been used in corpus annotation efforts in combination with human correction, for example in the creation of the Penn treebank (Marcus et al., 1993)

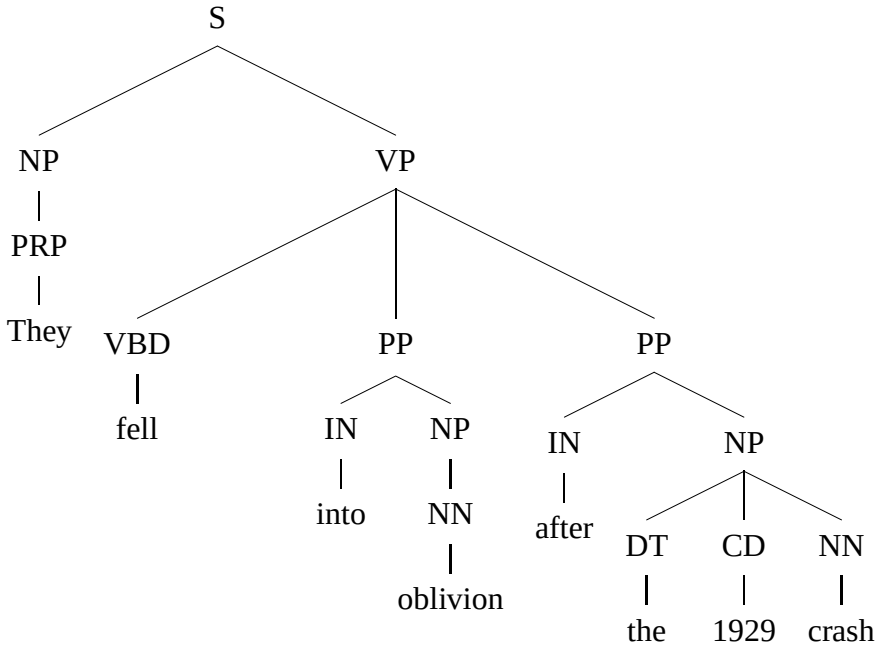

Figure 2.1: Syntactic analysis of an example sentence from WSJ

2.3.2 Approaches

There is a wide range of approaches to parsing. These can be broadly classified by the type of grammar formalism employed, and if the parser has a stochastic component.

Context-free grammars are formal production systems to describe recursive constituency in human languages (Chomsky, 1956). In a CFG, all rules have the form $X \rightarrow y$, where X is a non-terminal symbol and y is a sequence of non-terminals and/or terminals. Rules are said to be context-free because they can be applied regardless of the context of X. Despite linguistic arguments that at least some natural languages are mildly context-sensitive, such as Swiss-German (Shieber, 1985), CFGs are widely used for parsing. Feature-based or constraint-based grammar formalisms are popular from a linguistic perspective, for example Lexical Functional Grammar (LFG) (Bresnan, 1982), or Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1988). They allow fine-grained description of natural languages. Other approaches include categorial grammars and dependency grammars.

All of these approaches can be endowed with a stochastic component which assigns probabilities to every reading of a sentence. The main strength of such probabilistic parsers is their principled way of dealing with the ubiquitous ambiguity in natural languages by selecting the most probable parse. The simplest probabilistic version of a CFG is an unlexicalised probabilistic context-free grammar (PCFG). Charniak, 1996

reports work on estimating unlexicalised PCFGs using treebanks. Rules of the form LHS $\rightarrow$ RHS are read off directly from the parse trees of the Penn treebank. Rule probabilities $p(RHS|LHS)$ are estimated by smoothed relative frequencies according to the maximum likelihood principle. This simple scheme achieved surprisingly good results. More recently, it has been shown that such unlexicalised methods can achieve near state-of-the-art results (Klein and Manning, 2003). Also, feature-based approaches have been made probabilistic, in particular LFG (Johnson et al., 1999) and HPSG (Toutanova and Manning, 2002).

Treebank Grammars vs Manually Coded Grammars

PCFGs, either lexicalised or unlexicalised, are typically induced from an existing treebank where all local tree configurations in the gold-standard trees of a treebank are taken to be valid production rules. While such grammars which are derived from treebanks in this way are data-driven, it is important to bear in mind that the treebanks themselves were annotated with an implicit grammar formalism in mind as evidenced by comprehensive manuals in annotation projects such as the Penn treebank (Marcus et al., 1993). By contrast, grammars in constraint-based formalisms are explicitly specified, often with a focus on linguistically interesting problems, and not necessarily with a focus on frequent phenomena. But, large scale development of English formal grammars, aiming at coverage (Flickinger, 2000), and the development of treebanks for HPSG has helped to narrow the gap between these paradigms (Oepen et al., 2002).

Lexicalisation

Unlexicalised PCFGs can make overly strong independence assumptions with negative effects. For example, the probability of a verb phrase expanding into a verb and two following noun phrases is independent of the verb in an unlexicalised PCFG. This is clearly wrong since this probability should be considerably higher for ditransitive verbs such as ‘give’ or ‘tell’ than for other verbs (Manning and Schütze, 1999).

To remedy such problems, we decorate syntactic trees with lexical head information. For each local tree, we need to identify the *head*, in other words, the constituent which determines the syntactic character of the phrase. For example in the left tree in Figure 2.2, VB is the head of the VP, and VP the head of S. Such head assignments are not given in a treebank and need to be implemented as a set of head-finding rules (Jelinek et al., 1994; Magerman, 1995). Lexical information is projected upwards from

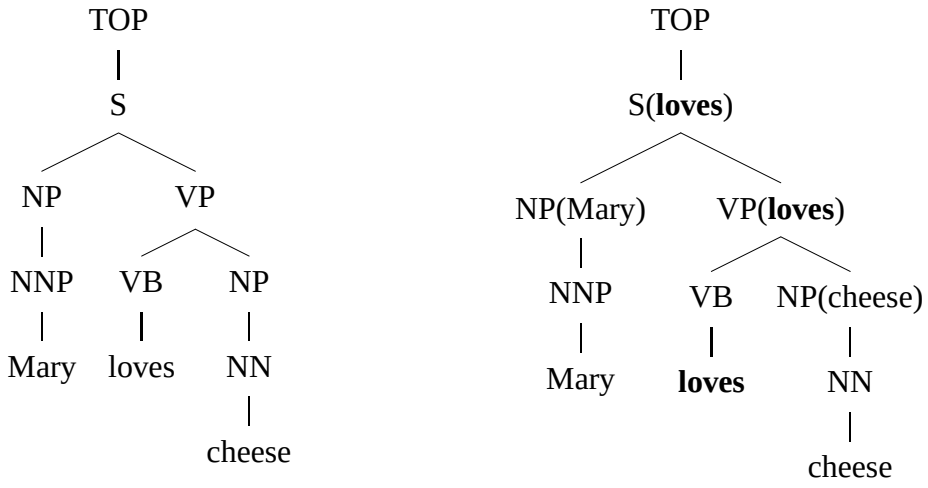

Figure 2.2: Head lexicalisation information percolates upwards from yield of tree.

the yield of a treebank tree along lexical head lines. This results in a decorated tree as in the right of Figure 2.2. From such a decorated tree we can read off rules like:

$$\mathrm{S(loves)} \rightarrow \mathrm{NP(Mary)VP(loves)}$$

In the general format which is the starting point for virtually all lexicalised PCFGs, non-terminal category X is decorated with additional lexical information x and we write $X(x)$ in the notation of (Collins, 1997) for each non-terminal in a rule:

$$P(h) \rightarrow L_m(l_m) \ldots L_1(l_1) H(h) R_1(r_1) \ldots R_n(r_n) \tag{2.19}$$

where P is the parent, H the head-daughter of the phrase, and L_i and R_i are left and right modifiers of the head-daughter. Head-word h is inherited from H to P.

Lexicalised PCFGs may differ from each other with respect to the kind of lexical information they include. Carroll and Rooth, 1998 only include words, while Collins, 1997 includes words and part-of-speech tags.

2.3.3 An Exemplary Parsing Approach – Collins, 1997

Reliably estimating the expansion probability of rules with a format as in Example 2.19 is impossible because of massive sparse data problems. In order to make parameter estimation robust, lexicalised parsing models must make further independence assumptions. Collins, 1997 introduces three lexicalised parsing models which successively

encode more and more linguistic knowledge. In the following, we discuss Model 2 in more detail which makes the following two crucial independence assumptions:

- Left daughters L_i, right daughters R_i, and head H are generated independently of each other.
- The generation of daughters on either side is independent of each other (0th order Markov assumption).

Thus, we arrive at the following formulation for rule expansion probabilities.

$$\begin{aligned} P(RHS|LHS) &= P(H|P) \\ &\quad \times \prod_{i=0}^{m} P(L_i|P,H,d(i)) \\ &\quad \times \prod_{i=0}^{n} P(R_i|P,H,d(i)) \end{aligned} \tag{2.20}$$

In order to compensate for these drastic independence assumptions, daughter generation is also conditionalised on distance measure $d(i)$ which models if there is an intervening verb or some form of punctuation between the ith constituent and head-daughter H. The linguistic modelling specific to Model 2 concerns subcategorisation information. Here, an incremental subcat frame feature maintains arguments of H which have yet to be generated on that side.

Despite these independence assumptions, all parameter types condition on large contexts such that direct parameter estimation would suffer from sparse-data problems in many situations. To counter this problem, a number of back-off levels are foreseen for each parameter type. For example, the head daughter expansion probability $P(H|P,w,t)$ smoothes with $P(H|P,t)$ and $P(H|P)$. As with TnT, these parameters are combined via linear interpolation.

The Model 2 parser achieves 88.6% precision and 88.1% recall on Section 23 of the Penn treebank (for sentences shorter than 40 words). The results published (Collins, 1997) were a leap forward over other models at the time and the paper was hugely influential. In fact, Dan Bikel devoted considerable effort to the reimplementation of Model 2 (Bikel, 2004a). Bikel showed that a clean-room implementation based entirely on information from (Collins, 1997) and Collins' PhD thesis (Collins, 1999) does not match the reported performance. In order to bridge this gap, he provided a thorough analysis of all missing details to reach the published results. Importantly, he finds

in his analysis that bilexical dependencies are much less important than previously thought. We use Bikel's implementation for our experiments which can be downloaded at `http://www.cis.upenn.edu/~dbikel/software.html`.

2.3.4 Annotated Resources

The supervised training of probabilistic parsers requires syntactically annotated corpora, so-called treebanks. A number of treebanks has been released over the last decade for a variety of languages, including English, German, Chinese, Korean, etc.

English – The Penn Treebank

Perhaps the most prominent example is the University of Pennsylvania treebank, or Penn treebank (Marcus et al., 1993; Taylor et al., 2003). In a period of eight years, from 1989 until 1996, American English texts from a wide range of genres were annotated at different levels of linguistic analysis. Genres included computer manuals, Wall Street Journal (WSJ) articles and transcribed telephone conversations. About 7 million words were part-of-speech tagged; 1.6 million words annotated for speech disfluencies. 3 million words were skeletally parsed, that is to say, annotated with context-free bracketing with limited empty categories and no indication of non-contiguous structures and dependencies. Over 2 million words were parsed for predicate-argument structure, Figure 2.3 shows an example annotation for a sentence from the WSJ section.

Marcus et al., 1993 explicitly state that a primary motivation for the enormous endeavour of creating the Penn treebank was to support "the automatic construction of statistical models for the grammar of the written and the colloquial spoken language". Indeed, since its release it has spawned a considerable body of research into learning probabilistic parsers from annotated data.

German – NEGRA and TIGER Treebanks

The NEGRA project constructed the first large-scale corpus for German (Skut et al., 1997; Brants et al., 2003). The treebank consists of about 20,000 sentences (350,000 tokens) from articles from the Germain daily newspaper *Frankfurter Rundschau*. The NEGRA annotation scheme is more flexible than the Penn phrase-structure framework in order to accommodate for free word order phenomena in German. Grammatical

```
( (S
    (NP-SBJ (PRP He) )
    (VP (VBZ succeeds)
      (NP
        (NP (NNP Terrence) (NNP D.) (NNP Daniels) )
        (, ,)
        (NP
          (ADVP (RB formerly) )
          (DT a) (NNP W.R.) (NNP Grace) (NN vice) (NN chairman) )
        (, ,)
        (SBAR
          (WHNP-11 (WP who) )
          (S
            (NP-SBJ (-NONE- *T*-11) )
            (VP (VBD resigned) )))))
    (. .) ))
```

Figure 2.3: Syntactic analysis of an example sentence in bracketed format. Annotation includes part-of-speech (for example personal pronoun, PRP), syntactic category (noun phrase, NP), grammatical function (subject, SBJ), and coindexing. The wh-noun phrase dominating *who* is coindexed with the understood subject of the embedded sentence.

functions can be annotated explicitly by labelling branches. Furthermore, branches may cross, thus allowing for the convenient annotation of dislocated constituents.

The TIGER project extended NEGRA both in size and level of annotation (Brants et al., 2002). TIGER Release 2 comprises of 50,000 annotated sentences. The extended annotation scheme contains morphological information, a more detailed scheme for prepositional phrase functions, and secondary branches for the appropriate representation of coordinations.

Cost of annotation With automatic support from part-of-speech taggers and parsers, annotation speed for the syntactic annotation of the Penn treebank ranges between 750 and 1,000 words per hour for experienced annotators (Taylor et al., 2003). Brants et al., 2003 report approximately 1,300 words per hour for trained annotators.[3] Given the size

[3]The higher performance for NEGRA annotation might be explained by the use of part-of-speech tagger and chunker trained on previously labelled material from the same domain.

of the corpus one can imagine the extent of human effort invested into this project. A further complication in the construction of high-quality annotated corpora is the need to guard against human error. In annotating the NEGRA corpus, each sentence is independently labelled by two annotators (Brants et al., 2003). Diverging labels are detected automatically and submitted to the annotators for comparison. Reportedly, comparison requires much more time than the initial annotation. Brants, 2000a reports, among other things, on annotator errors in syntactic annotation of the NEGRA corpus. Two annotators achieve labelled f-scores against the final version of 94.9% and 95.1%, disregarding the annotation of grammatical functions. In general, manual syntactic annotation is a very costly enterprise. Double annotation and subsequent comparison for higher quality standards typically aggravate the cost problem.

2.3.5 Summary

We have provided an overview over approaches to parsing with a strong emphasis on lexicalised probabilistic models. In particular, we outlined Collins' Model 2 parser which we use – in Bikel's implementation – for our active learning experiments for parsing in Chapter 6. We also gave an overview over treebanks as an indispensable factor in the training of probabilistic parsing models and discussed the expense necessary for their creation. The application of active learning to parsing should help to incur savings in the annotation of treebanks for new languages.

2.4 Active Learning

The preceding sections made clear the necessity of annotated data for the supervised training of classifier. Standardly, the training of supervised classifiers assumes randomly sampled training data. This risks the inclusion of redundant or irrelevant data points, thereby wasting human annotation effort. We have motivated the use of active learning in the Introduction Chapter as a means to reduce the annotation effort over random sampling. In contrast to the random sampling of data points, active learning can select data points for annotation by directing human annotation effort towards useful data points.

Dasgupta, 2004 gives the important theoretical result that even optimal active learning methods are not always guaranteed to perform better than random sampling. In fact, it is easy to construct situations where this would be the case. In a domain where

all data points have disjoint feature bundles, selecting some data points according to a selection function is as good as selecting data points randomly.

In practice, of course, active learning is found to incur substantial savings in annotation effort over random sampling. We will look at a variety of different approaches to active learning for the remainder of this chapter. Uncertainty sampling and Query-by-Committee are very commonly used methods in the field and we use these as starting points in all our experiments. Correspondingly, we pay particular attention to the description of these two types of methods. Other methods we consider are statistically optimal solutions, error reduction sampling, density estimation, kernel-based methods and hybrid methods.

2.4.1 Uncertainty Sampling

Uncertainty Sampling estimates the usefulness of a data point according to the uncertainty of the model about the correct label of that data point. In a binary classification scenario, uncertainty is highest when probabilities are at the decision boundary of 0.5 (Lewis and Gale, 1994; Lewis and Catlett, 1994). Accordingly, examples with class probabilities near the decision boundary are preferably selected.

For multinomial distributions, we can quantify uncertainty using *entropy*. The entropy of a discrete random variable H is defined as follows:

$$H(X) = -\sum_{x \in X} p(x) \log p(x) \tag{2.21}$$

Application to Parsing

Uncertainty sampling has been successfully used for active learning of a probabilistic parser (Hwa,). Uncertainty of the parser about its analysis of a given sentence is expressed in terms of *tree entropy*, the entropy of the output distribution over all possible analyses. Hwa gives a dynamic programming algorithm to efficiently compute entropy over the exponential number of parse trees. Tree entropy as a measure of uncertainty has also been demonstrated to work well for different types of parsers (Hwa, 2001b).

Does Entropy Quantify Parameter Unreliability?

In this thesis, we posit that active learning methods should address unreliable parameters. This raises the question if uncertainty sampling does in fact do that. Or in other

words, is entropy a reasonable quantity to express parameter unreliability? We can answer this question with two illustrative examples from the domain of PP-attachment.

First, imagine we have a single empirical count for a feature f in favour of being noun-attached. Using maximum likelihood estimation without smoothing (as in (Collins and Brooks, 1995)), we arrive at the following probability estimate:

$$P(\mathbf{n}|u) = \frac{C_{\mathbf{n}}(f) = 1}{C(f) = 1} = 1.0 \tag{2.22}$$

For such a peaked distribution, entropy is 0.0, meaning that there is no uncertainty. However, this is clearly a case of an extremely unreliable parameter estimate. For example, if we were to acquire one more instance which is in favour of verb attachment, the estimate would undergo a major shift from 1.0 to 0.5. By contrast, imagine a parameter with a large number of counts equally distributed between verb and noun attachment.

$$P(\mathbf{n}|u) = \frac{C_{\mathbf{n}}(f) = 500}{C(f) = 1000} = 0.5 \tag{2.23}$$

This is a situation of a highly reliable probability estimate for $P(\mathbf{n}|u)$ as the acquisition of another data point would do very little to change this estimate in either direction. Clearly, spending more annotation effort on such a parameter estimate would be wasteful. However, the probability distribution has high entropy, and uncertainty sampling would assess this as a worthwhile target for annotation.

The examples demonstrate that uncertainty and unreliability are not equivalent. As seen above in Equation 2.21, entropy is a function over a probability distribution only, and we do not consider information about the empirical counts from which the distribution was estimated. We will see in later experiments, in particular for PP-attachment, how this can cause problems for uncertainty sampling.[4]

2.4.2 Query-by-Committee

Query-by-Committee is an active learning method which requests annotation of data points for which a committee of models disagrees the most (Seung et al., 1992). It can be shown analytically that the generalisation error of the algorithm decreases exponentially with the number of examples. Muslea, 2002 provides an intuitive explanation for QBC: Whenever an ensemble disagrees over the labelling of a data point, at least

[4] These problems have been acknowledged already in Lewis and Gale, 1994; Lewis and Catlett, 1994.

one ensemble member must have predicted a wrong label, and learning the true label will benefit at least the member that made the mistake. Seung et al., 1992 suggested QBC for a perceptron-learning task and provided theoretical foundations for the algorithm. Freund et al., 1997 established theoretically that the approach is valid for a larger class of models. QBC has since been applied to a variety of applications, including document classification (McCallum and Nigam, 1998), part-of-speech tagging (Argamon-Engelson and Dagan, 1999), and parsing (Baldridge and Osborne, 2003).

Ensemble Creation

QBC requires a method to create diverse ensembles. In *Co-Testing*, diverse ensemble members are based on multiple views (Muslea, 2002). For example, web page classification can be factored into views based either on the words on a web page or based on the words in the hyperlinks pointing towards that page (Blum and Mitchell, 1998). As in other QBC approaches, Co-Testing selects examples for annotation where the ensemble classifiers disagree. Similarly, feature-based approaches allow to create multiple views by creating suitable feature splits (Jones et al., 2003; Osborne and Baldridge, 2004; Becker et al., 2005). However, not all problems have a way of being factored into alternative views of the learning problem. Furthermore, this method requires manual intervention to create a feature split. For these reasons, we consider only automatic ways of inducing ensemble diversity in this thesis. Two popular methods to do this are *bagging* and *sampling from a Dirichlet distribution*.

Bagging Bagging (Breiman, 1996) and boosting (Schapire, 1990) are well-known ensemble methods in machine learning and have been applied in the context of active learning as *Query-by-Bagging* and *Query-by-Boosting* (Abe and Mamitsuka, 1998). Both methods achieve significant (and similar) improvements over random sampling for a variety of machine learning tasks. Because of their similar performance, we restrict our studies of such methods in later experiments to bagging. Bagging is a general machine learning technique that reduces the variance of the underlying training methods (Breiman, 1996). It aggregates the parameter estimates from classifiers trained on bootstrap replicates (bags) of the original training data. Creating a bootstrap replicate entails sampling with replacement n examples from a training set of n examples. Training a classifier on such a perturbed training set results in fluctuating model parameters across bags. What is important for the application in QBC is that this variance is higher for parameters based on infrequent events.

Sampling from a Dirichlet Distribution As an alternative to perturbing the entire training set, one can sample the individual distributions of the model according to their posterior distributions (McCallum and Nigam, 1998; Argamon-Engelson and Dagan, 1999). The probability distributions which we perturb in this way are multinomial in general.[5] The parameters of multinomial distributions are described by the Dirichlet distribution. The probability density function of the Dirichlet distribution is parametrised by the empirical counts in the training set. As in bagging, lower counts suffer from higher variance which contribute to higher ensemble divergence.

Discussion Since both bagging and Dirichlet sampling are capable of creating diverse ensembles, and both methods create higher variance for lower counts, it may seem as if the methods are equivalent. However, there is one crucial difference.

Dirichlet sampling, as applied in (McCallum and Nigam, 1998; Argamon-Engelson and Dagan, 1999), samples from each probability distributions which is present in the model individually. As a consequence, the total number of distributions (and hence the model structure) remains unchanged. By contrast, Query-by-Bagging (Abe and Mamitsuka, 1998) eliminates some of the low frequency event types and thus has the potential to change the model structure. We will consider both methods in all our QBC experiments. To the best of our knowledge, no one has compared the effects of using these alternative perturbation methods previously. In particular, the potential of eliminating events through bagging will be put to use in a novel method for parsing which we describe in Section 6.4.

Divergence Metrics

Once the ensemble classifiers have been applied to the unlabelled instances in the pool, the divergence within the ensemble can be quantified in a number of ways. The two most popular methods are *vote entropy* and *Jensen-Shannon divergence*.

Vote Entropy Vote entropy is defined as the entropy of the distribution which results from each classifier in the ensemble voting for its top-ranked label (Argamon-Engelson and Dagan, 1999).

$$D(e) = -\sum_{c} \frac{V(c,e)}{k} \log \frac{V(c,e)}{k} \tag{2.24}$$

[5] The binomial distribution, appropriate for binary classification tasks such as PPA, can be adequately treated as a special case of the multinomial.

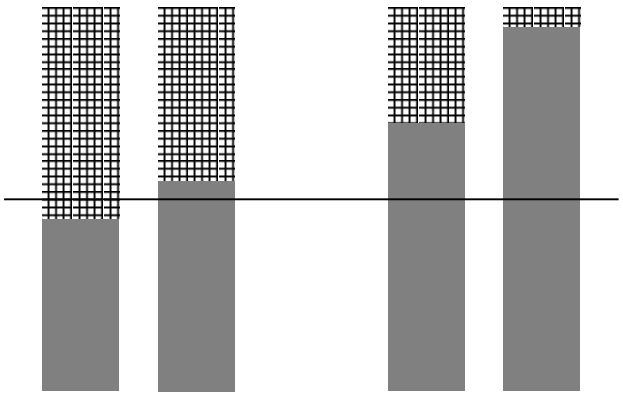

Figure 2.4: Output distributions of two binary classifiers for two examples

where k is the size of the committee and $V(c,e)$ is the number of committee members assigning a class c for the example e. One can further normalise this score by taking the logarithm of the number of possible labels so that the score ranges from 0 to 1.[6]

Jensen-Shannon Divergence The Jensen-Shannon (JS) divergence is a measure for the distance between a set of distributions $\{p_1, p_2, \ldots, p_n\}$ (Lin, 1991). It is defined as:

$$JS_\pi(\{p_1, p_2, \ldots, p_n\}) = H\Big(\sum_{i=1}^{n} \pi_i p_i\Big) - \sum_{i=1}^{n} \pi_i H(p_i) \tag{2.25}$$

where $H(\cdot)$ is entropy as defined above in Equation 2.21 and π is a weight vector such that $\sum_i \pi_i = 1$. JS-divergence can be seen as an extension of the well-known KL-divergence towards multiple distributions (Kullback and Leibler, 1951).

$$KL(p,q) = \sum_{x \in X} p(x) \frac{p(x)}{q(x)} \tag{2.26}$$

For a uniform weight vector π, it can be shown that the JS-divergence is the average KL-divergence to the mean of the set of distributions (Dhillon et al., 2002). This metric has been previously applied as divergence metric in active learning (McCallum and Nigam, 1998; Melville and Mooney, 2004).[7]

[6] In fact, Argamon-Engelson and Dagan, 1999 normalise by committee size and the number of possible labels. Since these are constant for one experiment, normalising in this way does not alter the ranking of scores.

[7] McCallum and Nigam used the term *KL-divergence to the mean*.

Discussion McCallum and Nigam, 1998 found empirically that QBC applied to the problem of text classification achieved better results when using JS-divergence than when using vote entropy. They attributed it to the fact that JS-divergence takes into account the classifiers' confidence by considering their entire distribution, in contrast to vote entropy. On the other hand, it can be argued that ensemble divergence is more critical when it occurs near decision boundaries. Consider the output distributions of an ensemble of two binary classifiers for the two examples in Figure 2.4. The ensemble exhibits more uncertainty around the decision boundary for example A, and vote entropy would yield a higher score for example A. This may be beneficial since example A is potentially a more informative data point to label than example B. By contrast, JS-divergence would preferably select example B because it shows a higher divergence in its output distributions. This shows that the advantage of using one divergence metric over the other cannot be derived from first principles, and we will therefore consider both metrics in all our QBC experiments.

2.4.3 Other Active Learning Methods

We are going to describe a range of other possible active learning methods which we do not consider in this thesis. Some of these methods are computationally to expensive to apply such as statistically optimal solutions or error-reduction sampling. Other methods such as density estimation and online choice algorithms are orthogonal to our research and could be combined in principle.

Statistically Optimal Solutions

There are approaches to active learning which have statistically optimal solutions, in other words, on average they reach the highest possible error reduction. Cohn et al., 1996 suggest artificially constructing a query such that the expected variance is minimised after knowing the label. This approach takes advantage of the *bias and variance decomposition analysis* (Geman et al., 1992). This analysis states that the expected error in supervised learning can be derived from first principles as the sum of the intrinsic target noise, bias, and variance. The intrinsic target noise can be understood as the error of the Bayes-optimal classifier. Bias expresses how closely the average guess of a classifier matches the target. Variance measures how much the classifier "moves" around. According to this analysis, minimising variance also minimises the expected error as long as the learner is unbiased or the bias is constant. Of course, guaranteeing

unbiased or constant-bias classifiers may not always be possible. Also, it is difficult in general to compute the variance of a classifier in closed form. Furthermore, the construction of artificial examples for manual labelling may be either not feasible or would result in unnatural examples for most applications in NLP. For these reasons, most research in active learning has focused on approximate approaches which optimise indirect measures other than variance in order to minimise the classification error.

Error-Reduction Sampling

Roy and McCallum, 2001 suggest *error-reduction sampling* and report improvements over other approaches in a document classification task using naive Bayes. Error-reduction sampling successively considers each example in the pool and estimates by how much adding this example to the training set would reduce classification error on test data. An example is selected for annotation when it results in the lowest expected error as compared against all other examples in the pool. Of course, pool examples are unlabelled. Therefore, each example is added tentatively with all possible labels, and error reduction is averaged over the label distribution according to the current state of the classifier. This approach requires to retrain the training set very often: the number of pool examples multiplied by the number of labels. Unfortunately, this continuous retraining is prohibitively expensive for most applications. Roy and McCallum, 2001 acknowledge that this approach is only feasible for applications which are either very fast to train or allow incremental retraining.

Active Learning for Support Vector Machines

In the following, we will discuss a number of different approaches of applying active learning to support vector machines (SVM) or large margin classifiers. SVMs are a class of discriminative classifiers with strong theoretical foundations and very good generalisation performance (Vapnik, 1982). Data points in SVMs are represented as vectors in an n-dimensional space. Assuming linearly separable data, the decision boundary is a hyperplane of dimensionality $n-1$ which separates positively and negatively labelled data points in a binary classification setting.[8] Given a set of labelled data, training a SVM means finding the unique separating hyperplane which maximises the *margin*, that is the minimal distance between data points and the decision boundary.

[8] Non-linearly separable cases can be addressed by using slack variables (Cortes and Vapnik, 1995).

Campbell et al., 2000 *Support vectors* are data points which fall exactly on the margin. A characteristic of SVMs is that they can be constructed by using support vectors only. In other words, in training a SVM one can safely ignore non-support vectors and still arrive at the same hypothesis. This fact motivates the application of active learning in (Campbell et al., 2000). If one knew a priori which data points are support vectors, one could focus on labelling only these. Campbell et al., 2000 suggest a heuristic for active learning which selects data points closest to the decision boundary. Empirically, they found best results for active learning with *sparse* data sets. These are data sets which require only few support vectors (in relation to the size of the full training set). By contrast, *dense* data sets which require a relative large number of support vectors to accurately represent the hypothesis cannot achieve the same result with less data.

Schohn and Cohn, 2000 A range of heuristics is discussed in (Schohn and Cohn, 2000). They present empirical results for a heuristic which selects examples which are close to the decision boundary, as in (Campbell et al., 2000). The motivation is to maximally narrow the margin. Schohn and Cohn, 2000 discuss cases where selecting examples from sparse regions may result in a displacement of the hyperplane without significant change in the margin. However, empirically they found substantial improvements over random sampling for a variety of text classification tasks.

They also discuss a heuristic which prefers examples which are orthogonal to the space spanned by the current training set, thus giving the learner information about unobserved dimensions. Aiming for such unobserved events would be similar in spirit to our main thesis of addressing unreliable parameters in the form of unknown events. However, they do not consider this heuristic in their experiments, even though this would be computationally inexpensive. Also, we are not aware of any other research in the area of large margin classifiers which target unobserved dimensions.

Tong and Koller, 2001 Tong and Koller, 2001 use the notion of *version space* as a theoretical motivation for their approach to active learning for SVMs. Version space is the size of the subset of parameter space that correctly classifies the labelled examples (Mitchell, 1982). An optimal active learning method should try to reduce the size of the version space as quickly as possible.

Starting from the observation that an explicit computation of the version space size is not practical, they present different approximations to this problem. A first approximation follows the work discussed above (Campbell et al., 2000; Schohn and

Cohn, 2000), by selecting data points closest to the decision boundary since these would most accurately bisect the version space. Tong and Koller, 2001 argue that this is a rough approximation relying on the assumption that the data point is centrally placed. To overcome such problems, another approximation estimates the expected reduction in the version space space by computing the reduction in the size of the margin. This entails retraining the SVM for every unlabelled data point added to the training set with a stipulated positive and negative label. Empirically, they found best results with this latter method.

One can see a correspondence of this method with the idea of targeting unreliable parameters as proposed in this thesis. Large margins in an SVM are a source of unreliability, and minimising such regions contributes to better classification performance as evidenced by the good empirical results found by Tong and Koller, 2001.

Online Choice

It is a well-known fact that there is no single active learning method which is always guaranteed to be optimal across applications. This motivates the combination of active learning algorithms under one master learner. Baram et al., 2004 introduce an algorithm that, for a given iteration, selects the example which has been suggested by the best current learner. The key idea is to define a metric which evaluates learners. Of course, the true accuracy, as measured against a test set, is not available. Furthermore, evaluation over the labelled training set is not necessarily a good indicator of the true accuracy since active learning algorithms tend to collect hard, rather than representative examples. Instead, a *Classification Entropy Maximization* score is introduced to estimate learner quality. This is the binary entropy of the learner over an unlabelled set of data points. In an evaluation over different tasks, Baram et al., 2004 report that their approach consistently performs nearly as well or better as the best algorithm in the ensemble.

Density Estimation

Active learning can have a tendency to select atypical examples or even *outliers*. An outlier is an observation that lies outside the overall pattern of a distribution (Moore and McCabe, 1999). For instance, a French sentence in an English corpus would constitute an outlier. A classifier trained on English sentences would be highly uncertain as to the proper analysis of such a French sentence, so that a method like uncertainty

sampling would preferably select it for annotation. *Density estimation* can help active learning to avoid such atypical examples. In the context of text classification, McCallum and Nigam, 1998 approximate the density of a document by measuring its average KL-divergence to all other documents. This method essentially quantifies the overlap between a document and all other documents. Examples are selected which have both high density and high disagreement according to QBC.

In application to parsing, Tang et al., 2002 estimate density using k-means clustering. Pointwise distance between two sentences is computed as the edit distance between the derivations of the individual best parse trees of two sentences under the current parameter model. Similar to (McCallum and Nigam, 1998), sentences are selected for annotation which have high density and high uncertainty. This is an computationally intense process since all pairwise edit distances between all sentences in the pool have to be computed in every iteration.

As we will see in later chapters, active learning methods, especially uncertainty sampling, can suffer severe performance problems from selecting atypical examples and density estimation may well help to alleviate such problems. In general, we see density estimation as a general purpose technique which is orthogonal to the choice of active learning method. Of course, this raises the question if we should continue using such methods if they require such a fix. However, the application of density estimation to active learning is a matter of ongoing research. For instance, it is not clear if density estimation is best applied over unlabelled instances, as in (McCallum and Nigam, 1998) or tentatively labelled instances, according to the current state of the classifier, as in (Tang et al., 2002). For this reason, we do not consider density estimation in the experiments of this thesis.

2.5 Conclusion

In this chapter, we described the natural language processing tasks to which we apply active learning methods in Chapters 4, 5, and 6 and gave a more detailed overview of one representative model for each of the tasks. Furthermore, we discuss relevant applications and available data sets. Next, we described active learning with a special focus on uncertainty sampling and QBC. We also described other popular methods in the field which we do not consider in our research.

Chapter 3

Experimental Conditions

This chapter details the experimental conditions which are common to all experiments in this thesis. More details will be discussed later in place with the descriptions of the individual experiments.

In Section 3.1, we outline the general data split scheme for data appropriate for active learning and discuss details of the 10-fold cross-validation scheme we employ throughout all experiments. In Section 3.2, we discuss evaluation metrics, such as accuracy, coverage and f-measure, which are appropriate to the diverse tasks in this thesis. In Section 3.3, we explain and motivate the choice of statistical significance test which we apply throughout all experiments. In Section 3.4, we motivate the choice of (graphical) learning curves for the display of performance results in active learning, plotted alongside p-levels to indicate significance of results. Section 3.6 concludes this chapter.

3.1 Data Splits and 10-Fold Cross-Validation

For any active learning experiment, we need to decide on experimental parameters regarding the sizes of the following sets:[1] i) the initial *training set L* of manually labelled examples, ii) the *test set* which consists of manually labelled examples which provide the gold standard for evaluation, iii) the *pool U* of unlabelled candidates and iv) the *batch* of examples which is selected from the pool in one iteration.

For the experiments in the following chapters, we set these as summarised in Table 3.1. We consider the size of the (initial) training set and the batch to be particularly interesting parameters, and demonstrate their effects on the relative performance of

[1] The variable names refer to the algorithm in Figure 1 in the first chapter.

	PPA	Sequencing	Parsing
Training set	1/100/1000	100/1000	100/1000
Test set	2000	1000	1000
Pool	all	1000	1000
Batch	1/100	100	100

Table 3.1: Data splits for experiments in prepositional phrase attachment, sequencing, and parsing tasks

different active learning methods. We include experiments of varying training set sizes for all applications. For pragmatic reasons, we deal with varying batch sizes only in the context of the very fast prepositional phrase attachment (PPA) tasks since smaller batch sizes entail a larger number of iterations and correspondingly longer experimental running times.

The test set and pool size were set as large as conveniently possible. Again, for the fast PPA application, we can afford to have a very large pool comprising all instances which are neither in the training nor in the test set. As sequencing and parsing are considerably slower, we use a smaller pool of 1000 instances which are randomly sampled in each iteration from all remaining instances (not in training or test set), which we will refer to as the *global pool*.

For all experiments, we carry out a 10-fold cross-validation for improved statistical significance of the results. For each fold, we randomly sample (without replacement) an initial training set, a test set and a pool according to the specifications in Table 3.1, while ensuring that test sets are disjoint across folds. We run each fold until completion, that is, until the pool is exhausted for PPA, or for a fixed number of rounds for all other tasks since running until completion can be too time-consuming. Finally, we report results averaged across all 10 folds.

Specifics of Parsing Experiments

We can achieve better parsing results by providing the parser with part-of-speech tagged input. For realistic parsing experiments, we tag the test set automatically (as opposed to using gold-standard tags), using the TnT part-of-speech tagger (Brants, 2000b), trained on 30,000 sentences from the global pool. For methodological reasons, it would be desirable to automatically tag the sentences in the global pool, too. However, there are not enough sentences in the treebank to allow a further split into

a disjoint training set for the tagger and the pool in order to avoid application on the training set. For this reason, we do not use automatically tagged sentences when parsing the pool, but manually tagged sentences.

3.2 Evaluation Metrics

In order to compare methods against each other and to trace learning progress for individual methods, we use a variety of evaluation metrics depending on the task at hand.

Accuracy is the proportion of correctly labelled instances as compared to the gold standard. We use accuracy to evaluate prepositional phrase attachment and part-of-speech tagging, that is, the proportion of correctly labelled tuples in prepositional phrase attachment and the proportion of correctly labelled word tokens in tagging. A related metric in the context of parsing is *exact match rate*, the proportion of entire sentences which are parsed perfectly. This is a very harsh metric, because no credit is given if only a single constituent has been misclassified.

Accordingly, for structured labelling tasks such as parsing and NER, a more appropriate evaluation metric, PARSEVAL, measures *precision* and *recall*, based on constituents (for parsing) or on entities (for NER) (Black et al., 1991).

$$\text{(Labelled) Precision} = \frac{\text{\#identical (labelled) constituents in parse and correct tree}}{\text{\#constituents in parse tree}}$$

$$\text{(Labelled) Recall} = \frac{\text{\#identical (labelled) constituents in parse and correct tree}}{\text{\#constituents in correct tree}}$$

In order to determine if a predicted constituent (or entity) is identical to the one in the gold standard, we require that they at least have the same span, in other words, cover the same input tokens. Additionally, it is customary to require that they have the same label. We refer to these measures as *labelled* precision/recall. In this thesis, we follow the usual practice in the literature and exclusively report labelled precision and recall.

Because it is possible to increase precision at the cost of recall, and vice versa, a common summary statistic is the *F-measure*, the harmonic mean of precision and recall.

$$\text{F-Measure} = \frac{2PR}{P+R}$$

Furthermore, we report the *coverage* score, a commonly used metric for parsing. It is the proportion of parsable sentences in a test set, that is, those sentences which receive at least one analysis. We will also report coverage in the context of prepositional phrase attachment, as the proportion of test instances with known prepositions. Coverage can be conveniently evaluated on unannotated corpora. This metric does not tell us about the quality of the predicted analyses, though. Also note, that as a design decision, a parser may or may not recourse to more aggressive backoff or smoothing schemes in order to increase coverage, possibly at the expense of parsing quality. However, we find coverage a useful notion because high coverage is a necessary precondition for good performance in general.

3.3 Statistical Significance of Results

In order to establish that the performance difference between two methods is in fact significant, and has not just arisen by chance, we need to apply statistical significance testing.[2] Statistical significance is reported with respect to a single *test statistic* of interest such as classification accuracy. The usual methodology is to state a null hypothesis of the form "Method A and method B do not differ with respect to the test statistic". Then, we can determine the probability that an observed difference in the test statistic of that magnitude has arisen by chance given the null hypothesis. This probability is known as the *p-value*. If the p-value is lower than a predefined *significance level* we can reject the null hypothesis. Commonly used significance tests are $p < 0.01$ and $p < 0.05$.

We use randomisation tests for significance testing (Noreen, 1989). Randomisation tests are a class of computer-intensive statistical methods which can compute p-values for more complex test statistics such as the f-measure where analytical methods fail. They also free us from "making troubling assumptions about sampling models and population distribution" (Cohen, 1995). Randomisation tests automatically generate sample distributions by randomly reshuffling observed data points between experimental conditions. For small enough test sets, one can enumerate all possible outcomes of this procedure and compute an *exact randomisation*. For many practical purposes, this is not possible and we must resort to *approximate randomisation* where the collection of test statistics is based on a large enough number of reshuffles.

In particular, we use an (approximate) randomisation version of the paired t-test,

[2] The description in this paragraph is based on (Chinchor et al., 1993).

APPROXIMATE RANDOMISATION
1. Collect difference in test statistic for methods A and B $\lvert stat_A - stat_B \rvert$
2. Shuffle *ns* times (*ns* is 9,999 in our case)
3. Count the number of times (number greater than or equal, *nge*) that $\lvert stat_{PseudoA} - stat_{PseudoB} \rvert \geq \lvert stat_A - stat_B \rvert$ (stat can be accuracy or F-measure)
4. The estimate of the p-value is $\frac{nge+1}{ns+1}$ (1 is added to achieve an unbiased estimate)

Table 3.2: Implementation of randomised paired-samples *t* test

motivated by the fact that, by design, we compare two methods at a time on exactly the same sequence of test items. Table 3.2 gives an outline of the implementation of this randomisation test, based on (Chinchor, 1992). Initially, we determine the absolute difference between test statistics over the original outcomes of methods A and B. Then, we repeatedly create *shuffled* versions of A and B, determine the absolute difference between their test statistics and count the number of times that this perturbed difference is equal or larger than the original difference. In order to create the shuffled versions of the data sets, we iterate over each data point and decide based on the outcome of a (simulated) coin-flip whether records should be exchanged between A and B. The p-value is the proportion of iterations in which the absolute difference in test statistics was indeed larger for the shuffled version (corrected to achieve an unbiased estimate).

Such randomised tests for significance results are common in natural language processing, for instance in the evaluation of information extraction systems (Chinchor et al., 1993) and parsers (Bikel, 2004b). For systematic purposes, we decided to use randomisation tests to compute significance results for all relevant test statistics reported in this thesis, including accuracy and f-measure, following the advice from Noreen, 1989 to "use them instead of ordinary t-tests because they free us from worrying about parametric assumptions and they are no less powerful."

3.4 Comparing Active Learning Results

Reporting and comparing results for active learning methods can be problematic. Sometimes, active learning performances are summarily reported using a single character-

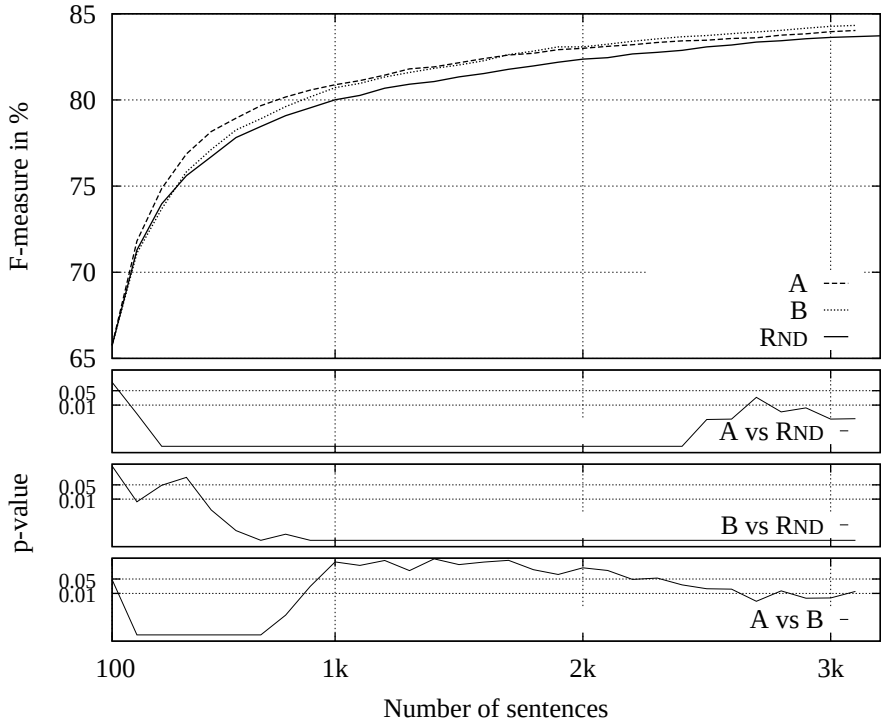

Figure 3.1: Graphical display of learning curves with associated p-level graphs

istic. For example, savings in training effort are reported for a fixed arbitrary performance level, which may be difficult to decide beforehand. Similarly, when reporting the increase in performance for a fixed arbitrary training set size, we need to decide on a reasonable size beforehand. Furthermore, such pointwise characteristics do not reveal trends in active learning performance.

As an alternative, one may decide to report the average performance over all iterations. This can be misleading if we are interested in the performance at the limit. Let us consider the learning curves for two active learning methods A and B in comparison to random sampling in the top panel in Figure 3.1.[3] While method A in this example has a higher average f-measure across all iterations (80.9%) than method B (80.8%), we would still prefer method B when dealing with larger training set sizes.

Since active learning performance really is a function of a growing training set size, we graphically display active learning results in the form of learning curves (Figure 3.1), as is common in the literature. This practice allows for convenient eye-balling of results and for comparing trends in learning curves. Looking at the graphs in the example, we can see that method A initially performs best, but is eventually overtaken

[3]This graph anticipates results from Chapter 6 on parsing.

by method B. Furthermore, we can glean from the graph the range within which the methods' performances are roughly equivalent.

To indicate the significance of results, we introduce a novel graphical convention in this thesis. In the three bottom panes of Figure 3.1, we report p-values of a (randomised) paired t-test for all pairwise comparisons between results of the same training set size, on a scale from 0.001 to 0.5.[4] We mark two standard levels of significance, 0.01 and 0.05, as grid marks for reference.

We believe that representing p-values in this graphical way allows for an appropriate qualitative discussion of results while incorporating trend information with respect to the growth of the training set. For example, we can now state more precisely that method A is significantly better than method B from the first iteration until 900 sentences have been sampled ($p < 0.05$); and method B is significantly better than A after 2,400 sentences have been sampled until the end.

P-values can also help to judge improvements which may be difficult based on raw performance measures, especially when given graphically. For example, performance levels around 3,000 sentences seem very similar but p-values show that the differences are in fact significant.

Throughout this thesis, we will frequently compare multiple experiments on the same data set and report statistical significance results. Such multiple comparisons call for a downward correction of significance levels to avoid finding spurious 'significant' results, see for example (Shaffer, 1995) for a discussion. The chosen graphical representation allows us to apply more stringent significance levels if required.

3.5 Length-Balanced Sampling

When applying active learning to sentence-labelling tasks such as tagging or parsing, the sentences need a variable number of labelling decisions. This may confound active learning metrics and introduce a bias towards longer sentences. To ensure a balanced selection of examples, it is necessary to control for this factor. For example, tree entropy may be directly normalised by sentence length (Hwa,), or by the binary logarithm of the number of parser readings (Hwa, 2001a).

In practice, we found non-linear dependencies between sentence length and scores,

[4] After some experimentation, we decided to give p-values on a logarithmic scale since this resulted in smoother graphs. Also note that for the chosen number of 9,999 shuffles in the randomised test, 0.001 is the smallest possible p-value that can be computed. For any lower p-values the graph will be a flat line as we will commonly see in later result discussions.

such that most normalisation schemes introduce a selection bias either towards longer or shorter sentences. In order to control for this selection bias in active learning, we use the following method: Given a batch size b, we randomly sample b sentences from the pool and record the number e_l of selected examples for sentence length l. Then, for all lengths $l = 1, 2, \ldots 40$, we select from all sentences in the pool of length l the e_l examples with the highest score according to our sample selection metric. Of course, the union of these sets will have b examples again. Since we randomly sampled the batch from the pool, we may assume that the batch distribution reflects the pool distribution, particularly with respect to the distribution of sentence lengths.

By construction, this method effectively reproduces the sentence length profile of the original corpus and therefore guards against the selection of sentence length biased subsets. Furthermore, it is equally applicable for all metrics and allows a direct comparison between metrics. We apply this method to all active learning tasks where sequential data are involved, namely sequencing tasks and parsing since one may expect to find correlations between sample length and score.

3.6 Conclusion

In this chapter, we discussed the data split scheme we used for all experiments in this thesis. Furthermore, we introduced diverse evaluation metrics which are appropriate to the tasks in this thesis and motivated the choice of statistical significance test. Finally, we motivated the choice of (graphical) learning curves for displaying the performance results of active learning, plotted alongside p-levels to indicate significance of results.

With this chapter, we have provided a convenient overview of all experimental conditions which will allow a faithful reproduction of all our results.

Chapter 4

Unreliable Parameters in Prepositional Phrase Attachment

In prepositional phrase attachment (PPA), one decides whether a prepositional phrase is an argument or modifier of the verb in the matrix clause, or if it modifies the directly preceding noun phrase. This is an important step towards determining the argument structure of a sentence and its semantic interpretation. Previous work has shown that this can be done accurately based on lexical head information for the involved verb, noun, and prepositional phrases (Ratnaparkhi et al., 1994; Collins and Brooks, 1995). Standalone PPA has been obsoleted by the advent of state-of-the-art lexicalised parsers such as (Collins, 1997; Charniak, 2000), where PPA is addressed jointly with other lexical and structural disambiguation steps. Nevertheless, PPA is still attractive as a test bed for active learning research (Hwa, 2004). A quick turn-around in (otherwise highly time-consuming) active learning experiments is facilitated through the fast training and application of classifiers, particularly when using maximum likelihood estimation, as in (Collins and Brooks, 1995). Following Hwa, we also use the Collins and Brooks algorithm for the experiments in this chapter, as described in Section 2.1.

We argue in this thesis that popular active learning methods, such as uncertainty sampling and QBC, have been previously misapplied in the context of natural language processing tasks. These methods are defined in such a way as to improve the quality of parameter estimates within the current model structure. However, an important factor in learning accurate stochastic models is the acquisition of an appropriate model structure such that as many cases as possible are covered when predicting unseen test data. This is particularly important for natural language processing where probability distributions often need to be estimated for a potentially unbounded number of indi-

vidual words or combinations of words. Neither uncertainty sampling nor QBC have a well-defined mechanism to actively pursue expansion of the given model structure.

In the context of prepositional phrase attachment, unknown prepositions are a prime example of insufficient model structure due to missing events. In the absence of specific information for a preposition, we cannot expect the classifier to do well. However, assigning the most likely attachment for each preposition instead of assigning the majority label (noun-attached) results in an increase in accuracy from 59.0% to 72.2%. As mentioned in the Literature Review in Section 2.1, we can actually determine the most likely attachment relatively cheaply by labelling just a few instances for all prepositions. Accordingly, active learning methods should explicitly pursue unknown prepositions and thereby increase the coverage while simultaneously improving the parameter estimation within the current model structure.

Chapter Structure

We begin in Section 4.1 by demonstrating experimentally that a naïve application of uncertainty sampling to prepositional phrase attachment, without targeting coverage, can result in suboptimal performance. To address this problem, we introduce a change in the base classifier in order to preferentially select instances with unknown prepositions. This obviates the need to change the active learning method itself. We show that this method improves classification accuracy. Targeting unknown prepositions generally proves to be an effective way to improve unreliable parameters in the model. However, we find that uncertainty sampling can still underperform compared to random sampling, particularly when starting with small training sets. We give a detailed analysis as to why this is the case.

In Section 4.2, we show for QBC that a similar change in the base classifier can target out-of-coverage instances and substantially improve performance. We examine a range of sampling and scoring methods, and show that QBC can have vastly different experimental outcomes depending on the combination of methods used. We achieve best overall results with bagging for sampling and vote entropy for scoring. An analysis shows that the selection of sampling method allows one to bias the composition of the training set against prepositions that are easier to label.

4.1 Uncertainty Sampling for Prepositional Phrase Attachment

Uncertainty sampling is a popular and intuitive active learning method. However, it is essentially heuristic and may not accurately target unreliable model parameters if applied naïvely, as discussed in Subsection 2.4.1 in the Literature Review.

Unknown prepositions present an obvious and important case of unreliable parameters in prepositional phrase attachment. An important contribution of this section is to show that the targeted selection of unknown prepositions increases the classifier's coverage and thereby substantially improves classification accuracy. This preferential selection of unknown prepositions can be superimposed on top of standard uncertainty sampling.

Furthermore, we show that uncertainty sampling can exhibit degenerate behaviour and perform worse than random sampling even when it is well within the range of standardly used experimental parameter settings. In an experiment with a particularly small (randomly sampled) initial training set, we show that sample selection becomes stuck repetitively choosing instances with the same preposition. This happens even when targeting unknown parameters. In this situation, parameter estimates can be too coarse to support a meaningful selection process for uncertainty sampling.

Targeting Unknown Prepositions

The targeted selection of unknown prepositions can be done, entirely within the framework of uncertainty sampling, by applying a minimal, but crucial change to the base classifier. This change concerns the setting of the backoff probability for instances with unknown prepositions. In (Collins and Brooks, 1995), this value is defined as $P_5(\mathbf{n}|u) = 1.0$, such that instances with unknown prepositions are deterministically decided to be noun-attached, cf. Equation 2.2 in the Literature Review in Chapter 2. This is motivated by the fact, that the majority of instances in the given data set are noun-attached.

However, using this value in sample selection results in ignoring unknown prepositions since a probability of 1.0 indicates minimal uncertainty. To alter this behaviour, we choose a setting such that instances with unknown prepositions will be preferred. Of course, such a value is $P_5(\mathbf{n}|u) = 0.5$, indicating maximal uncertainty. In this way, we control the preferred or dispreferred selection of out-of-coverage instances by set-

ting the final backoff value without further changes to the active learning algorithm.

In the experiments of this section, we will demonstrate the benefits of applying such a targeted selection of out-of-coverage instances. We will use the following naming convention:

- UNC-STD: Uncertainty sampling, standard setting $P_5(\mathbf{n}|u) = 1.0$
- UNC-ALT: Uncertainty sampling, altered setting $P_5(\mathbf{n}|u) = 0.5$
- RND: Random sampling, base-line

4.1.1 Pure Uncertainty Sampling

Many active learning experiments reported in the literature use relatively large, randomly sampled initial training sets, for example 500 instances in (Hwa, 2004). However, there is no general way to determine the optimal size of this initial training set. In principle, one should trust an active learning method to be able to exclusively drive the selection process without any contribution of random sampling. In fact, very small training sets have been used (successfully) in the literature, for instance starting with a single example in (Osborne and Baldridge, 2004). In this first experiment, we also start with a single, randomly sampled instance to show the unadulterated effect of uncertainty sampling alone without the influence of random sampling.[1]

For this experiment, we select one instance from the pool per round. A batch size of one instance is considered optimal because, in theory, it should avoid the problem of selecting redundant examples. For subsequent experiments in this and later chapters, we will use larger batch sizes for efficiency reasons. This is also common practice in work reported in the literature (Hwa, 2004).

We show results in the form of accuracy learning curves for the two experimental conditions in comparison with random sampling in Figure 4.1. As expected, we find that the altered setting is consistently better than the standard setting. This improvement is significant until ca. 12.5k instances have been sampled.

To explain why the altered setting performs better, we look at coverage, the proportion of known prepositions in the test set, in Figure 4.2.

[1] One could in principle also start with an empty training set. This would have the same effect, though since, in the absence of any annotation, all instances in the pool would receive the same score, and the sampling of the first instance is effectively random.

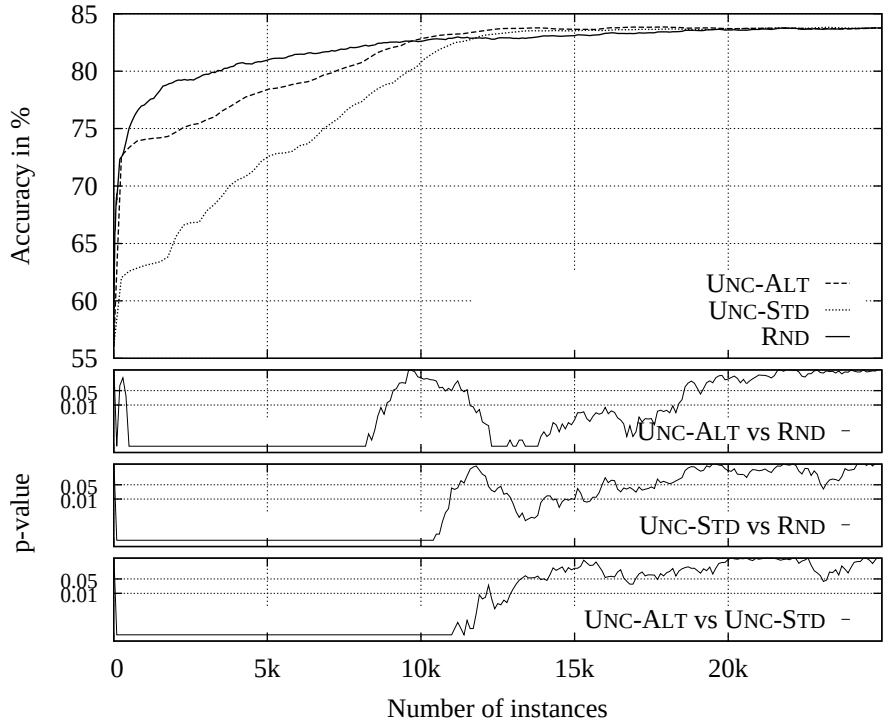

Figure 4.1: Backoff settings for uncertainty sampling (from one instance)

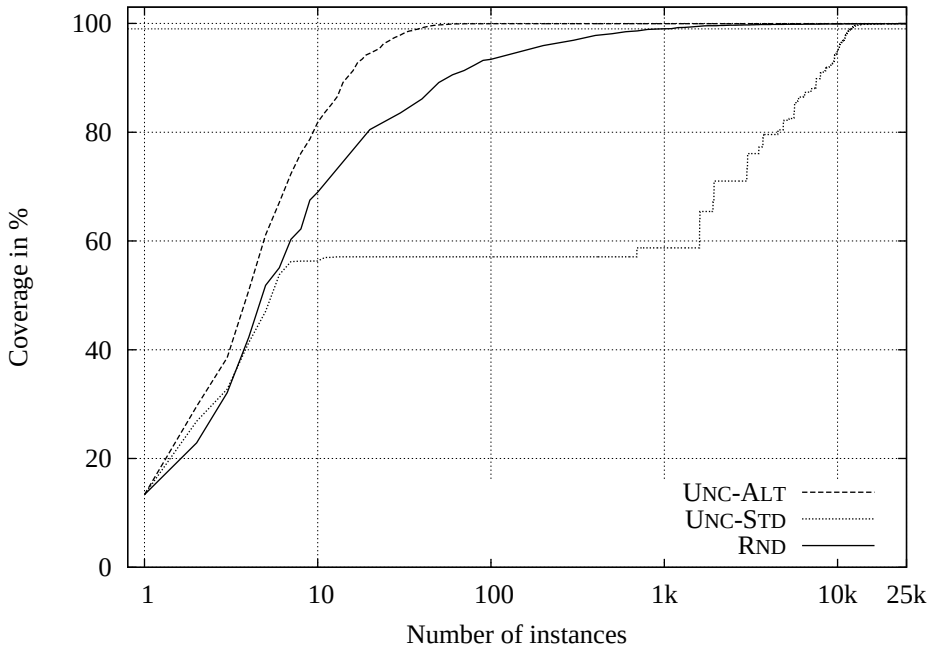

Figure 4.2: Backoff settings and impact on coverage for uncertainty sampling (from one instance); note that x-axis is on a logarithmic scale.

round	1	2	3	4 - 70	71	72	73	...	3041	...
preposition	*to*	*of*	*in*	*from, for, ...*	*of*	*in*	*in*	...	*in*	...

Table 4.1: Prepositions of selected instances in uncertainty sampling

In random sampling, coverage increases steadily and eventually converges towards 100%.[2] 99% coverage is reached at around 1000 instances.

The altered setting results in a considerably faster convergence than random sampling: 99% coverage is reached after 39 instances, and all prepositions in the pool have been selected at least once well before 100 instances have been selected. This is of course because, by construction, the altered setting probes all unknown preposition instances first: Initially, instances with known prepositions have empirical counts of (exactly) one. They assign a probability of 1.0 for being either noun- or verb-attached and thus flag minimal uncertainty. Those instances with unknown prepositions show maximal uncertainty at 0.5 and will be preferred on a one-on-one basis.

By contrast, coverage using uncertainty sampling with the standard setting markedly underperforms compared to both random sampling and the altered setting. Coverage exhibits a stepping behaviour with extremely long stretches, up to 1000 instances, where it is stalling. This stepping behaviour arises because, again by construction, this method cannot select instances with unknown prepositions while there are still known prepositions in the pool.

As a second important observation from Figure 4.1, we find that both uncertainty sampling conditions markedly underperform with respect to random sampling until late in the learning curve. The altered condition breaks even with random sampling after ca. 10.5k instances have been sampled, the standard one only after ca. 11.5k instances. This is in marked contrast to the results of (Hwa, 2004) where uncertainty sampling clearly outperforms random sampling from the beginning.[3]

Error Analysis

To explain this underperformance, we looked at the sequence of instances which where selected during a single run of uncertainty sampling with the beneficial altered condition. Table 4.1 schematically shows the main prepositions of these instances for the first part of the sequence.

[2] In fact, even with a maximal training set, coverage is a little under 100% since in some of the cross-validation folds low-frequency prepositions occur only in the test set but not in the training set.

[3] However, as we will see later, uncertainty sampling performs better at larger values of n and b.

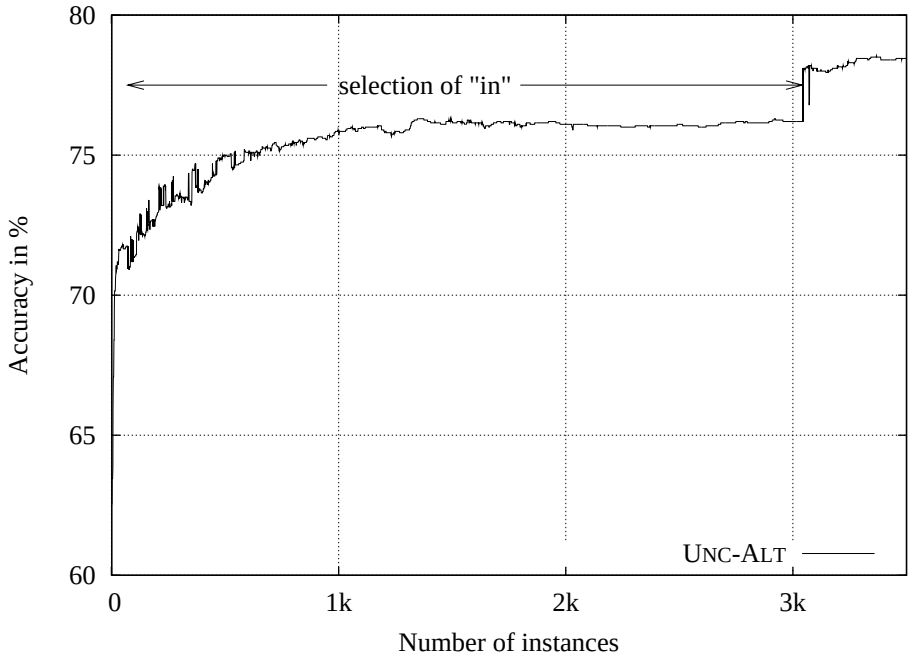

Figure 4.3: Accuracy for a single run of uncertainty sampling (from one instance)

After the first 70 rounds, each preposition in the pool has been selected exactly once. In this situation, predictions for all remaining instances will have no uncertainty since they are supported by a single empirical count in favour of being either noun-attached or verb-attached. Then, in round 72, a second instance of preposition *in* is randomly selected. This happens to be labelled differently from the instance sampled in the third round.[4] From this moment on, the selection algorithm is trapped into selecting further instances of preposition *in*, since their predictions have at least a minimal amount of uncertainty due to the two initial differing labellings. This continues for 2969 rounds until eventually all instances with this preposition have been labelled. Even worse, shortly after this the selection gets trapped into selecting instances of preposition *from* for the same reason.

This selection behaviour is reflected in the accuracy learning curve for this run in Figure 4.3. The early increase in accuracy is due to the comprehensive selection of different prepositions. From round 72 onwards, the further increase in accuracy is entirely due to the continued modelling of instances of preposition *in*. However, performance soon converges around 76%, with no further progress until round 3042,

[4]The first double labelling actually occurs in round 71. Since the second instance of preposition *on* happens to have the same label as the first no uncertainty is introduced.

when the onsetting selection of instances of preposition *from* causes a new boost.

We found this degenerate selection behaviour not only in this run, but across all runs of a 10-fold validation. The selection mechanism always got trapped early on into the selection of instances of a single kind of preposition for extended stretches. The main problem here can be pinpointed to the confusion of uncertainty and unreliability which is prevalent in uncertainty sampling. Prepositions with low empirical counts exhibit high certainty in their classification and get ignored; on the other hand, prepositions with high empirical counts show a higher degree of uncertainty and thus continue to receive more attention. Clearly, rather than directing annotation effort towards instances of a single type, prepositions with small empirical counts should receive more attention. These findings reveal serious problems of uncertainty sampling, at least in the pure form we used in this experiment.

A Discussion of Potential Remedies

These problems raise the question how uncertainty sampling can be fixed and we will now discuss a number of potential remedies.

By modelling the distribution of examples, *density estimation* can help to avoid problems of distorted selection in active learning (McCallum and Nigam, 1998; Tang et al., 2002). Here, density estimation is applied to avoid the selection of outliers. However, the problematic prepositions from the above example run are clearly not outliers but rather highly typical examples – *in* and *from* are among the most frequent prepositions in this data set. It is unclear if density estimation can help in such a situation in order to ensure the selection of a more comprehensive range of prepositions.

The impact of over-confident estimates, especially for events with single observations, could potentially be reduced to some degree by *smoothing*. Using a simple technique such as Lidstone smoothing (add-λ smoothing), probability estimates for events with a single observation would be considerably less peaked. For example, with $\lambda = 1$, estimates for such cases will be either 1/3 or 2/3 in favour of noun-attachment and, accordingly, their uncertainty is $f_{unc}^{bin}(u,M) = 1/3$. In the case of prepositions with a more peaked distribution than that, repeated selection would eventually bring their uncertainty below this threshold of 1/3, and selection stops. On the other hand, the estimate of prepositions with a less peaked distribution will stabilise above that threshold so that the same problem of over-selection persistently continues. Also, smoothing would only be needed for sample selection, not for testing, and it is not clear how to determine the optimal setting of smoothing parameter λ.

4.1.2 Starting with a Larger, Randomly Selected Training Set

This section deals with another potential remedy, namely starting with a considerably larger initial training set. This should have a similar effect to the application of smoothing. Larger empirical counts would result in less peaked probability estimates for a wider range of prepositions such that they will have a better chance of being selected according to an uncertainty criterion. In this section, we will examine to what degree larger initial training sets can help solve the problems of uncertainty sampling identified above, and we let uncertainty sampling start with 100 and 1000 instances.

At the same time, we continue our investigation of the selection of out-of-coverage instances. The last experiment has shown that the preferred selection of out-of-coverage instances substantially improves uncertainty sampling. Clearly, the previous experiment has intensified this effect by starting with minimal coverage. This raises the question whether this factor still plays a role when using larger initial training sets with higher coverage, as is often done in active learning experiments. For example, after randomly sampling 100 instances, coverage is 93.5%; after 1000 instances, we have already covered 99% of all instances in the test set (Figure 4.2). Hence, one might assume that the effects from different settings for the final backoff level do not bear out anymore.

From now on, we will use a larger batch size of 100 instances in order to accelerate experimental turnaround. This corresponds to general usage of active learning. Figures 4.4 and 4.5 show results for initial training sets of 100 and 1000 instances. Interestingly, we find that using the altered setting still results in consistently better performance than the standard setting, conforming to the results from the previous experiment. This is the case for both starting points. When starting with 100 instances, improvements are significant from the beginning until ca. 13k instances have been annotated. When starting with 1000 instances, differences are less pronounced and significance between the two active learning conditions is given only in some parts of the learning curve. However, we can argue that the altered setting is significantly better than random sampling until ca. 16k instances, whereas the significance of the improvements over random sampling with the standard setting is at best sporadic. At any rate, these results clearly indicate that out-of-coverage instances should always be expressly pursued.

Returning now to the question of intrinsic shortcomings of uncertainty sampling, we see that uncertainty sampling still performs worse than random sampling until late

into the experiment when starting with 100 instances in both conditions. Looking at the sequence of selected prepositions reveals similar problems as in the previous experiment. Only when starting with 1000 instances, does uncertainty sampling gain sufficient momentum to outperform random sampling.

Such problems with uncertainty sampling on small training sets are not always so pronounced for other applications. Uncertainty sampling can afford us improvements over random sampling even for small training sets when applied to sequencing tasks and parsing, as we will see in the two chapters to follow. However, using other methods such as QBC generally gives better performance.

4.1.3 Summary

Uncertainty sampling as such ignores questions of coverage. We demonstrated that the naïve application of uncertainty sampling to a standard prepositional phrase classifier results in inferior performance due to a failure to target out-of-coverage examples. We addressed this problem by merely adjusting the backoff parameter for unknown prepositions in the base classifier, thus obviating the need to change the sample selection algorithm itself. This results in the preferred selection of out-of-coverage instances and in consistent improvements in terms of accuracy.

However, even when addressing coverage problems in this manner, we found severe shortcomings of uncertainty sampling to the degree that it can perform worse than random sampling. In our analysis, this can be attributed to the fact that uncertainty sampling cannot accurately identify unreliable parameter estimates as such. For the remainder of this chapter, we will examine the capacity of Query-by-Committee in this respect.

4.2 Query-by-Committee for Prepositional Phrase Attachment

In general, we expect Query-by-Committee methods to be well-suited to improve model parameters based on infrequent events. The random perturbation of models will result in larger variance for such parameters which, in turn, will be reflected by a higher degree of disagreement across the committee. Labelling instances associated with these parameters will help to decrease variance, and potentially increase classification accuracy. The experiments in this section will show to what degree QBC can

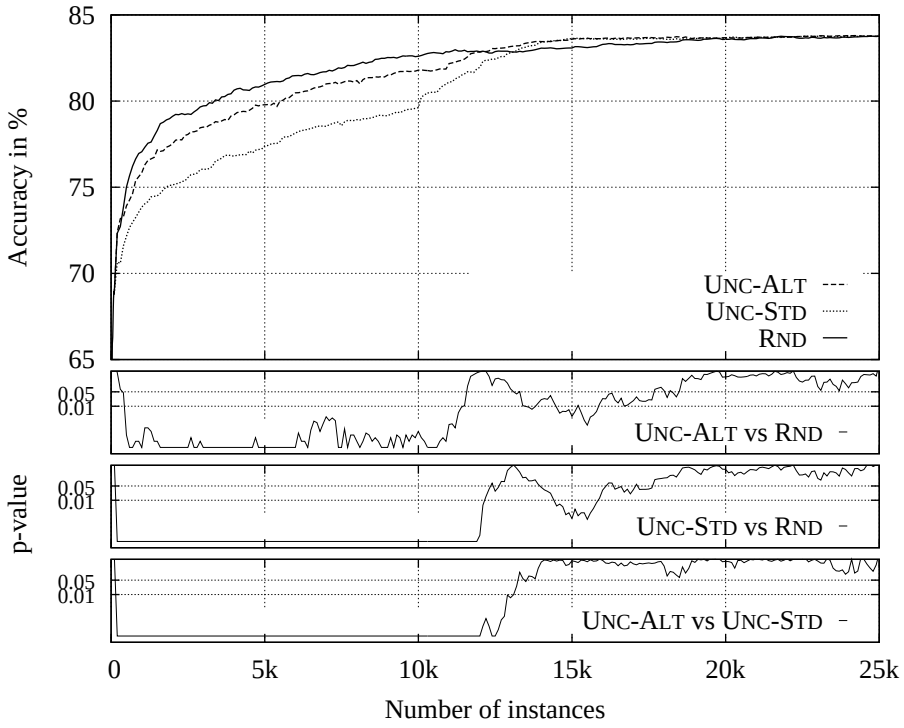

Figure 4.4: Backoff settings for uncertainty sampling (from 100 instances)

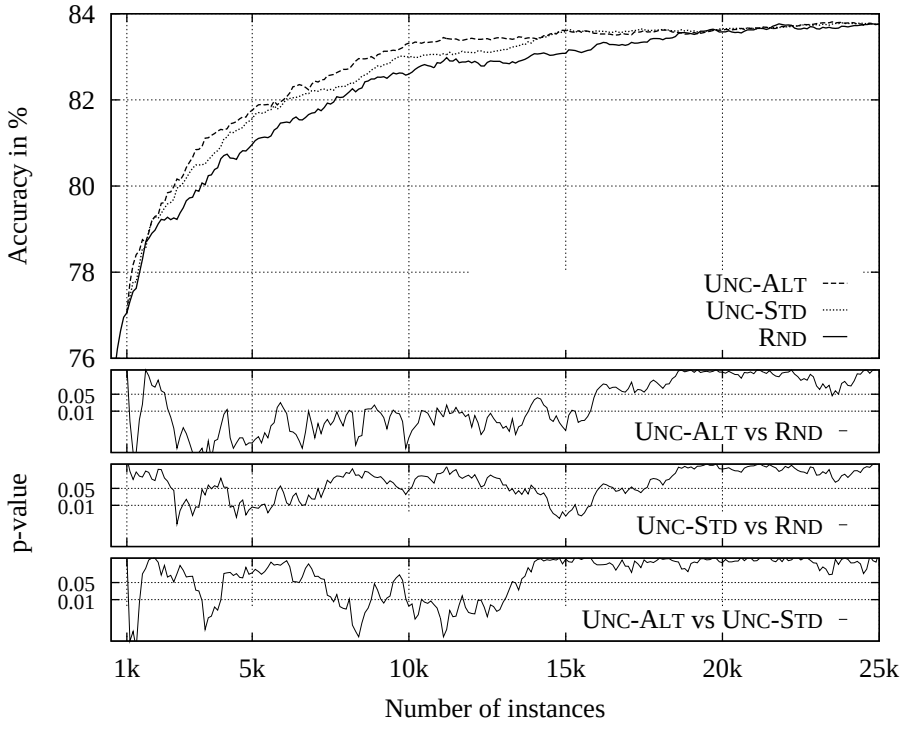

Figure 4.5: Backoff settings for uncertainty sampling (from 1000 instances)

overcome the problems we encountered with uncertainty sampling. However, introducing QBC brings with it a number of design decisions:

Which methods are suitable for model perturbation? We will examine two popular methods which have been used previously for active learning: i) bagging (Abe and Mamitsuka, 1998)), and ii) Dirichlet sampling (McCallum and Nigam, 1998).

Which metrics are suitable to quantify disagreement? We will compare the two most popular metrics i) vote entropy (Argamon-Engelson and Dagan, 1999) and ii) Jensen-Shannon divergence (McCallum and Nigam, 1998).

Changing the Base Classifier for QBC

Coverage plays an important role in the course of sample selection as we demonstrated above for uncertainty sampling. As part of the on-going thesis that unreliable parameters need special attention, we will examine the problem of out-of-coverage instances also in the context of QBC.

We show experimentally that also QBC suffers from coverage problems when using the unmodified classifier as defined in (Collins and Brooks, 1995). This is for similar reasons to the ones which we have identified above in the context of uncertainty sampling. It is easy to see that using the standard value $P_5(\mathbf{n}|u) = 1.0$ for the final backoff level or, in fact any constant value, will result in perfect agreement of the committee for instances with unknown prepositions, regardless of the chosen disagreement metric. Accordingly, such instances will be not be selected. To remedy this problem, we randomly sample values for $P_5(\mathbf{n}|u)$ from the uniform distribution in the range 0.0 to 1.0, $P_5(\mathbf{n}|u) \sim \text{uni}(0,1)$, to create a high degree of ensemble divergence for such instances.

For these experiments, we will use the following naming convention:

- QBC-STD: QBC, standard setting $P_5(\mathbf{n}|u) = 1.0$
- QBC-ALT: QBC, altered setting $P_5(\mathbf{n}|u) \sim \text{uni}(0,1)$
- RND: Random sampling

4.2.1 Experiments

In the experiments of this section, we evaluate all combinations of classifier perturbation methods and disagreement metric for the standard and the altered setting in the base classifier, using an ensemble size of 10. This gives rise to the following four different experimental conditions.

- Dirichlet sampling/JS-divergence
- Dirichlet sampling/vote entropy
- Bagging/JS-divergence
- Bagging/vote entropy

Dirichlet Sampling/JS-Divergence

Results for the first condition, combining Dirichlet sampling and JS-divergence, are shown for initial training set sizes of 100 and 1000 instances in Figures 4.6 and 4.7. The most notable finding is that QBC massively underperforms compared to random sampling, both for the standard and the altered setting, regardless of the starting point. We will explore later in this section what are the potential problems of this condition.

Furthermore, we find that the altered setting consistently outperforms the standard one. This is significant throughout except for the last few iterations when starting from 100 sentences; and significant after ca. 10k instances have been seen when starting with 1000 instances.

Dirichlet Sampling/Vote Entropy

Substituting vote entropy as a disagreement metric results in Figures 4.8 and 4.9. Again, we find that the altered setting consistently outperforms the standard one; significantly so until 18k instances have been sampled when starting with 100 sentences, and from 5k - 18k when starting with 1000 sentences.

QBC using the standard setting massively underperforms random sampling when starting with 100 instances; significantly until 18k instances have been sampled; and is only about as good as random sampling when starting from 1k instances. By contrast, QBC with the altered setting is at least as good as random sampling or better but significantly so only in some parts.

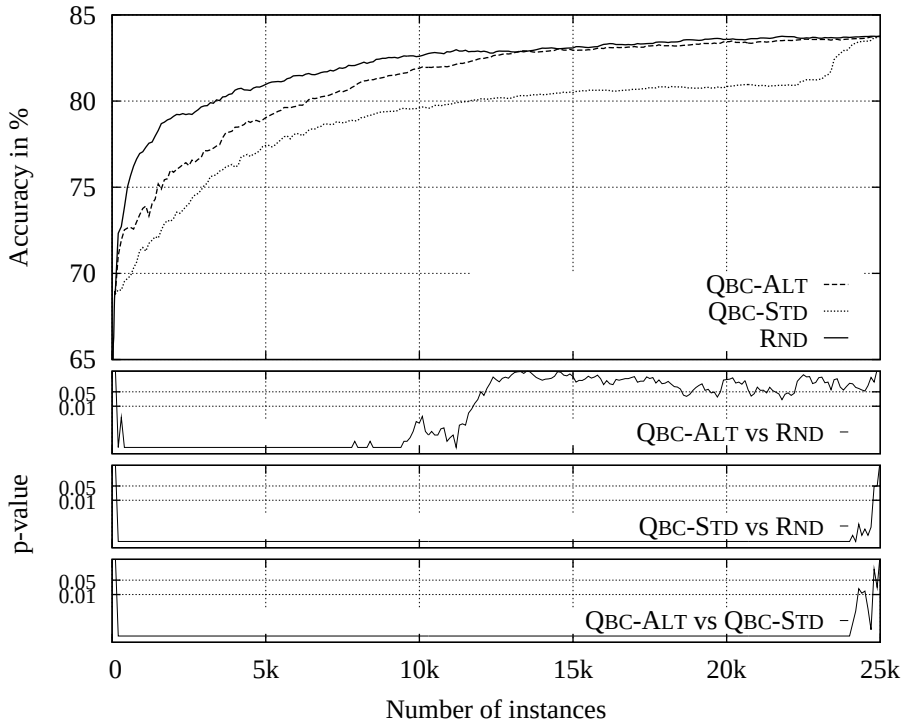

Figure 4.6: Backoff settings for Dirichlet sampling/JS-divergence (from 100 instances)

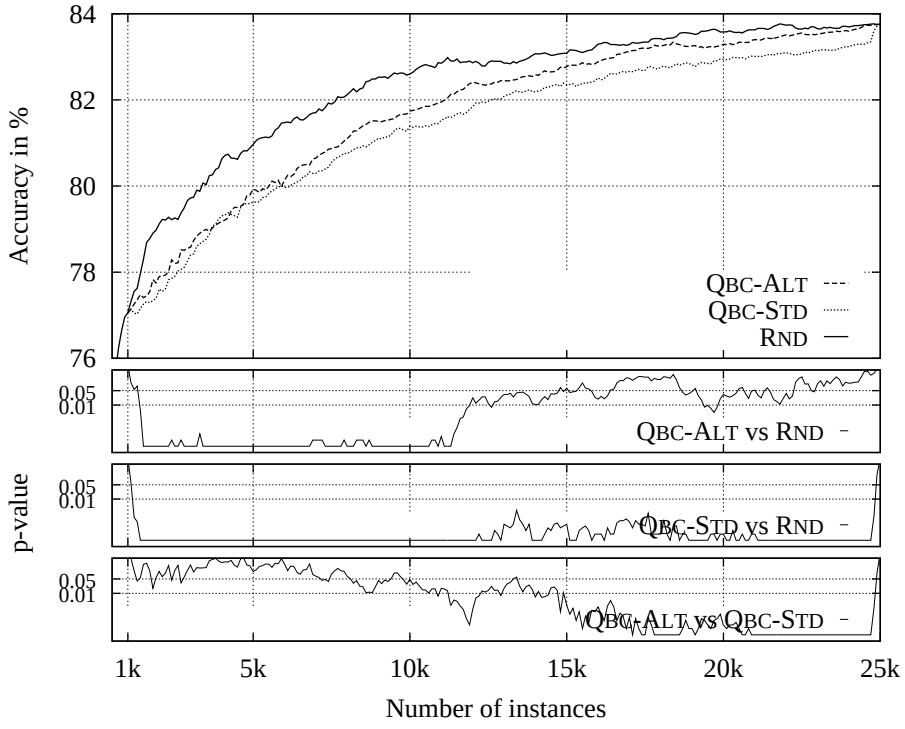

Figure 4.7: Backoff settings for Dirichlet sampling/JS-divergence (from 1000 instances)

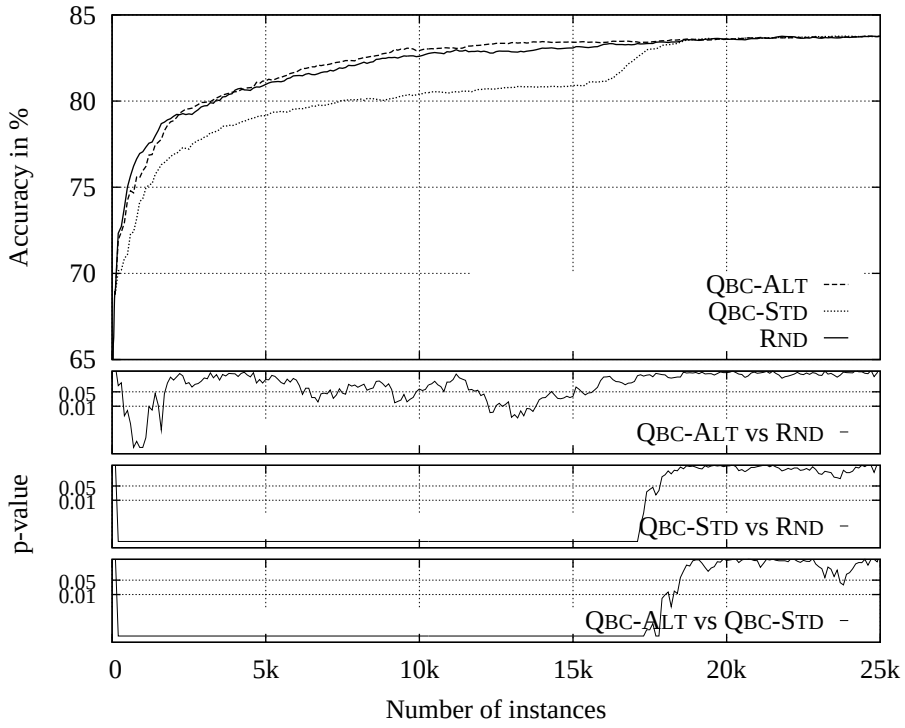

Figure 4.8: Backoff settings for Dirichlet sampling/vote entropy (from 100 instances)

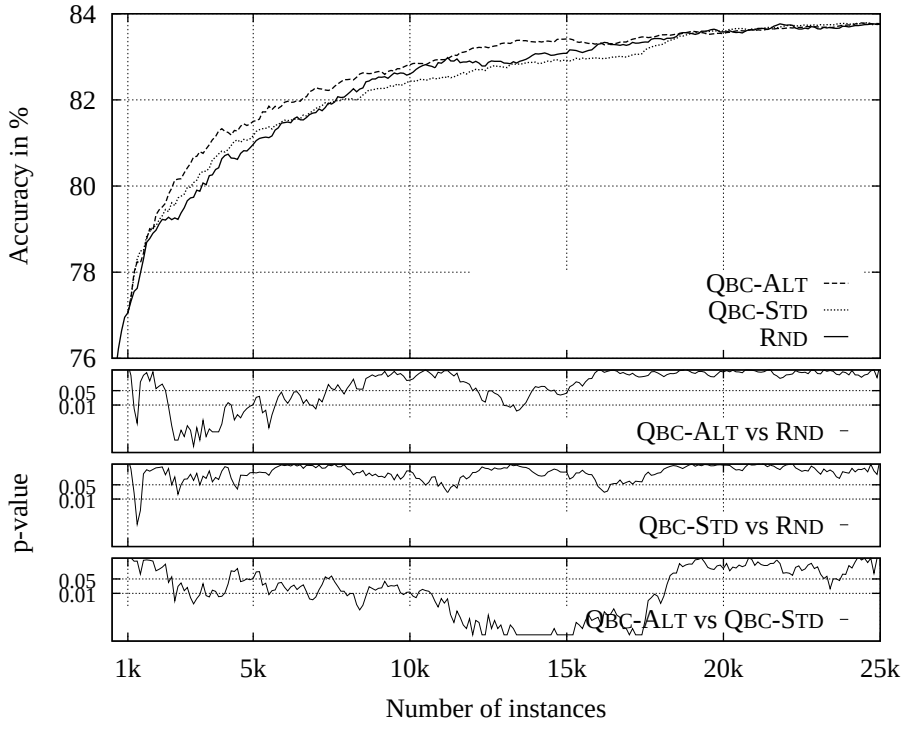

Figure 4.9: Backoff settings for Dirichlet sampling/vote entropy (from 1000 instances)

Bagging/JS-Divergence

Using a combination of bagging and JS-divergence in this experiment, we arrive at the results in Figures 4.10 and 4.11. Again, the altered setting consistently outperforms the standard setting for both starting points. The standard setting underperforms random sampling consistently when starting with 100 sentences; and is only about as good as random sampling when starting with 1000 sentences.

QBC with the altered setting, when starting from 100 sentences, is worse than random sampling until ca. 5k sentences have been sampled, and just as good afterwards. Only when starting from 1000 sentences, this combination is better than random sampling; but the difference is significant only in some parts of the learning curve.

Bagging/Vote Entropy

Finally, using bagging and vote entropy we get the results in Figures 4.12 and 4.13. Again, the altered setting outperforms the standard one throughout. The standard setting is worse than random sampling until ca. 11k instances have been sampled when starting with 100 instances. It performs better than random sampling when starting late, but not significantly throughout.

Most importantly, this is the best out of the four examined conditions and the only one where QBC with the altered setting consistently outperforms random sampling both when starting at 100 or at 1000 instances. QBC with the altered setting is significantly better than random sampling early on and until ca. 16k instances have been sampled for both starting points.

4.2.2 Summary

We saw in all examined conditions that it is beneficial to use the altered setting over the standard one. This conforms to the findings we made for uncertainty sampling and, again, supports our thesis that the preferred selection of out-of-coverage instances is important for sample selection.

Equally, or even more, important for QBC is a good choice of perturbation method and disagreement metric. Figure 4.14 and 4.15 gives an overview of results for all four combinations under a QBC-ALT setting, starting from 100 and 1000 instances. The best combination is bagging and vote entropy; a particularly bad combination is Dirichlet sampling and Jensen-Shannon divergence. More generally, we find that

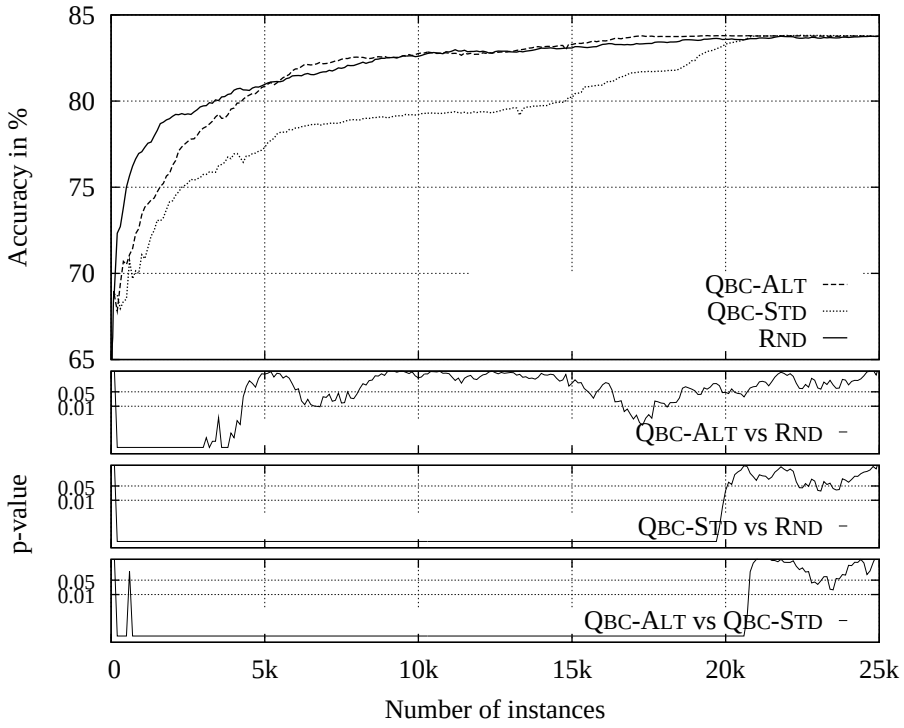

Figure 4.10: Backoff settings for bagging/JS-divergence (from 100 instances)

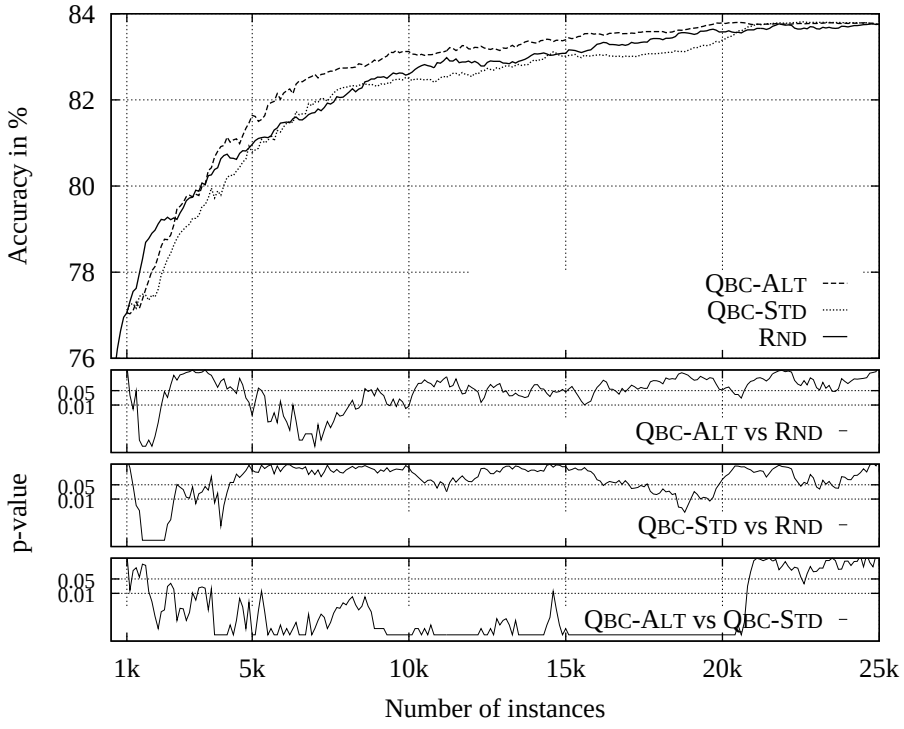

Figure 4.11: Backoff settings for bagging/JS-divergence (from 1000 instances)

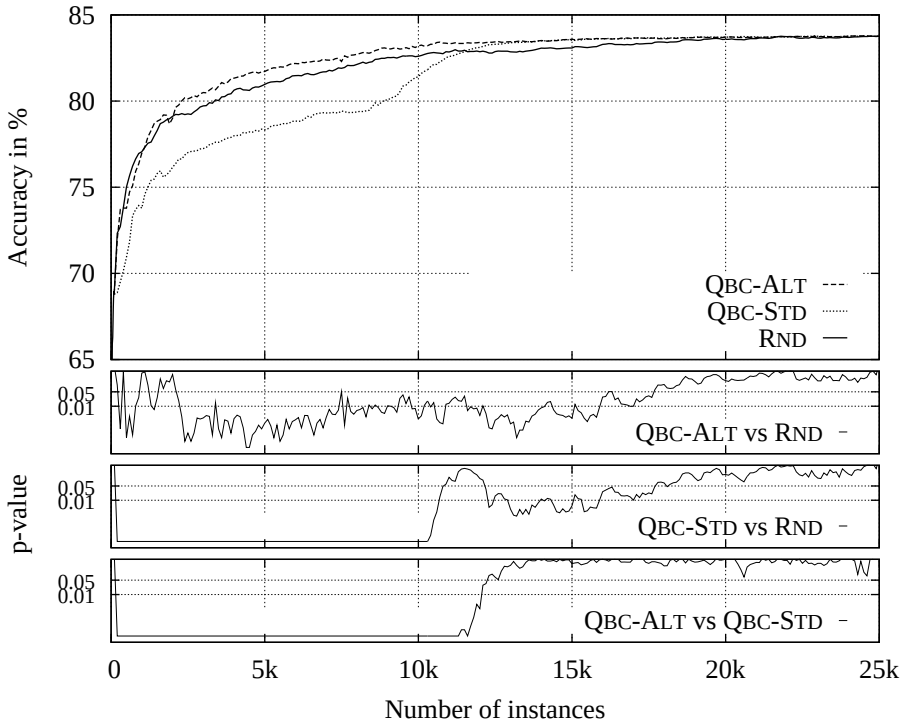

Figure 4.12: Backoff settings for bagging/vote entropy (from 100 instances)

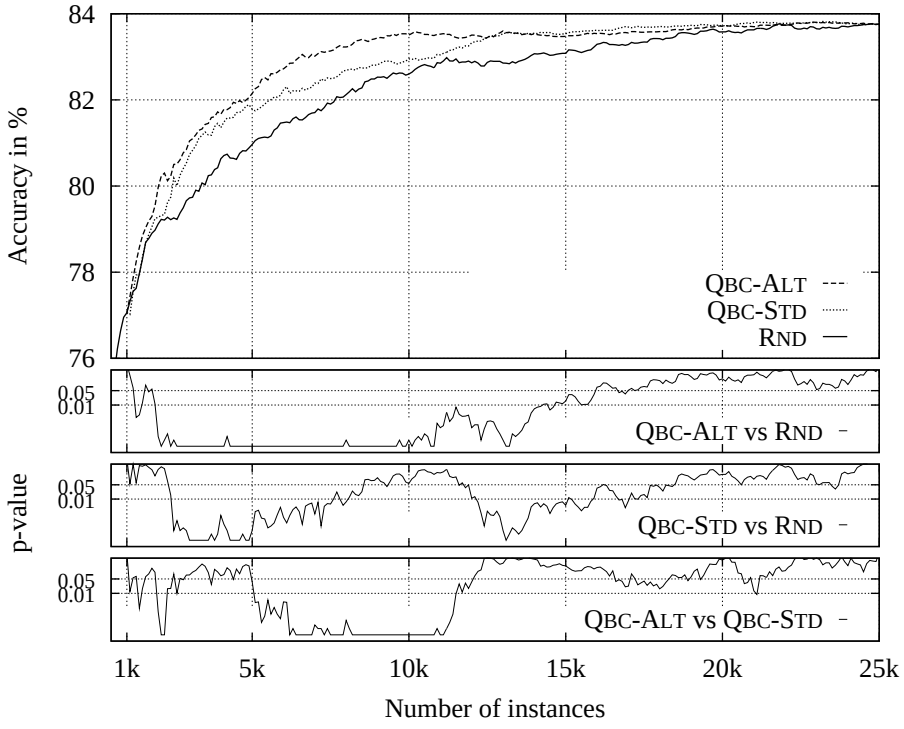

Figure 4.13: Backoff settings for bagging/vote entropy (from 1000 instances)

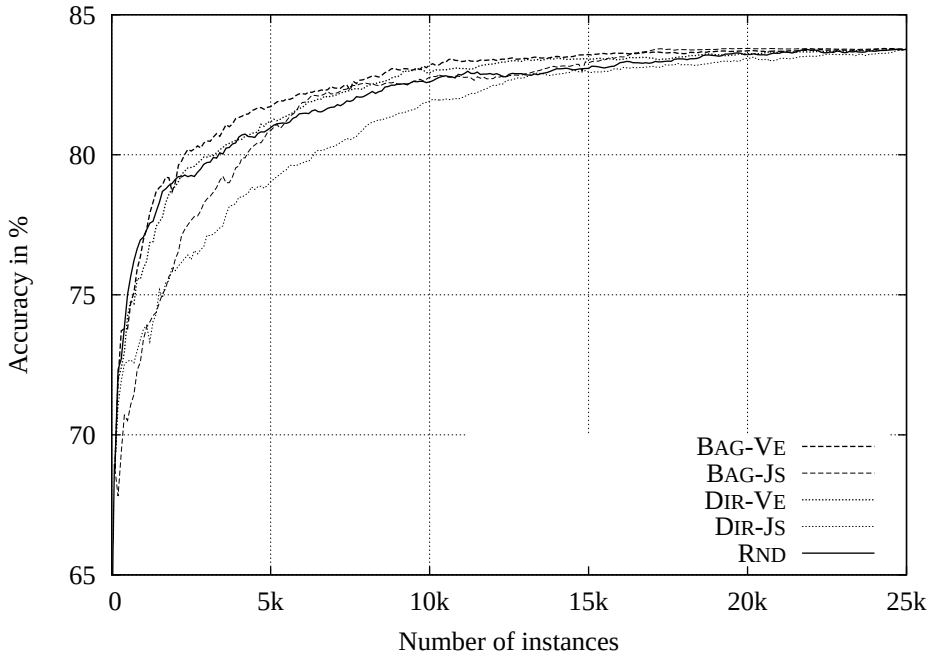

Figure 4.14: Ensemble creation methods and divergence metrics for QBC (from 100 instances)

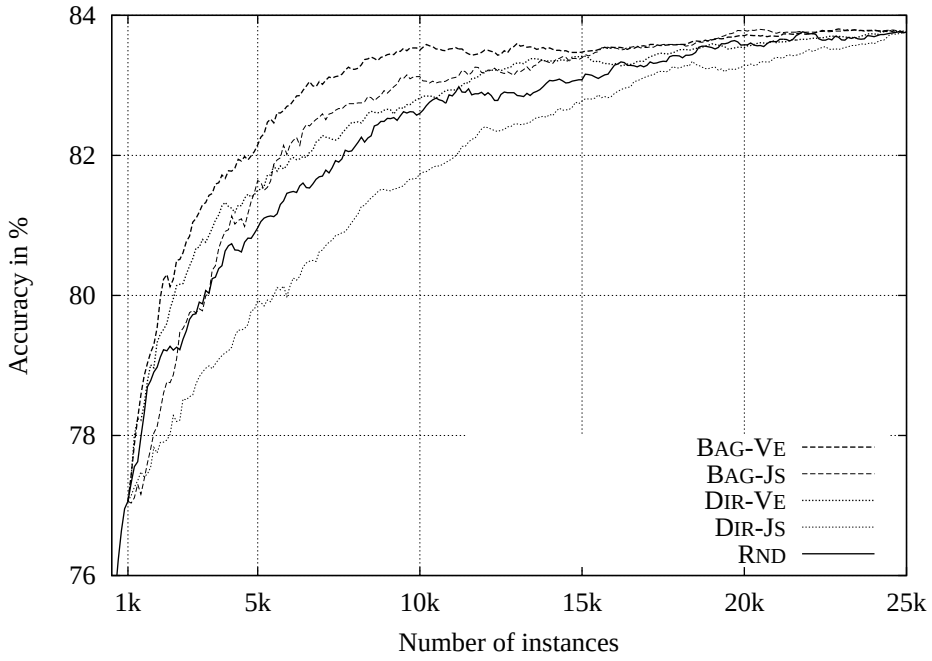

Figure 4.15: Ensemble creation methods and divergence metrics for QBC (from 1000 instances)

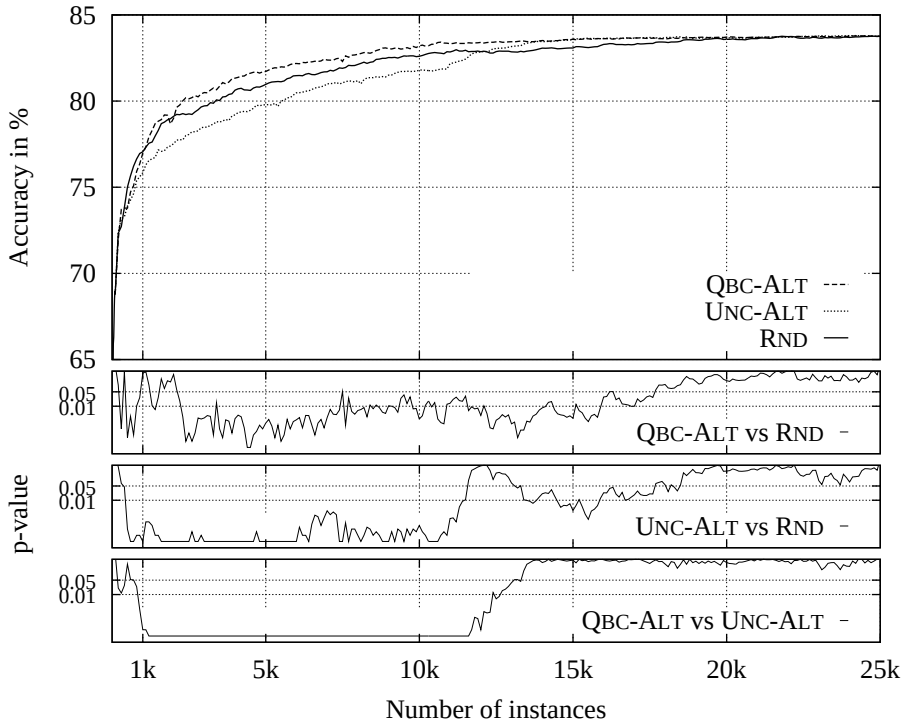

Figure 4.16: Best settings for uncertainty sampling and QBC (from 100 instances)

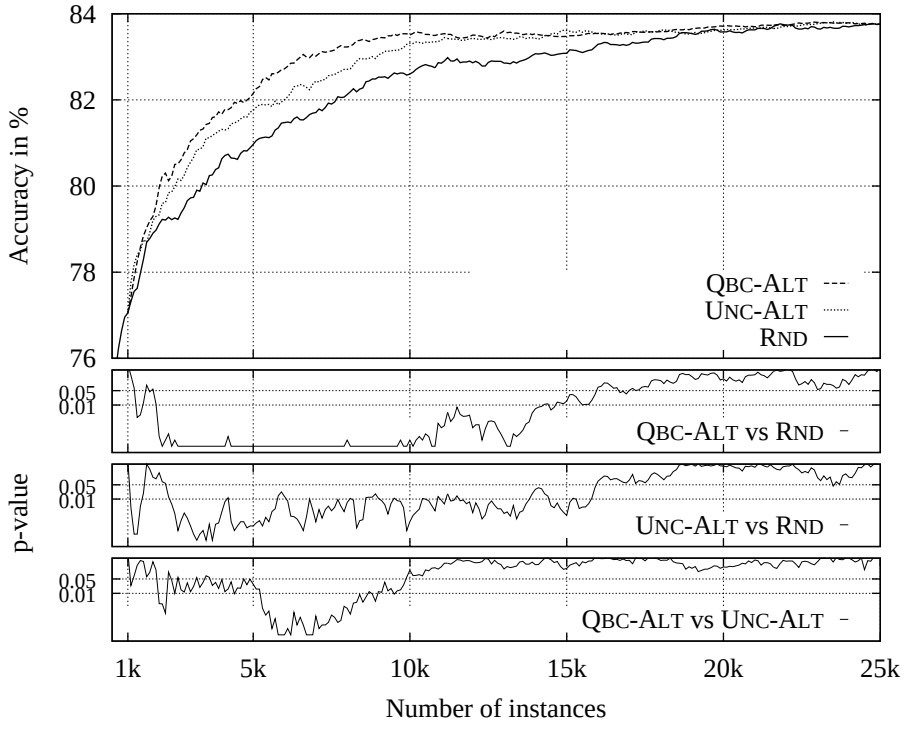

Figure 4.17: Best settings for uncertainty sampling and QBC (from 1000 instances)

		Bagging		Dirichlet Sampling			Proportion
Preposition	Random	Vote Ent.	JS-Div.	Vote Ent.	JS-Div.	Uncert.	noun-att.
1. of	26.5	2.9	3.0	15.5	24.3	2.9	98.8
2. in	16.8	26.5	23.4	23.0	15.7	28.6	46.6
3. to	12.5	10.5	13.2	11.0	13.7	12.7	22.4
4. for	10.2	19.4	16.6	16.4	10.8	18.3	45.9
5. on	6.5	11.7	10.9	9.1	7.4	10.2	46.1
6. from	4.5	4.9	7.5	4.7	5.2	6.6	34.2
7. with	4.4	6.0	7.3	5.6	5.3	7.5	36.9
8. at	3.2	3.2	3.7	3.3	4.3	0.7	20.1
9. as	2.4	0.4	0.7	1.5	2.8	3.2	18.9
10. by	2.2	2.9	2.8	1.4	1.5	0.2	27.0

Table 4.2: Distribution of prepositions in training set for different conditions

bagging outperforms Dirichlet sampling (dashed lines versus dotted lines); and vote entropy outperforms Jensen-Shannon divergence (thick lines versus thin lines).

Contrasting the best QBC result, bagging/vote entropy with the best uncertainty sampling result (both with the altered setting), we can see that QBC is always better than uncertainty sampling or at least as good, see Figures 4.16 and 4.17.

Discussion

To shine a light on these performance differences between QBC conditions, we look at the proportion of the ten most frequent prepositions in the training set in Table 4.2. For reference, we indicate their proportions when randomly sampling a training set of 10k instances in the second column. We find a roughly Zipfian distribution; the most frequent preposition *of* constitutes 26.5% of all instances in the training set; the ten most frequent preposition together cover almost 90%.

For each of the four QBC conditions (and for uncertainty sampling) using the altered setting, we ran a single fold of active learning; starting with 1000 randomly sampled sentences and iterating 90 times with a batch size of 100 instances, we eventually reach 10k instances in total.

For the two bagged conditions (bagging/vote entropy and bagging/JS divergence), the profiles of sampled prepositions in the training set (in columns 3 and 4) clearly deviate from random sampling: the most frequent preposition *of* now only consti-

tutes ca. 3% of all instances. Other prepositions are over-represented, for instance the preposition *in* went up from 16.8% to 26.5% under bagging/vote entropy. Incidentally, uncertainty sampling shows very similar deviations from random sampling.

On the other hand, we see that QBC using Dirichlet sampling and JS divergence (in column 6) quite faithfully reproduces the random profile. (QBC using Dirichlet sampling and vote entropy seems to follow a more hybrid pattern.)

How do distributional differences bear on classification accuracy? It may be surprising at first that methods which deviate from random sampling in their distributional patterns should perform better than methods which reproduce random sampling behaviour. We have to consider, though, that prepositions have vastly different biases with respect to their attachment preferences, cf. the last column of Table 4.2. For example, preposition *of* is very strongly biased in favour of noun-attachment; preposition *in*, on the other hand, is almost balanced between noun- and verb-attachment.

In this light, the undersampling of preposition *of* found in the bagging-based methods turns out to be very economical since the bias of preposition *of* allows one to learn its distribution from only a small number of training instances, and still achieve almost 99% accuracy. This is particularly beneficial since it will help to reliably classify a substantial proportion of the test set. On the other hand, the almost balanced preposition *in* presumably is harder to learn. Under a bagging-based method, this preposition receives considerably more attention than its proportion would predict from random sampling.

By contrast, methods with Dirichlet sampling more closely reproduce the distributional profile of random sampling and thus spend substantial annotation effort on the peaked and easy-to-learn *of* distribution, thus withholding annotation effort from more difficult cases such as *in* and *for*.

This is a clear demonstration that a good sample selection method must not necessarily reproduce the distributional patterns found in random sampling. Rather it should devote annotation effort to difficult distributions and spend less effort on easy ones.

How does a method *know* which distributions are easy? We choose *of* as a prototypically easy preposition, and contrast the scoring of such instances under one condition which assigns high scores (using Dirichlet sampling and JS-divergence) and another condition which assigns low scores (using bagging and vote entropy).

Running a single selection round using Dirichlet sampling and JS divergence, with

an initial training set of 1000 instances, we picked from the batch of selected examples a (prototypical) *of* instance with features $\langle retain, title, of, chairman \rangle$ and a score of 0.45. It had no matches on the first two levels, $P_1(\mathbf{n}|u)$, and $P_2(\mathbf{n}|u)$ (according to the the backoff probability scheme from Equation 2.2); on level 3, there was one matching training instance which was noun-attached, with features $\langle retain, *, of, * \rangle$. On the next back-off level, there were 268 instances, matching $\langle *, *, of, * \rangle$, the vast majority being noun-attached.

$$P_3(\mathbf{n}|u) = \frac{1}{1}$$
$$P_4(\mathbf{n}|u) = \frac{265}{268}$$

A characteristic of Dirichlet sampling is that it only uses empirical counts on one level at a time; in this particular example with the most specific match on level 3, counts are $\langle \text{nouns} = 1, \text{verbs} = 0 \rangle$. Assuming a uniform prior of one, we sample from a Dirichlet distribution with parameters $\langle 2, 1 \rangle$. Using a simulation of a very large ensemble, we find that the expected Jensen-Shannon divergence for such a set of counts converges towards 0.19. However, with just 10 trials we can expect considerable variance, and the score of 0.45 is well within that range.

Under bagging and vote entropy, the same instance receives a score of 0. If the single training instance which supports parameter $P_3(\mathbf{n}|u)$ is present in the bagged training set, the test instance $\langle retain, title, of, chairman \rangle$ will be classified as noun-attached. If the training instance is deleted (through bagging), the classifier backs off to parameter $P_4(\mathbf{n}|u)$. In this case the estimate will be close to 1.0, and again the test instance will be classified as noun-attached. Thus, ensemble members will always vote for noun-attachment under a bagged training set. The minimal disagreement for such instances entails a dispreferred selection which is beneficial as we have seen above.

One reason why we did not see disagreement for the condition using bagging and vote entropy is that we did not have to introduce a prior for the Dirichlet distribution. Furthermore, both parameters $P_3(\mathbf{n}|u)$, and $P_4(\mathbf{n}|u)$ had empirical support with the same polarity (in favour of noun-attachment). On the other hand, if these parameters would have had support of different polarities we would expect a higher degree of disagreement. Presumably, it would be worthwhile to learn the true label of such an instance. Such a situation would not be recognised by a method using Dirichlet sampling, again, because the disagreement is established for one level at a time.

4.3 Conclusion

The experiments in this chapter have shown that the naïve application of popular active learning methods such as uncertainty sampling and QBC can result in suboptimal performance since they lack a principled way of targeting out-of-coverage instances. We demonstrated how a simple change in the base classifier can increase coverage and accuracy, both for uncertainty sampling and QBC, without actually having to change the definition of the sample selection schemes themselves.

Even when applying this method, we found that uncertainty sampling can underperform compared to random sampling, in particular when starting with small training sets. We show that the very notion of uncertainty used in uncertainty sampling can mislead the sample selection process. In certain situations, good parameter estimates attract more annotation effort simply because they continue to look more uncertain than parameters with low counts and unreliable estimates.

QBC generally has a more principled way of addressing unreliable parameters, beyond out-of-coverage instances, and can achieve substantially better results than uncertainty sampling. However, QBC performance depends on setting a number of parameters right. Best performance is achieved using bagging as a randomisation technique (rather than sampling from local distributions) and employing vote entropy for scoring (rather than Jensen-Shannon divergence). In a detailed analysis, we showed how this choice can influence the proportion of selected prepositions. Substantial savings were achieved by "recognising" that prepositions with a biased label distribution required less annotation effort than prepositions with a more balanced distribution.

In this chapter, we have demonstrated that treating unreliable parameters is an important objective for active learning in the domain of prepositional phrase attachment. In particular, the targeted selection of unknown prepositions causes significant improvements. Prepositional phrase attachment is arguably a very simple task which we selected mainly for expository purposes. The simple probabilistic model of the classifier makes error analysis very easy. Furthermore, previous work in the same domain allowed for comparison (Hwa, 2004).

In the next two chapters, we will look at natural language processing problems which are both more difficult and more currently researched, namely sequencing tasks and syntactic parsing, in order to demonstrate that the principle of directly treating unreliable parameters is vital for active learning.

Chapter 5

Unreliable Parameters in Sequence Labelling

As maintained throughout this thesis, it is not sufficient for active learning methods to only improve the quality of parameter estimates within the current model, they should also expand the model structure where appropriate. We show in this chapter that this is important in application to sequence labelling as well.

The labelling of sequences is ubiquitous in natural language processing. Examples for sequence labelling include part-of-speech tagging and named entity recognition both of which we will deal with in this chapter. Sequencing tasks are more challenging than prepositional phrase attachment which we investigated in the last chapter. Labelling decisions in a sequence are carried out jointly, in other words, the labelling of a token is not only dependent on its individual tag distribution but also on the labelling decisions in the neighbouring context.

We will use Hidden Markov Models for the experiments in this chapter. They show close to state-of-the-art performance for sequencing tasks (Brants, 2000b) but are considerably faster to train than discriminative models (Ratnaparkhi, 1996). This is of course critical for active learning experiments where we have to train and retrain models many times. In particular, we will use Ingo Schroeder's Acopost tagger (Schröder, 2002). This is a freely available, reverse-engineered version of the TnT tagger.

In the context of prepositional phrase attachment, we dealt with unknown prepositions as a prime example for insufficient model structure. A corresponding problem in sequence labelling are unknown words, that is, a situation where the current lexicon does not cover all inputs. While backoff methods such as suffix tries can provide reasonable guesses for unknown words, tagging accuracy in such cases is generally

considerably lower than for known words. Another manifestation of insufficient model structure concerns label sequences which have not been observed in the training set, but are necessary for decoding test examples. In HMMs, these occur as unobserved state transitions. In order to avoid zero probabilities when estimating parameters in such cases, we need to apply smoothing or backoff. Still, we expect unobserved state transitions to contribute to a higher variance and higher error rate.

Standard active learning methods such as uncertainty sampling and QBC have no direct mechanism to deal with unknown word problems or unsupported transition probabilities. We introduce a novel method for sequencing tasks which directly addresses deficient model structures by selecting examples which suffer from many unsupported model parameters. In particular, this method counts the unknown words in a sentence and computes the expected number of unsupported transitions. These two quantities then are combined into a single sample selection score. Unknown word problems are easy to quantify: we use the number of unknown words in a sentence directly as a score. The problem of counting the number of unsupported transitions can be formulated as an expectation over all possible label sequences. We propose a novel dynamic programming algorithm which computes the expected number of unsupported transitions over all possible sequences implicitly.

This chapter extends joint (unpublished) work with Trevor Cohn.

Chapter Structure

In Section 5.1, we start by explaining why we do not consider alternative smoothing settings for active learning in application to sequence labelling in contrast to the previous and the subsequent chapter.

In the following four sections, we present results for active learning in application to part-of-speech tagging as a prototypical sequencing task. Section 5.2 has results for uncertainty sampling, Section 5.3 for QBC, and Section 5.4 for the count-based method. In Section 5.5, we summarise results for part-of-speech tagging.

In Section 5.6, we apply this range of active learning methods to named entity recognition. We discuss the unexpected result that uncertainty sampling outperforms all other methods, and conduct experiments which link this result to the size of the tagset involved in the task.

Section 5.7 concludes the chapter.

5.1 No Altered Smoothing Settings in Sequence Labelling

In the previous and in the next chapter, we demonstrate how modified smoothing or backoff settings allow the identification of training instances which help to overcome unreliable parameter estimates.

Can such techniques also be used for sequencing tasks? We can, for example, disable smoothing and allow zero probabilities to occur for unobserved transitions in the HMM and for unknown words. We can then easily identify sentences which suffer from such problems as they become undecodable. This would presumably help the selection process to identify unreliable parameters. In exploratory experiments, we found that such a method does not perform well in the context of sequencing task. Disabling smoothing renders the majority of instances in the pool as out-of-coverage and makes selection effectively perform like random sampling.

For these reasons, we do not consider different smoothing or backoff settings as in the other chapters. A major contribution of this chapter however is a more fine-grained measure than the Boolean out-of-coverage criterion. We will count the number of zero probabilities and the number of unknown words.

5.2 Uncertainty Sampling for Part-of-Speech Tagging

For active learning for sequencing tasks one has to decide on what to label. It is conceivable to select single words for annotation. In this case, one would have to train the classifier on partially annotated sentences using some kind of semi-supervised learning scheme. For example, Scheffer et al., 2001 explored the use of expectation-maximisation in this situation. Doing so however raises difficult questions about mediating between cost factors annotation time for a word and reading time for the surrounding context. To avoid this we decided to label instances on the sentence level. To determine an uncertainty score for a sentence we average over the uncertainty score for the tag distributions of individual tokens. We show results in Figures 5.1 and 5.2 for initial training set sizes of 100 and 1000 sentences. Uncertainty sampling (UNC) is significantly better than random sampling throughout after the first iteration.[1]

[1] We also let uncertainty sampling start from a single example. Results are very similar to the displayed results.

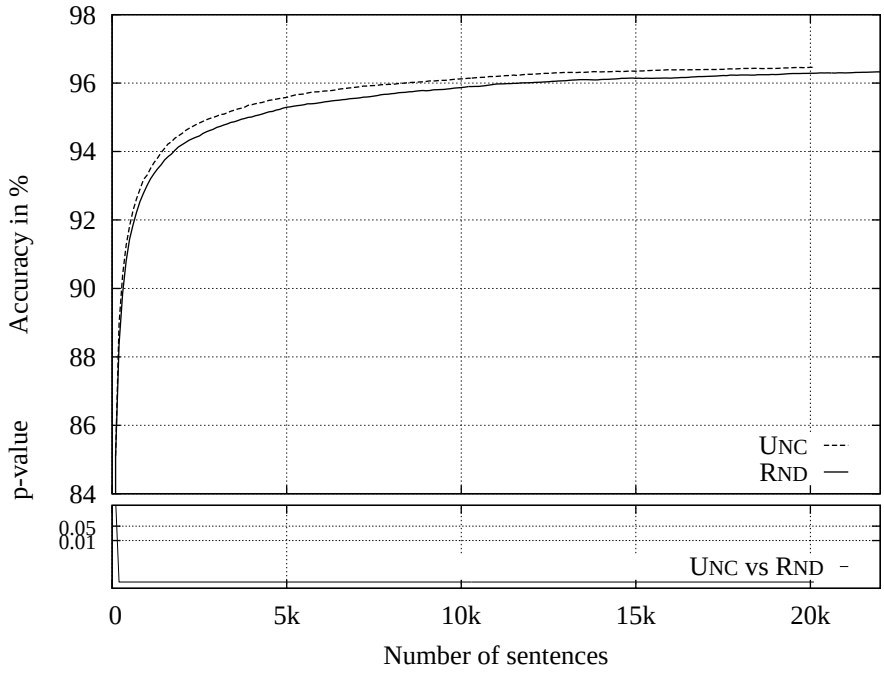

Figure 5.1: Uncertainty sampling (from 100 sentences)

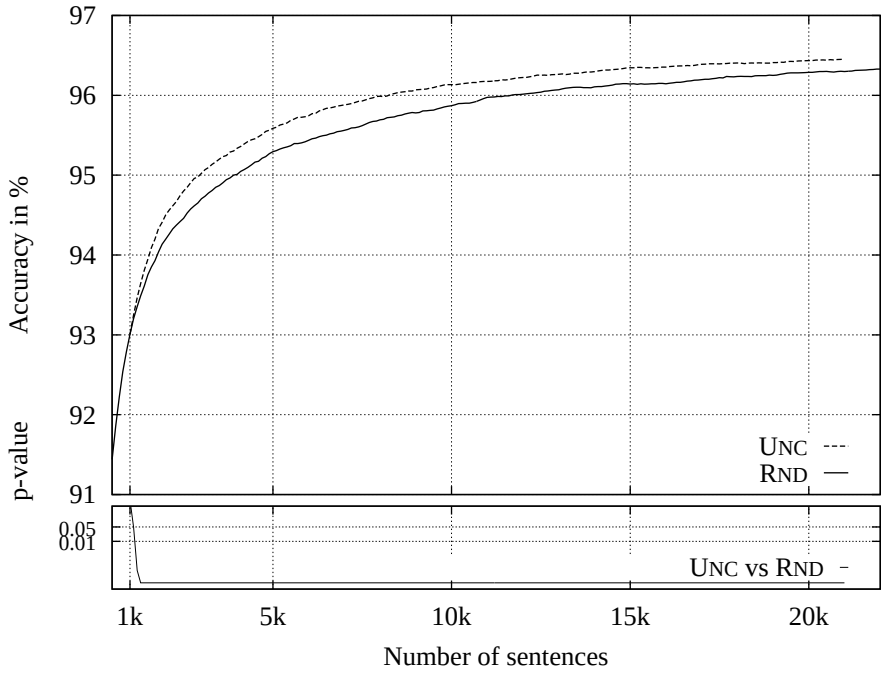

Figure 5.2: Uncertainty sampling (from 1000 sentences)

5.3 Query-By-Committee for Part-Of-Speech Tagging

As we have seen in the previous chapter, using QBC as an active learning method can result in substantial improvements over uncertainty sampling. In a first set of experiments for QBC in application to part-of-speech tagging, we try to find an optimal set of parameters. We will investigate the influence of the following parameters.

- Sampling Method
- Divergence Metric
- Ensemble size

5.3.1 Experiments

Experiment 1 In the first experiment, we compare sampling methods, namely bagging and Dirichlet sampling, while using vote entropy as the divergence metric. Incidentally, in this setting the condition with Dirichlet sampling corresponds to the settings employed in Argamon-Engelson and Dagan, 1999 which also concerns part-of-speech tagging. We however do not explicitly try to adjust their "heat" parameter.

We show results using initial training set sizes of 100 and 1000 sentences in Figures 5.3 and 5.4 respectively. We first note that both conditions significantly improve over random sampling throughout from the first iterations on. When comparing both conditions against each other, we find that the bagged method is consistently as good or better than the Dirichlet sampled method. Bagging is significantly better in almost all iterations when starting with 100 sentences. When starting with 1000 sentences, it is significantly better in almost all iterations until ca. 18k sentences have been sampled.

Experiment 2 Using Jensen-Shannon divergence as a disagreement metric, we contrast bagging and Dirichlet sampling in Figures 5.5 and 5.6. Again, we find that using QBC significantly improves over random sampling throughout. Comparing both conditions against each other, there are hardly any differences in performance. Only in the early phases, bagging can be seen to be significantly better than Dirichlet sampling.

Experiment 3 From the two previous experiments, we find that the combination bagging/vote entropy performs the best. Sticking with this setting we are now going to examine the influence of ensemble size on performance. We will contrast the performance of the best current setting using five ensemble members with an ensemble of

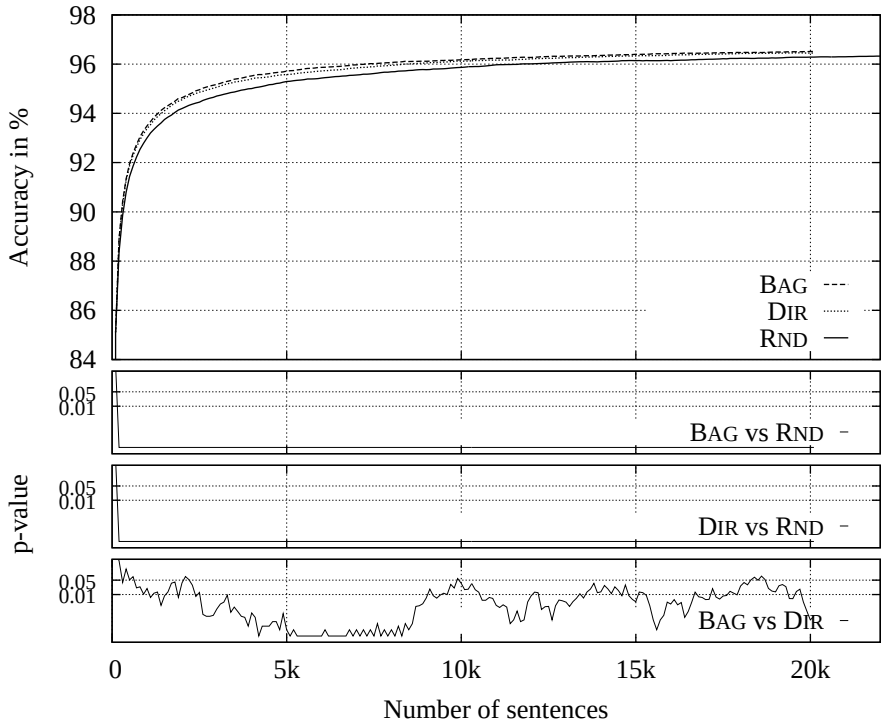

Figure 5.3: Ensemble creation methods using vote entropy (from 100 sentences)

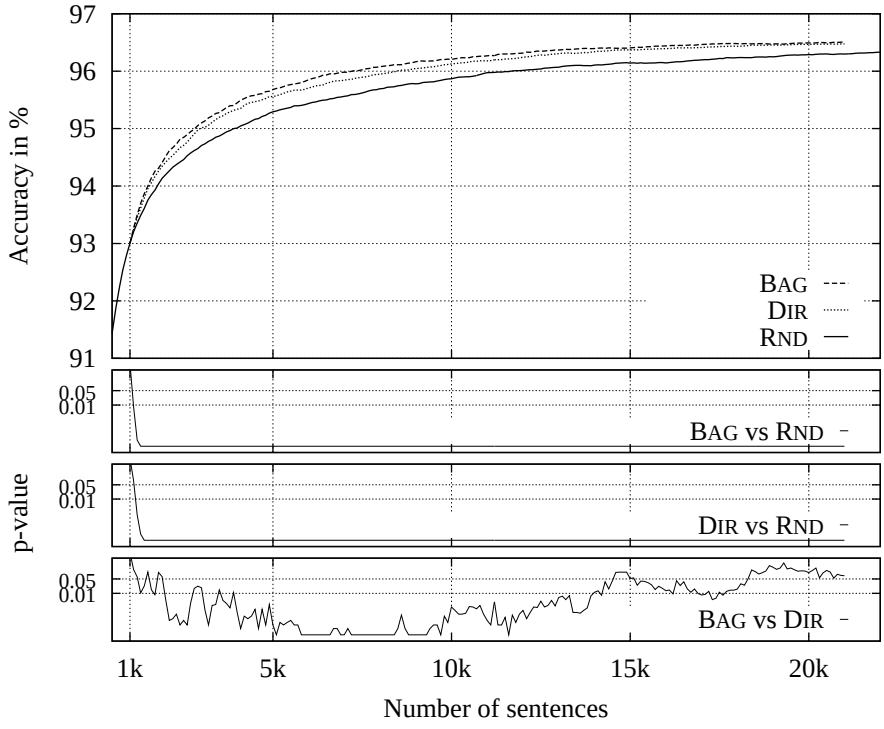

Figure 5.4: Ensemble creation methods using vote entropy (from 1000 sentences)

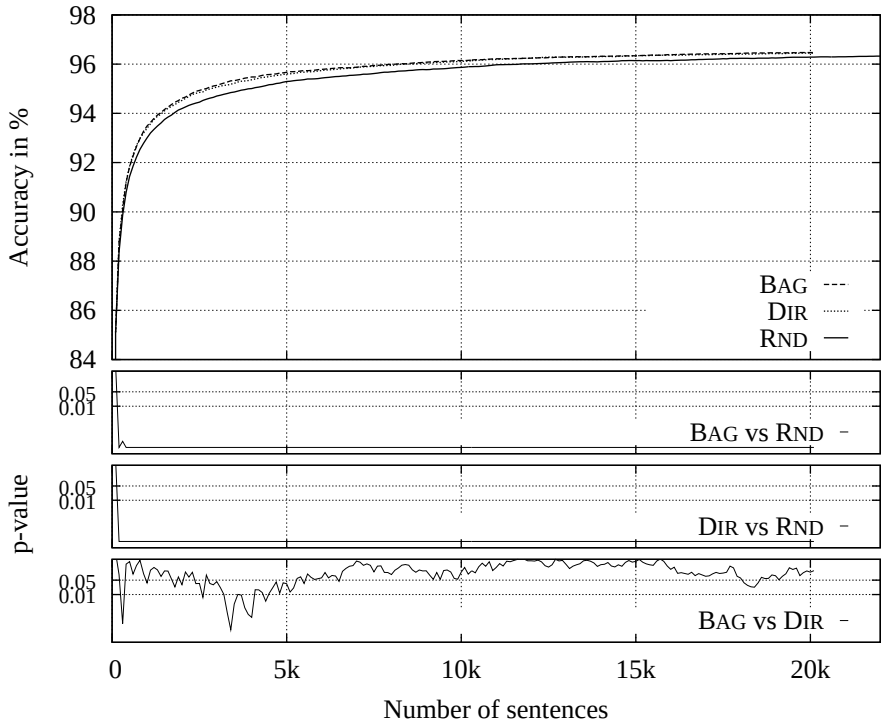

Figure 5.5: Ensemble creation methods using JS-divergence (from 100 sentences)

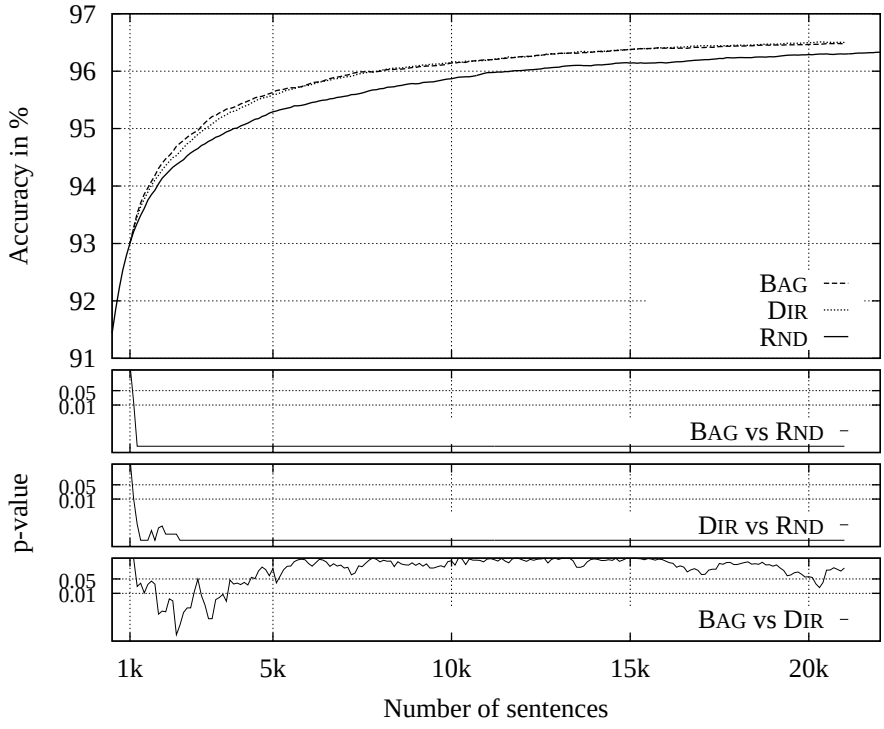

Figure 5.6: Ensemble creation methods using JS-divergence (from 1000 sentences)

10 members. Again, we show results for two different initial training set sizes in Figures 5.7 and 5.8. Performance is almost indistinguishable. Only in the final phase of the learning curve is there a significant improvement by using a larger ensemble.

Experiment 4 Finally, we are interested to see how the QBC method with the best set of parameters performs with respect to uncertainty sampling. Results are shown in Figures 5.9 and 5.10. Just like QBC, uncertainty sampling outperforms random sampling significantly throughout. In comparison with each other, QBC is always as good or better than uncertainty sampling with long stretches where QBC improvements are significant. However, uncertainty sampling is a surprisingly strong baseline. In particular, when starting with a smaller training set, improvements cannot be shown to be significant throughout.

5.3.2 Summary

In this first set of experiments, we have found an optimal set of parameters for QBC, namely bagging as a sampling method and vote entropy as a divergence metric. Increasing the ensemble size from 5 to 10 made virtually no difference. In this setting, QBC can be seen to be as good or better than uncertainty sampling. Also, this setting presents an improvement over the results in (Argamon-Engelson and Dagan, 1999) while using a conceptually very simple method of sampling from the training set without the need of setting an extra parameter.

5.4 A Novel Count-Based Method

The hidden Markov model we use for the experiments in this chapter uses smoothed probability estimates in the form of backing-off when observing novel state sequences and novel observations, as described in Section 2.2.3. Such smoothing allows the model to estimate the probability of sequences with unseen label sequences and out-of-vocabulary words. However, examples in the pool of candidates which can only be processed via (multiple) backing off will suffer from inaccurate estimates, and thus methods such as uncertainty sampling may be unreliable.

We present a novel active learning method which produces a score according to the number of times the model encounters probability estimates which are based on missing events. In the case of an HMM, this amounts to statistics over backed-off transition

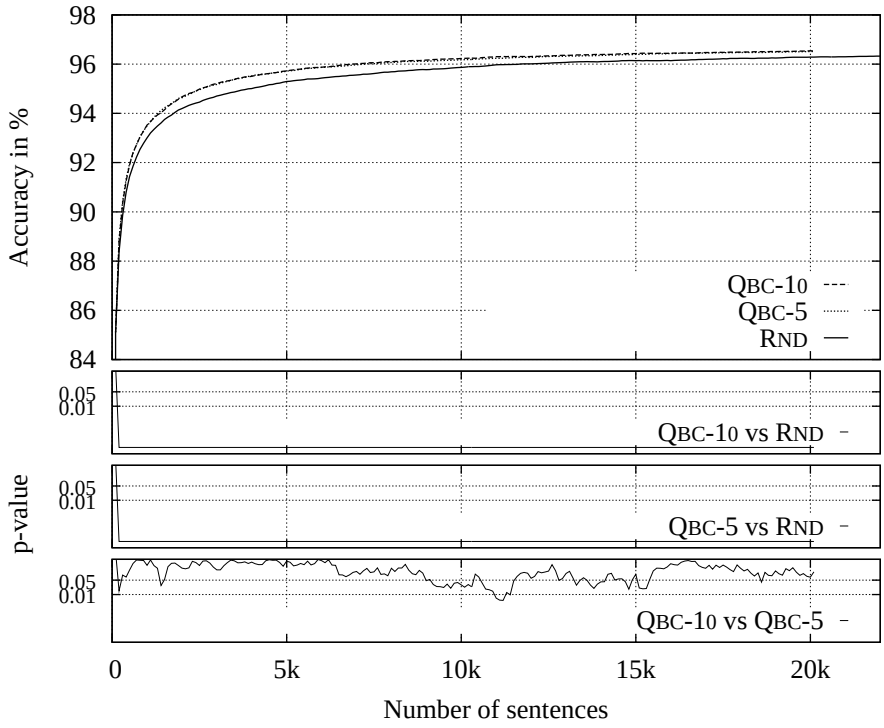

Figure 5.7: Ensemble sizes for best QBC (from 100 sentences)

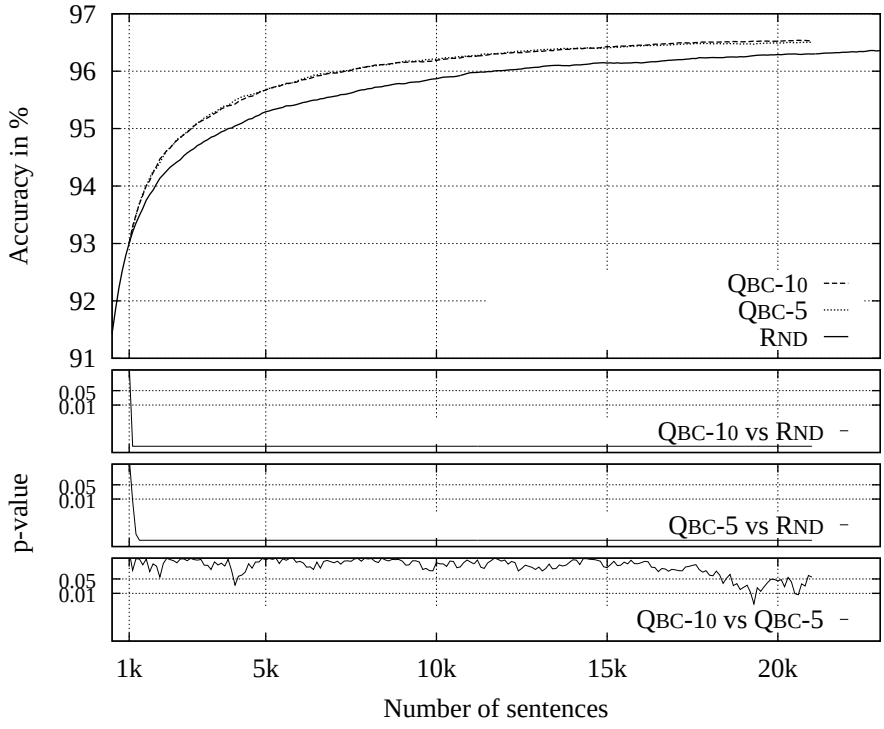

Figure 5.8: Ensemble sizes for best QBC (from 1000 sentences)

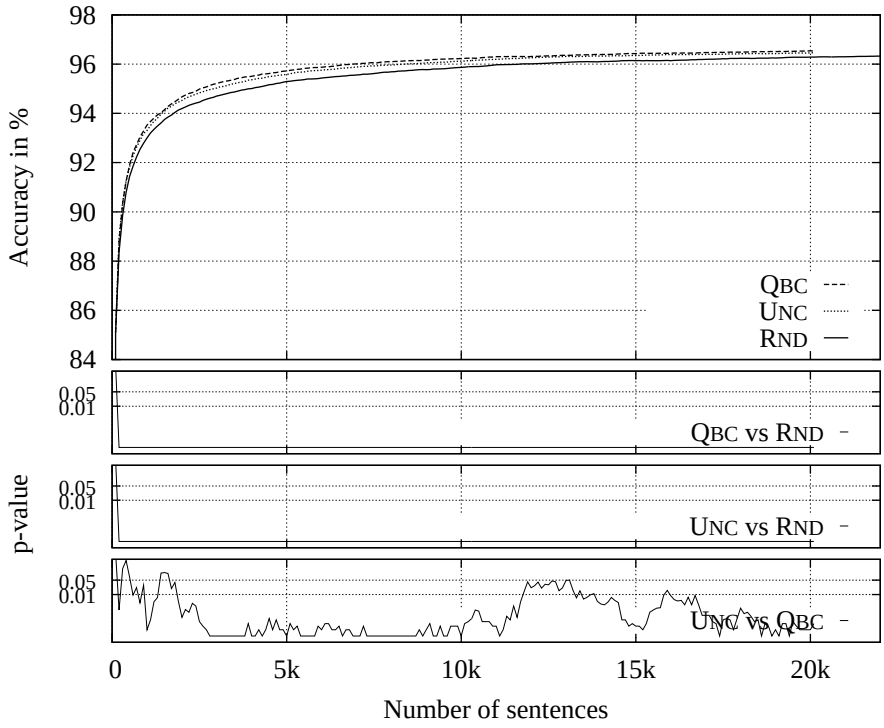

Figure 5.9: Comparing uncertainty sampling with best QBC (from 100 sentences)

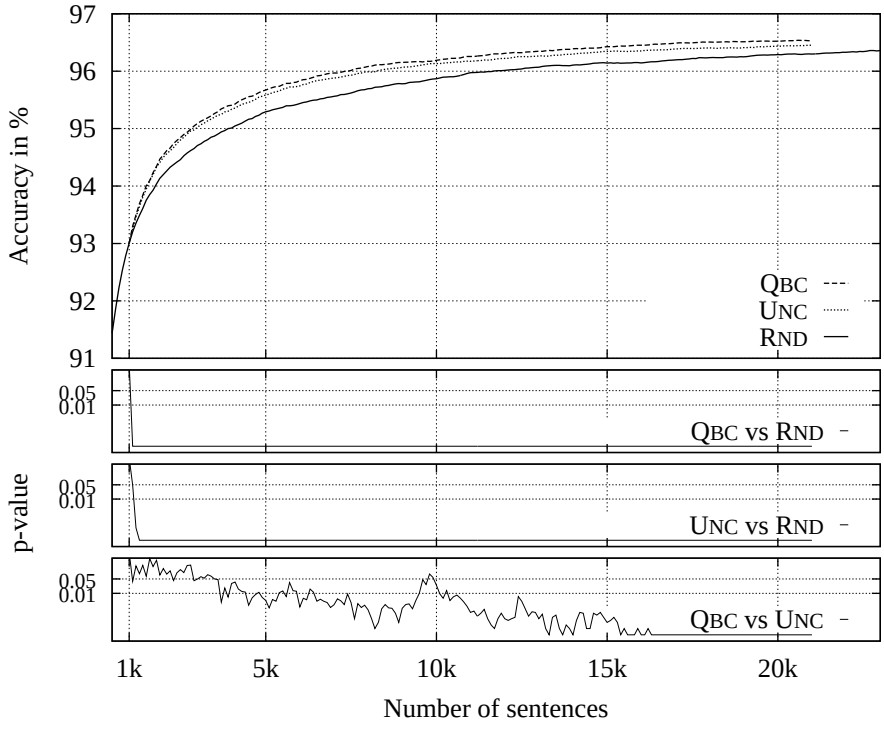

Figure 5.10: Comparing uncertainty sampling with best QBC (from 1000 sentences)

trigrams		words		average
count	norm	count	norm	of norm
2.08	0.52	5	0.5	0.51
1.28	0.32	2	0.2	0.26
0.64	0.16	3	0.3	0.23

Table 5.1: Creating a single score from the number of unknown words and the (expected) number of unsupported transitions

probabilities and backed-off lexical emission probabilities. Intuitively, a sentence for which the analysis depends on many such smoothed probabilities should be regarded as informative. Hence, its annotation and inclusion in the training set allows the system to model its novel properties.

The number of unknown words $c_w(\mathbf{o})$ in observation $\mathbf{o}$ can be established by simple lexicon lookup. The situation is more complicated with the number of backed-off transition probabilities since the generating state sequence $\mathbf{s}$ is hidden. If the sequence was labelled, the degree to which backing-off is required for interpretation can be measured by simply counting the number of unsupported trigrams. For an unlabelled example, we can average this count over all possible sequences, weighted by the probability of each labelling:

$$\mathrm{E}_\theta[c(\mathbf{o})] = \sum_{\mathbf{s}} p_\theta(\mathbf{s}|\mathbf{o}) c(\mathbf{s},\mathbf{o}) \tag{5.1}$$

where $c(\mathbf{s},\mathbf{o})$ counts the number of unsupported trigram transitions in sequence $\mathbf{s}$. We can compute this average over the exponential number of sequences efficiently using dynamic programming, see Appendix A.

We have introduced two new statistics for an observation $\mathbf{o}$: the number of unknown words $c_w(\mathbf{o})$ and the expected number of unsupported trigram transitions $\langle c_{tri}(\mathbf{o})\rangle$. In order to summarily express a single score for the number of unsupported model parameters, we combine these two scores by giving them equal weight, since we cannot know a priori which one might be more important. We combine scores in the following way. First, we normalise scores of both types such that they sum to one. Then, we take the arithmetic mean of the scores. (See Table 5.1 for an example.)

5.4.1 Experiments

First, we compare the count-based condition (CNT) against standard uncertainty sampling (UNC) under two different initial training set sizes of 100 and 1000 sentences in Figures 5.11 and 5.12 respectively. Count-based sampling consistently and significantly outperforms random sampling, as does uncertainty sampling. Count-based sampling is as good as uncertainty sampling when starting with 100 sentences with no significant differences apart from an initial dip of count-based sampling. When starting at 1000 sentences, count-based sampling is marginally better than uncertainty sampling after ca. 3.5k sentences have been sampled. However, this improvement is significant only in some phases of the learning curve.

Are selected examples different? This raises the question whether we can meaningfully combine uncertainty sampling and the count-based method. For this to be the case, we expect the methods to be complementary, such that they select different examples. To this end, we conduct the following experiment. We train a model on 1000 sentences, and apply it to a pool of another 1000 sentences. For each sentence in the pool, we record two scores, i) average entropy (as in uncertainty sampling) and ii) count-based method. In other words, this set-up mimics the first round of sample selection when starting at 1000 sentences. We find a Pearson coefficient of 0.2, indicating only a small correlation. In fact, when using these scores to select two batches of 100 examples each, we find an overlap of only 22 examples. In summary, we find that score types are sufficiently different to warrant investigating their combination.

A Hybrid Method Combining scores of different methods directly can be problematic. Scores might be arranged on different scales, for example one method may produce scores between 0 and 100, whereas the other method's scores may be bounded between 0 and 1. For such cases, we could use a simple linear weighting scheme. However, such a linear combination cannot accommodate situations when scores grow in different orders, for instance linear versus exponential. A simple but very effective solution is based on the observation that the main purpose of a score in active learning is to impose a ranking over the examples to be sampled from the pool. This allows us to determine the ranks of an example under both methods and then average these ranks. For the following experiment we use such an averaged rank-based combination method.

We present a comparison of the combined method UNC+CNT with uncertainty

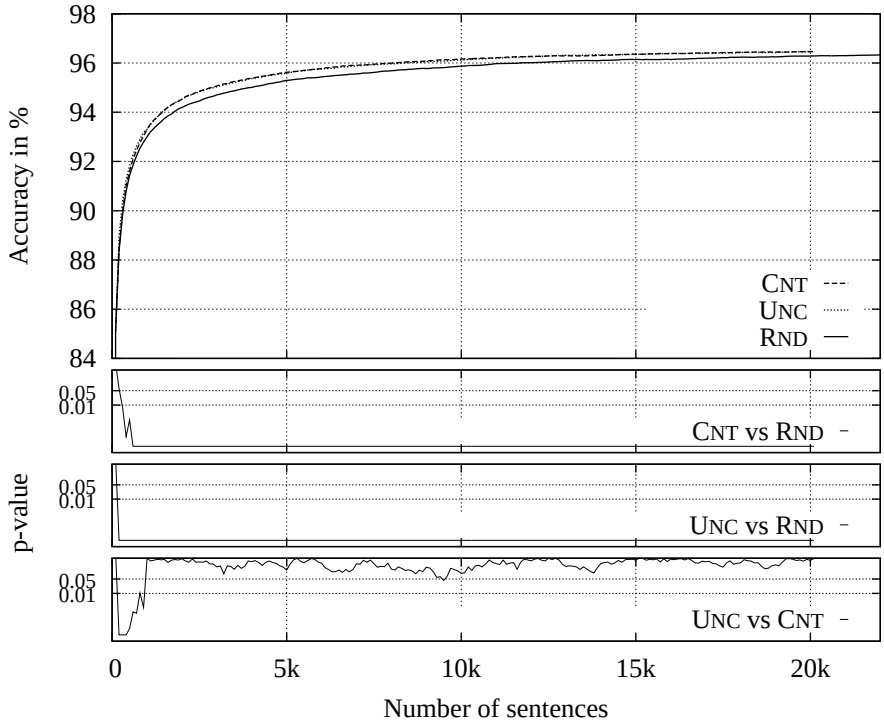

Figure 5.11: Comparing count-based method with uncertainty sampling (from 100 sentences)

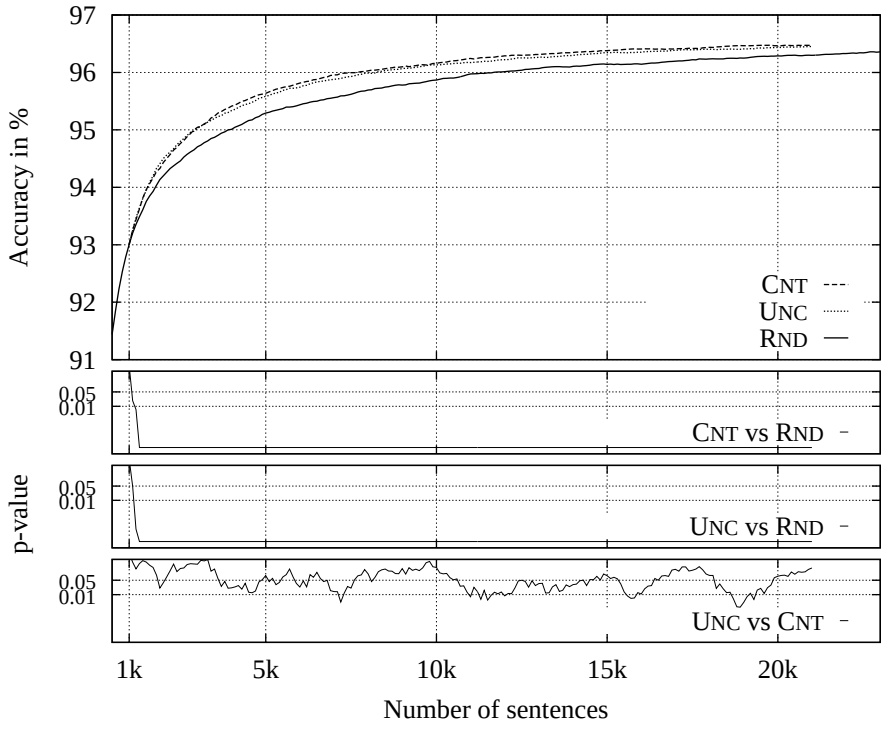

Figure 5.12: Comparing count-based method with uncertainty sampling (from 1000 sentences)

sampling UNC in isolation in Figures 5.13 and 5.14. UNC+CNT is significantly better than random sampling. When starting with 100 sentences significant improvements over UNC show from 500 to ca. 10k sentences and again towards the end of the learning curve. When starting with 1000 sentences, improvements are significant throughout after ca. 3k sentences have been sampled.

These findings present an improvement over uncertainty sampling as well as over the count-based method in isolation, and we can conclude that the combination of the two methods is indeed beneficial.

Finally, we should compare the combined method with the QBC method which we have found previously to be the best. We present results in Figures 5.15 and 5.16. Results are virtually indistinguishable in when starting with 100 sentences with no significant differences between the methods. Starting with 1000 sentences, there are almost no significant differences early on. Only after 15k sentences have been sampled is there a temporary but significant drop in performance.

5.4.2 Summary

The newly proposed count-based method is as good or better than uncertainty sampling for part-of-speech tagging. Furthermore, a combination of count-based sampling with uncertainty sampling can perform (almost) as well as the best QBC method. This presents more support for the on-going thesis that explicitly addressing model deficiencies helps to improve active learning.

5.5 Summary – Part-of-Speech Tagging

So far in this chapter, we have given an overview of active learning methods in application to part-of-speech tagging as a prototypical sequencing task. Confirming our findings from the previous chapter on prepositional phrase attachment, and as expected from previous work in the literature, we found that uncertainty sampling performs better than random sampling.

In order to see if QBC can outperform uncertainty sampling in this domain as well, we optimised relevant experimental parameters for QBC, such as sampling method, divergence metric and ensemble size. We found that a combination of bagging and vote entropy works best and performs significantly better than uncertainty sampling. Using a larger ensemble does not yield significantly better results.

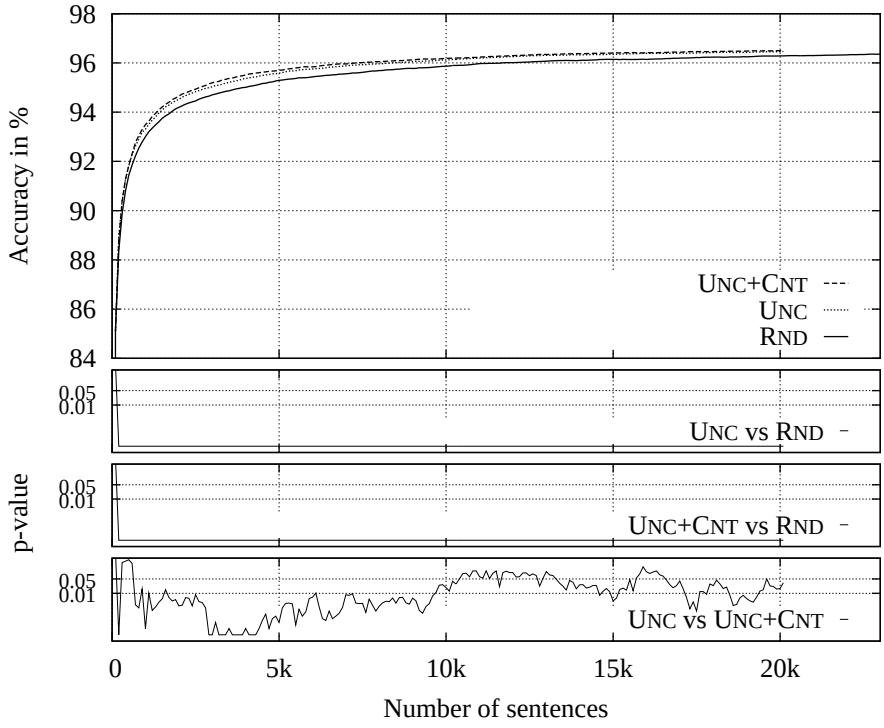

Figure 5.13: Comparing hybrid method with uncertainty sampling (from 100 sentences)

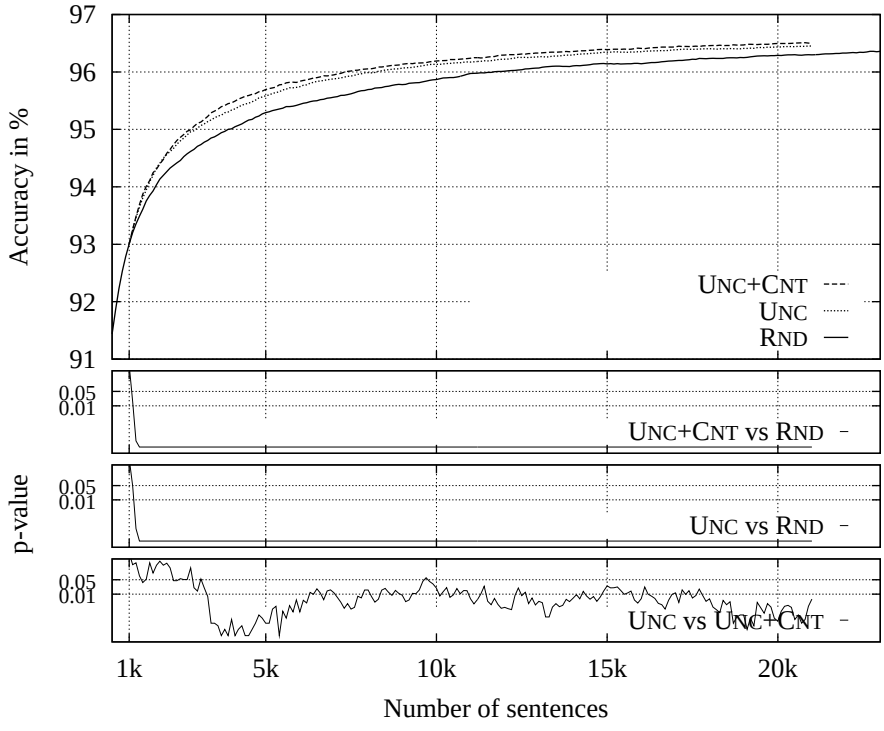

Figure 5.14: Comparing hybrid method with uncertainty sampling (from 1000 sentences)

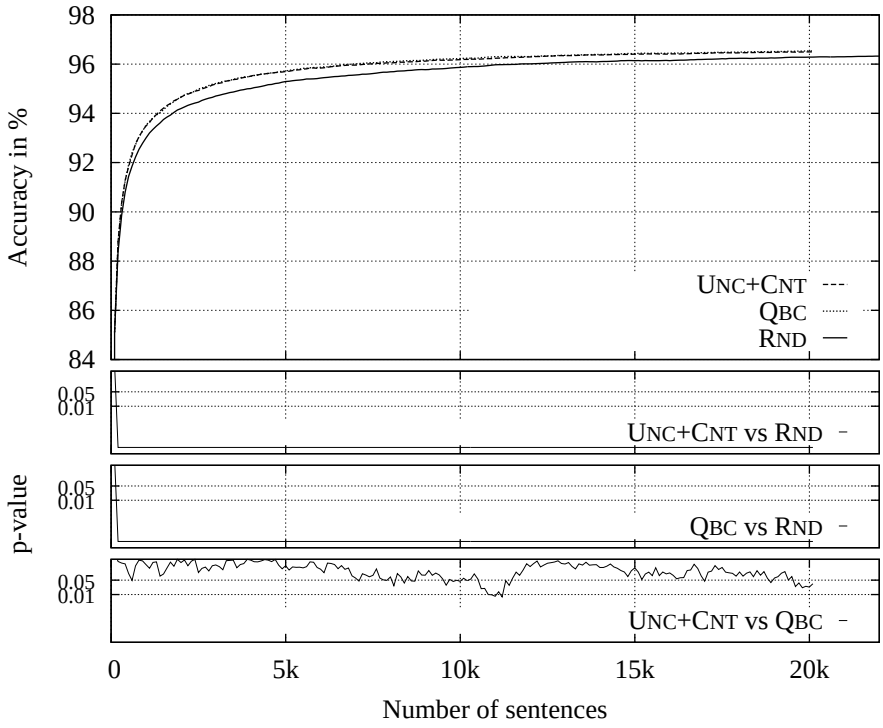

Figure 5.15: Comparing hybrid method with best QBC (from 100 sentences)

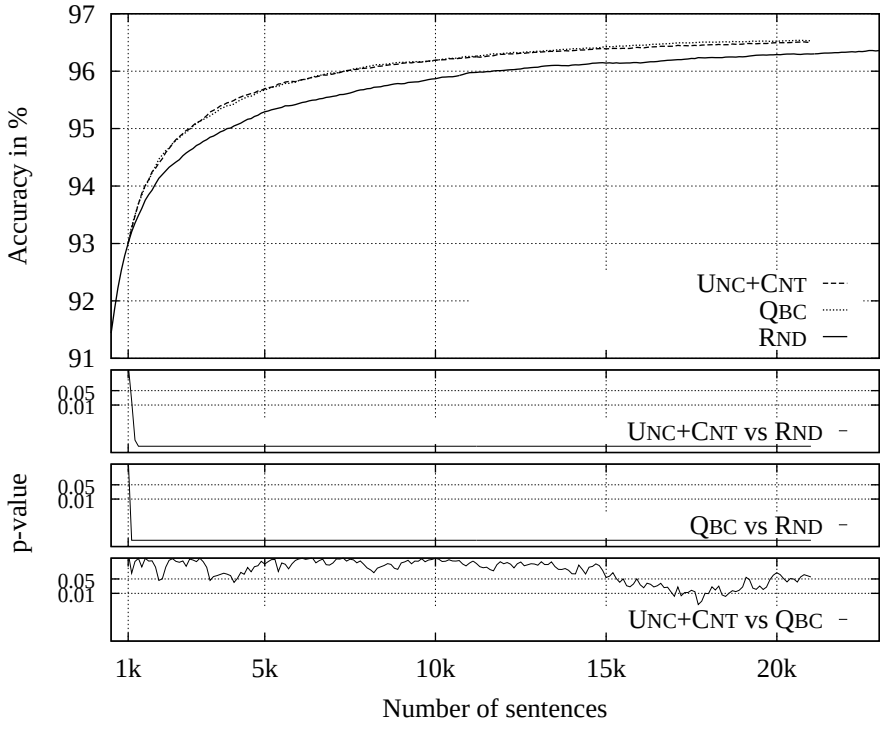

Figure 5.16: Comparing hybrid method with best QBC (from 1000 sentences)

We introduced a novel count-based method which counts the number of unknown words in a sentence and the expected number of unsupported transitions used in decoding. This method by itself is as good or better than uncertainty sampling. Furthermore, it tends to select examples which are different from uncertainty sampling. This observation suggests the combination of both methods. The combined method performs better than either method in isolation, and its performance is almost indistinguishable from best QBC performance.

5.6 Active Learning for Named Entity Recognition

To see how far the findings for part-of-speech tagging generalise, we consider named entity recognition as another sequencing task. NER and part-of-speech tagging differ in a number of important aspects. Most importantly, NER is a combined segmentation and classification task, as opposed to the tokenwise classification in tagging. Nevertheless, we can treat NER as a sequencing problem by adopting the BIO-markup scheme (Ramshaw and Marcus, 1995).

A direct ramification of the difference between tagging and NER is that performance in NER is usually reported as f-measure, whereas we used accuracy to measure performance for tagging. Another difference concerns the number of different labels. This of course may vary strongly even within tagging or NER, depending on the data set. At any rate, while there are 45 labels in the tagging task, the size of the label set is considerably smaller in NER. Assuming a task with four named entity types – ORG, LOC, MISC, and PER – we arrive at a potential set of 9 labels, B- and I- labels for each of the entity types and an additional O label.[2]

We present a summary of results for uncertainty sampling, the count-based method and the QBC using bagging and vote entropy in comparison with random sampling in Figures 5.17 and 5.18.[3] We note that all active learning methods outperform random sampling. Interestingly, uncertainty sampling outperforms all other methods by a wide margin. This demonstrates that uncertainty sampling can well be an effective method to improve unreliable parameters. However, this is in contrast to the relative performance of uncertainty sampling compared to other active learning methods for all other applications detailed in this thesis.

[2] In fact, one label is missing from this set, B-PER.

[3] We do not show results for combined count-based/uncertainty sampling which is better than count-based but worse than uncertainty sampling by itself. Also not shown here and in the following experiments are results for other QBC settings which all perform worse than bagging/vote entropy.

As is well known from the literature, relative performance of active learning methods can vary across tasks. Baram et al., 2004 suggest a meta-learning protocol which dynamically chooses among several sample selection methods according to a maximum entropy criterion. They found that this method almost always matches the performance of the best method for a given task. Still, it would be good to know why uncertainty sampling performs so well for NER. In general, it is desirable to have a criterion by which to predict the relative performance of active learning methods applied to a particular problem. This is in particular the case when having to choose a method for a novel type of application.

As pointed out above, there are differences between NER and tagging with respect to i) the choice of evaluation metric, ii) the size of tagset, and more generally iii) the type of task. In order to see if relative performance of methods may depend on the choice of evaluation metric, we evaluated the above-mentioned range of active learning methods for NER according to label accuracy. We find the same relative ranking for the methods, with uncertainty sampling still performing the best. Hence, we can discard evaluation metric as a potential explanation for these differences. This leaves tagset size and task type as potential factors. So far, we examined results for tagging with a large tagset and NER with a small tagset. In order to tease apart these factors, we introduce a fully-crossed design, where we carry out the two following additional experiments:

- Decrease size of label set for part-of-speech tagging
- Increase size of label set for NER

Simplified Part-of-Speech Tagging

To decrease the size of the label set for part-of-speech tagging, we collapsed the original set of 45 different part-of-speech tags to 5 labels as in (McCallum et al., 2003):

- Collapse all different types of nouns into one label NOUN.
- Collapse all different types of verbs into one label VERB.
- Collapse all different types of adjectives into one label ADJ.
- Collapse all different types of adverbs into one label ADV.
- Collapse the remaining POS labels into one label OTHER.

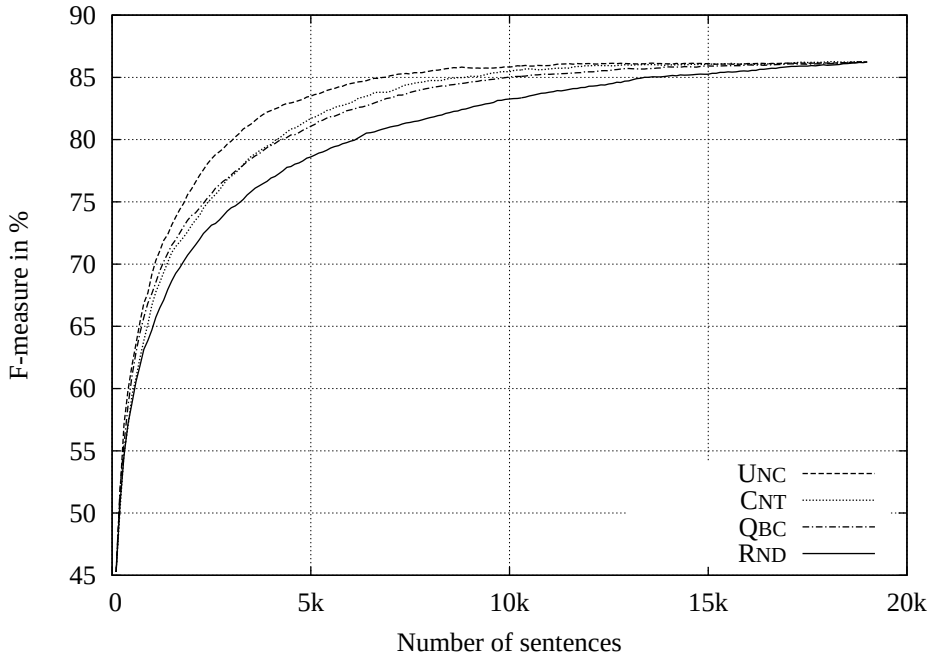

Figure 5.17: Overview of methods for NER (from 100 sentences)

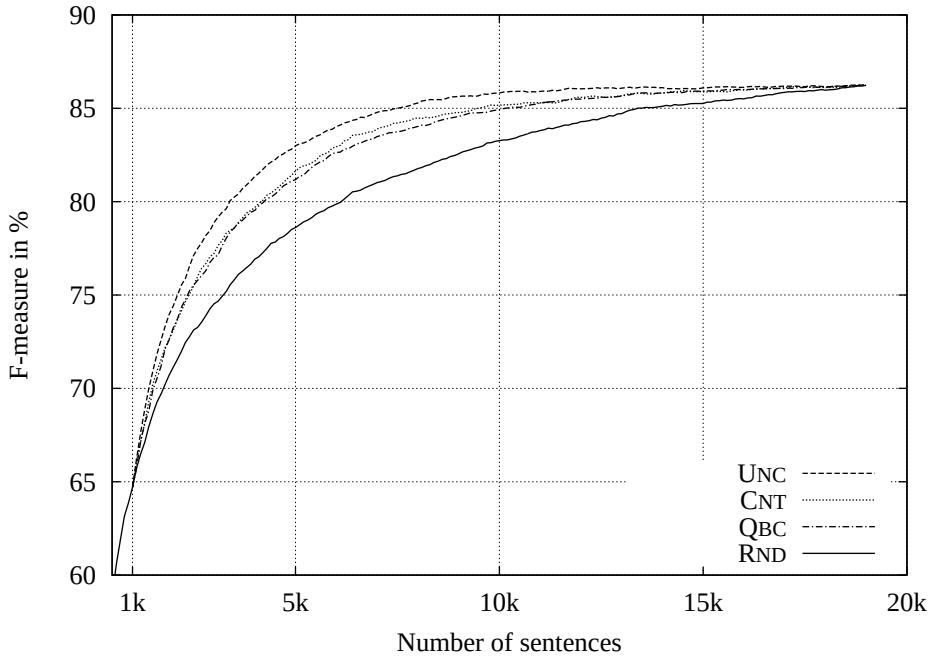

Figure 5.18: Overview of methods for NER (from 1000 sentences)

We present results for the same set of active learning methods in application to simplified part-of-speech tagging in Figures 5.19 and 5.20. Qualitatively, results are very similar to named entity recognition (with a small tagset). Uncertainty sampling outperforms all other methods by a wide margin while count-based and QBC perform quite similarly. This shows again that uncertainty sampling can be effective to deal with unreliable parameters under certain circumstances. This is also an indication that the relative performance of active learning methods may be related to the size of the tagset rather than the nature of the task.

Named Entity Recognition with Increased Label Set

In order to increase the size of the label set for NER, we use a more involved coding for outside tokens (previously labelled O). For these, we indicate the type of the tokens which surround them in a context two tokens to the left and two to the right. Contextual tokens within an entity are coded N; outside an entity O; sentence boundaries are coded X. For example, the sequence "BRUSSELS/I-LOC 1996-08-22/O" is now labelled as "BRUSSELS/I-LOC 1996-08-22/XN-O-XX". Using this coding, we increase the label set to 56 tokens, while performance remains roughly the same as when using the standard BIO set when using random sampling.

We show results in Figures 5.21 and 5.22. The previous advantage of uncertainty sampling over other methods has disappeared now and QBC is about as good as uncertainty sampling for this task. The performance of count-based sampling is only marginally better than random sampling. This is probably because considerable annotation effort is directed towards annotating the multitude of different O labels. Arguably, this is an artifact of the contrived tagset.

Again, we can see a clear dependence on the size of the tagset for the relative performance of different active learning methods.

Summary

For NER, we found that the count-based method and QBC perform roughly equally well. However, uncertainty sampling achieves by far the best results. This contrasts with the results for part-of-speech tagging in this chapter, and results in the chapters on prepositional phrase attachment and on parsing. Generally, however, this coincides with the perceived wisdom that there is no single best active learning method across all applications (Baram et al., 2004). The results of a cross-designed experiment with

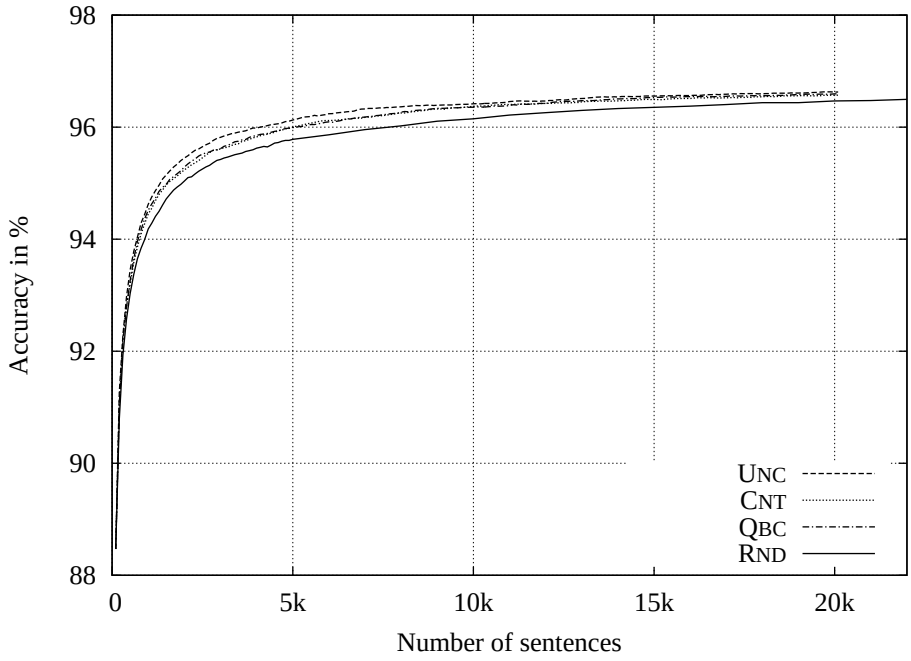

Figure 5.19: Overview of methods for simplified tagging (from 100 sentences)

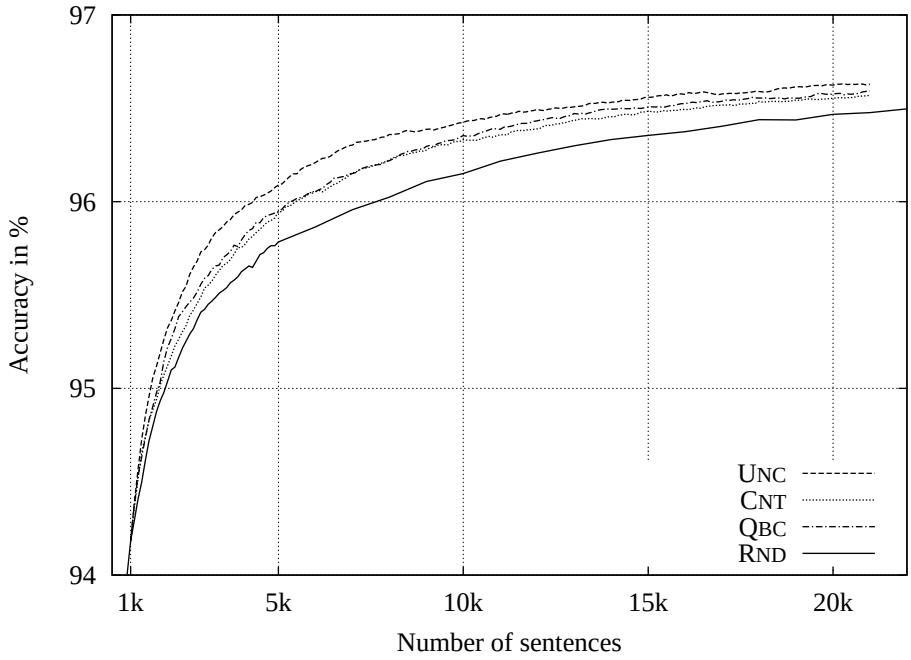

Figure 5.20: Overview of methods for simplified tagging (from 1000 sentences)

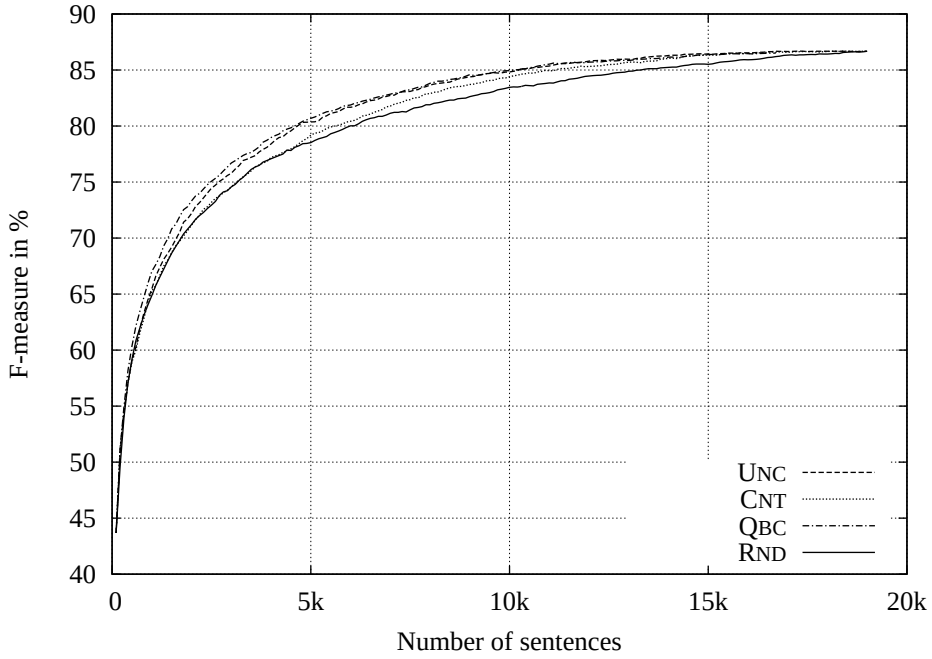

Figure 5.21: Overview of methods for NER with an extended tagset (from 100 sentences)

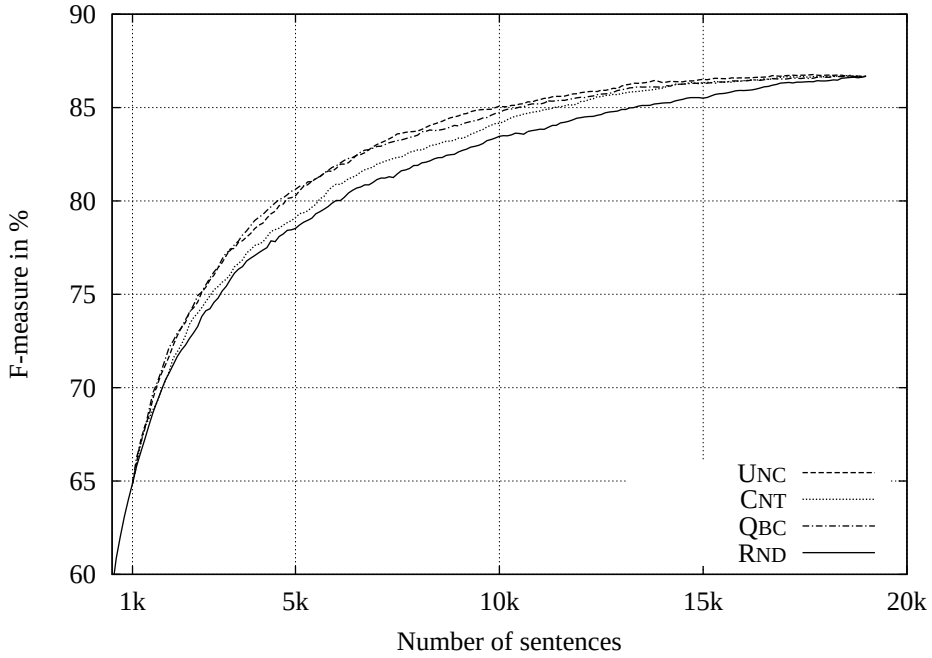

Figure 5.22: Overview of methods for NER with an extended tagset (from 1000 sentences)

tagset size and task type as independent factors indicate that the tagset size is the main reason behind this finding.

5.7 Conclusion

In this chapter, we demonstrated the need to specifically address unreliable parameters when applying active learning to sequencing tasks. In the first part, we focused on part-of-speech tagging as a task. We found that uncertainty sampling outperforms random sampling and that we can do even better than uncertainty sampling by using QBC. However, in order to make QBC perform properly requires adjusting experimental parameters. We found best performance by using a combination of bagging and vote entropy.

Furthermore, we introduced a novel method which directly counts the number of unreliable parameters based on missing events which are needed for decoding a given sentence. This method in isolation is as good as or better than uncertainty sampling. This again demonstrates that directly addressing unreliable parameters is a successful strategy. Furthermore, we established that it tends to select examples which are different from the examples that uncertainty sampling would select. This situation suggests the combination of the count-based method with uncertainty sampling and, in fact, the combination beats both methods in isolation and almost matches the best overall result from QBC.

We found surprising results when applying this range of methods to named entity recognition. In this domain, best results are achieved with uncertainty sampling. Experiments indicate that uncertainty sampling works well on problems with small tagsets. At least in some domains, uncertainty sampling may be suited to deal with unreliable parameters as well as other methods investigated in this thesis.

Chapter 6

Unreliable Parameters in Parsing

In the previous two chapters, we have demonstrated improvements over standard active learning methods by specifically selecting examples of which the analysis depends on unreliably estimated model parameters. We applied such techniques to prepositional phrase attachment and to sequencing tasks. In this chapter, we show that we can achieve similar improvements for parsing as well; using the same strategy of preferably annotating examples associated with unreliably estimated parameters.

As with the previous applications, the supervised training of probabilistic parsers (Collins, 1997; Charniak, 2000) requires large amounts of manually annotated material to achieve high performance levels. Syntactically annotated corpora are available for a variety of languages, for instance the Penn Treebank for English (Marcus et al., 1993), the Negra Treebank for German (Skut et al., 1997) or the Alpino Treebank for Dutch (van der Beek et al., 2002). However, for the many languages which as yet lack treebanks, active learning holds the promise to significantly reduce annotation costs.

Parsing is an interesting application for active learning. It presents a more complex task than prepositional phrase attachment or sequencing since labels are not only assigned to individual words, but to recursively embedded constituents. Also, the use of bilexical dependencies in lexicalised parsers causes sparse data problems which are more severe than for sequencing models. We expect our approach to be particularly useful for such cases where distributions have long tails of infrequent rules.

For the experiments in this chapter, we use a near state-of-the-art lexicalised parser, Collins' model 2 parser (Collins, 1997), which we described in more detail in Subsection 2.3.3. In particular, we use Dan Bikel's implementation described in (Bikel, 2004a). For the training of Collins' parsing model, manually annotated parse trees are decomposed into head- and modifier-generation events. The parsing performance of a

trained model will suffer in particular from missing or infrequently observed events.

Sentences are unparsable usually because of *missing events* in the grammar.[1] This suggests the selection of unparsable sentences to learn such missing events. The acquisition of the correct parse tree for an unparsable sentence trivially enables its analysis. More importantly, learning new parsing events augments the parsing model in general and should help the analysis of similar sentences, and thus increase coverage. In one set of experiments, we investigate this strategy to increase coverage in combination with standard uncertainty sampling. Results show that it is always beneficial to select unparsable sentences for annotation. At least in the early phases of training, we show that striving for coverage is more important than selecting uncertain examples. By contrast, we show that ignoring unparsable sentences in uncertainty sampling results in a performance worse than random sampling.

Parameter estimates based on *infrequently observed events* will typically exhibit a higher degree of variance and contribute to a higher error rate. Hence, acquiring the correct analyses for sentences where the analysis is based on such infrequently observed events will eventually increase low event counts in the model and thereby reduce variance and error rate. We identify such sentences by training the model on a bootstrap replicate of the original training set. This has the effect of (stochastically) eliminating some, but not all infrequent events. Clearly, the analysis of a sentence which becomes unparsable under such a bagged model has been based on a low frequency event. We note here that the eliminated analysis might not even have been the correct one. In either case, as a positive effect, acquiring the correct parse tree for such a sentence either increases some of the low frequency events or introduces new events. This approach is essentially an approximation to the method discussed in the previous chapter in order to target infrequently observed events. We conduct experiments with a two-stage method which first selects unparsable sentences according to a bagged parser, and then applies uncertainty sampling to the remaining sentences using a fully trained parser, and contrast these with a more conventional ensemble-based approach where the ensemble members have been created by the same method, namely by training on a bootstrap replicate of the original training set. Both methods clearly outperform uncertainty sampling. Furthermore, the simple two-stage method performs as well or better than the more involved ensemble-based method.

This chapter extends joint work with Miles Osborne, previously published as (Becker and Osborne, 2005).

[1] Other potential causes for parse failures are memory problems or time-out conditions.

Chapter Structure

We begin in Section 6.1, by showing the importance of coverage in parsing in a random sampling experiment. By using an appropriate smoothing mechanism, in the form of *constraint relaxation*, we improve coverage and other relevant evaluation metrics.

In Section 6.2, we demonstrate clear improvements over standard uncertainty sampling by specifically selecting unparsable sentences. This actually requires the switching off of constraint relaxation in the sample selection phase in order to let the parser fail on difficult sentences. We show that this method combines gracefully with other methods such as uncertainty sampling.

In Section 6.3, we investigate relevant parameters for QBC, in particular the choice of divergence metric and ensemble size. As with uncertainty sampling, we find improvements for parser ensembles which do not use constraint relaxation.

In Section 6.4, we introduce a novel method which targets unreliable parameters due to infrequent counts.

We conclude this chapter in Section 6.5.

6.1 Parsing Coverage in Random Sampling

Achieving high coverage clearly is important for a good parsing performance. Low coverage in a parser is directly reflected in low recall and thus also in a low f-measure. Increasing the coverage of a parser will generally be beneficial for recall, even if the predicted parse trees are not entirely correct, since we can expect at least partial credit for some of the constituents, assuming a Parseval evaluation. We show in this section how an appropriate smoothing regime can help to increase coverage. In the Bikel parser, smoothing occurs in a number of different forms, for example back-off smoothing for probabilities with unobserved conditioning contexts. One form of smoothing called *constraint relaxation* has been implemented in Bikel's parser which replaces all zero probability estimates with small probabilities for sentences which cannot be parsed with the default parsing scheme at the maximum beam width.

Experiment

In this experiment, we examine the effects of applying Bikel's constraint relaxation on coverage and other relevant evaluation metrics. Here, we will consider a random sampling setting, that is, apart from constraint relaxation, we do not consider any other

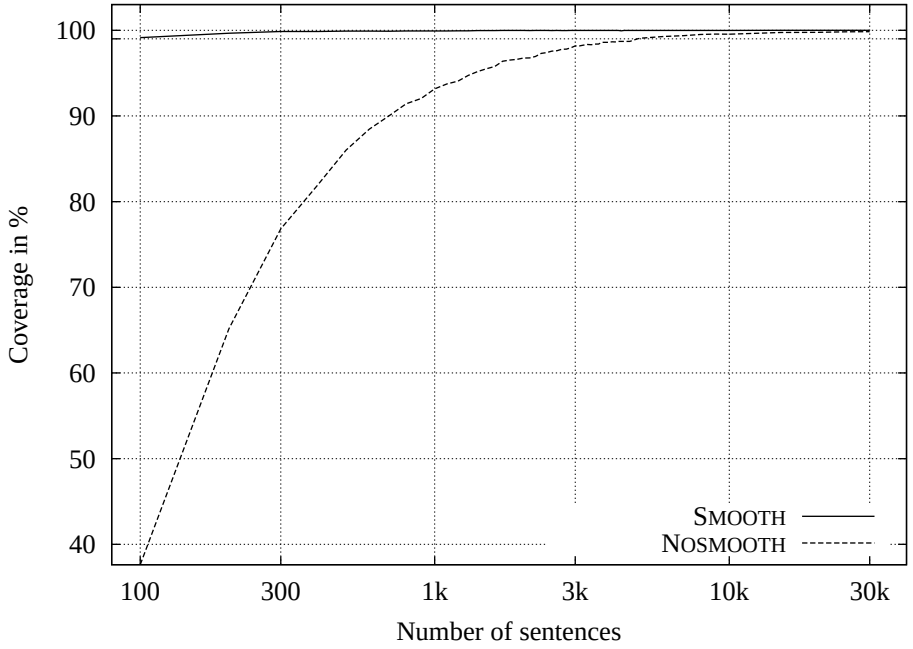

Figure 6.1: Coverage of a parser with and without constraint relaxation

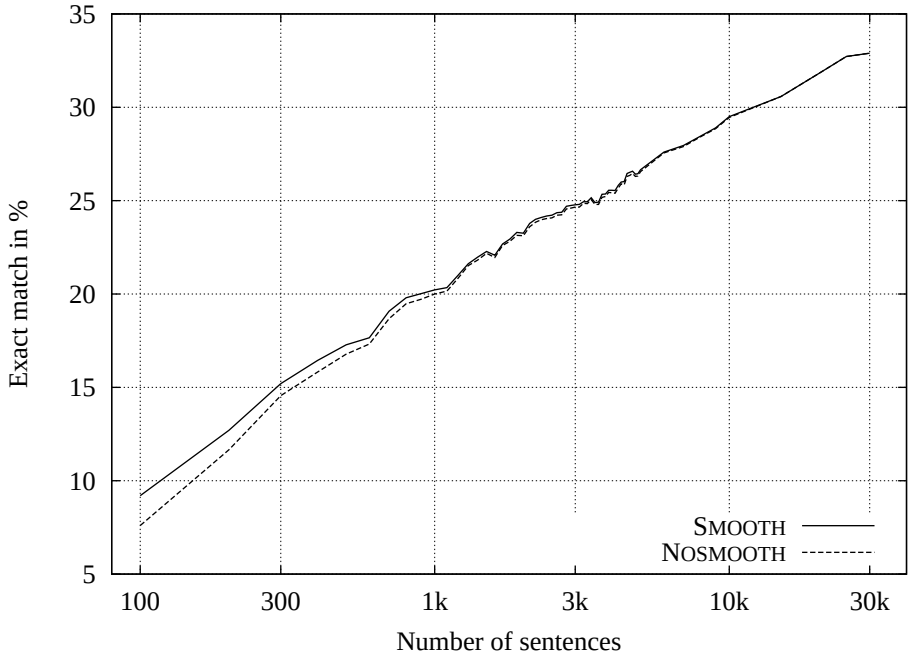

Figure 6.2: Exact match rate of a parser with and without constraint relaxation

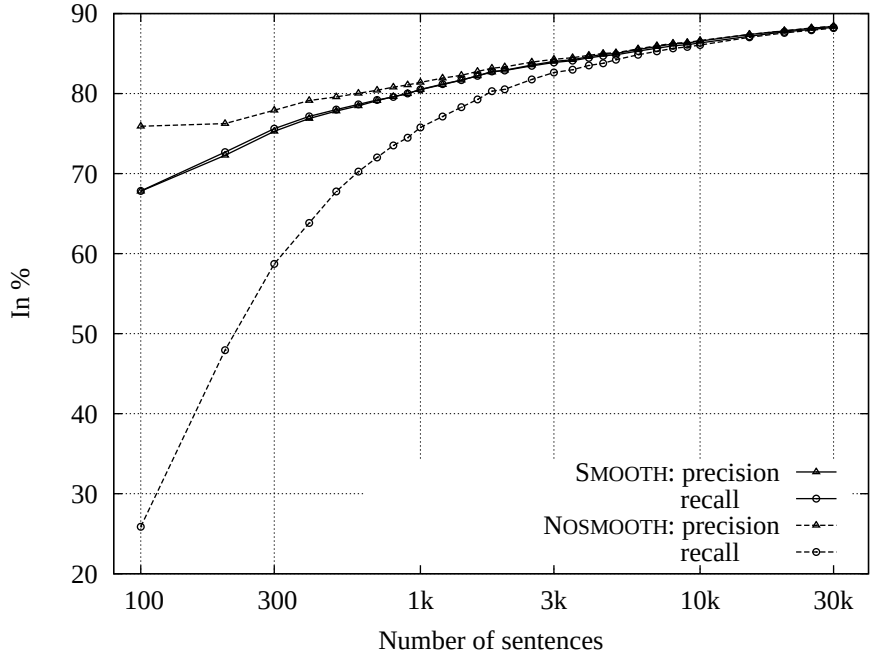

Figure 6.3: Precision and recall of a parser with and without constraint relaxation

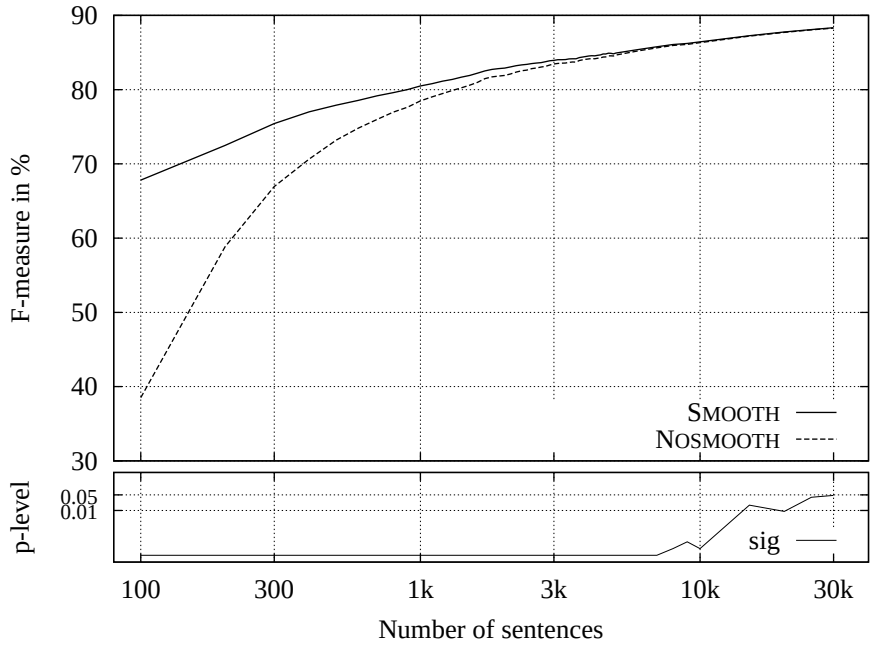

Figure 6.4: F-measure of a parser with and without constraint relaxation

means to increase coverage. We record learning curves for randomly sampled training sets of different sizes for coverage, exact match rate, precision, recall, and f-measure, both for a parser with constraint relaxation, and for a parser without.

Figure 6.1 gives results for coverage. A parser smoothed with constraint relaxation consistently has more than 99% coverage, even for very small training sets. By contrast, a parser without constraint relaxation shows much smaller coverage for small training sets, but eventually converges towards full coverage for a large training set. However, a level of 99% coverage is reached only after ca. 4,900 annotated sentences have been seen.

The effect of applying constraint relaxation is less drastic for the exact match rate, see Figure 6.2. A parser with constraint relaxation performs only slightly better than one without. This is somewhat to be expected, since the chances are fairly small of correctly predicting the entire structure for those sentences which could not be parsed without constraint relaxation and obviously are difficult.

Figure 6.3 gives precision and recall learning curves for both parser types. Recall and precision are fairly high for a parser with constraint relaxation, even with very small training sets. Also, recall and precision are almost completely balanced throughout. As predicted analytically, recall for a parser without constraint relaxation is severely impaired due to a lack of coverage. On the other hand, precision is higher than for the parser with constraint relaxation.

The net effect of drastically improved recall and slightly impaired precision can be seen in Figure 6.4. F-measure for the parser with constraint relaxation is significantly better throughout. The improvements are particularly striking for small training sets: With a training set of 100 sentences, f-measure goes up from 39% to 68%.

These results clearly demonstrate that employing constraint relaxation as a smoothing technique is beneficial as measured by a variety of relevant evaluation metrics, among them coverage, exact match rate, and f-measure. For the purpose of deploying the parser, that is, using it for testing purposes, we will henceforth use the parser with constraint relaxation.

6.2 Uncertainty Sampling for Parsing

The previous section has shown that smoothing in the form of constraint relaxation is beneficial when parsing test sentences. We stipulate the hypothesis that it might be beneficial to allow the parser to fail on difficult sentences in the pool by explicitly

switching off constraint relaxation in the selection phase, and to subsequently target such sentences for annotation. The acquisition of correct parse trees for unparsable sentences necessarily increases the size of the model structure for the grammar, and eventually may help to parse new sentences more reliably. Such a method may be combined with standard uncertainty sampling by selecting both unparsable sentences and parsable sentences with high entropy. These considerations motivate the following sample selection methods.

Coverage-based selection: (COV) Select out-of-coverage sentences based on an unsmoothed parser; fill the batch with random sentences. This method directly aims to acquire missing parsing events.

Coverage- and uncertainty-based selection: (COV-UNC) Select out-of-coverage sentences based on an unsmoothed parser; fill the batch with high-entropy sentences. By combining out-of-coverage and uncertainty sampling, this method also aims to acquire missing parsing events and to generally improve unreliable model parameters.

Uncertainty-based selection, unsmoothed: (UNC-NSM) Select high-entropy sentences based on an unsmoothed parser. This method generally aims to improve unreliable model parameters but, by implicitly dispreferring unparsable sentences, it will fail to acquire the missing parsing events associated with these sentences.

Uncertainty-based selection, smoothed: (UNC-SM) Select high-entropy sentences based on a smoothed parser. This method also generally aims to improve unreliable model parameters. It does not have a specific mechanism to acquire missing parsing events. It thus serves as a baseline to see if entropy by itself can reliably select useful sentences for annotation as conventionally done in active learning.[2]

Random selection: (RND) We compare all results against a parser trained on randomly sampled training sets of different sizes.

With larger training sets, unparsable sentences become less frequent, and the differences between the methods should be less pronounced. To investigate this effect, we start at initial training set sizes of 100 and 1000 sentences.

[2]Unparsable sentences will also be included preferably, even though they occur quite rarely when making use of constraint relaxation.

Sentence length restriction In this experiment, we will have to deal with an artefact of the implementation of the n-best parse enumeration in combination with smoothing. To compute the n-best parses, the implementation switches off dynamic programming which causes long parsing times in general. The application of constraint relaxation aggravates this problem, resulting in extremely long parse times (half hour per sentence and more) in particular for longer sentences. This requires an appropriate time-out threshold to keep experimental run-times feasible. Furthermore, memory requirements are very high for the combination of n-best enumeration and smoothing, resulting in occasional out-of-memory problems.

It is not immediately clear what to do with such sentences which suffer from either time-out or out-of-memory problems. One could ignore these sentences, and thereby deprive the smoothed parser of these examples. Given that this mostly happens for sentences which are unparsable for the unsmoothed parser, this seems like an unfair advantage for the unsmoothed parser. Alternatively, one could choose to include these sentences as selected examples just as we do with the unsmoothed parser, thereby blurring the distinction between the two approaches.

Given that we would actually like to contrast an approach which entirely relies on entropy scores with one that has the added possibility of flagging unparsed sentences due to coverage problems, we restrict the experiment by choosing training set and pool set only from sentences with length ≤ 30. This length threshold has been chosen since almost all sentences can be parsed without time-out or memory problems, even with n-best enumeration and constraint relaxation. For the test set, we use sentences with length ≤ 40.

6.2.1 Experiments

Experiment 1 We compare coverage-based selection (COV) to a baseline of smoothed uncertainty-based selection (UNC-SM). Using coverage-based selection will show how far out-of-coverage sampling can go by itself, without making use of uncertainty. We contrast both methods against random selection.

Figures 6.5 and 6.6 show results when starting with a small training set of 100 sentences and a large training set of 1000 sentences respectively. We find that both methods perform significantly better than random sampling for most of the iterations. There is however a clear difference between the two methods. The increase in f-measure is slow initially for standard uncertainty sampling, but after the first few iterations the

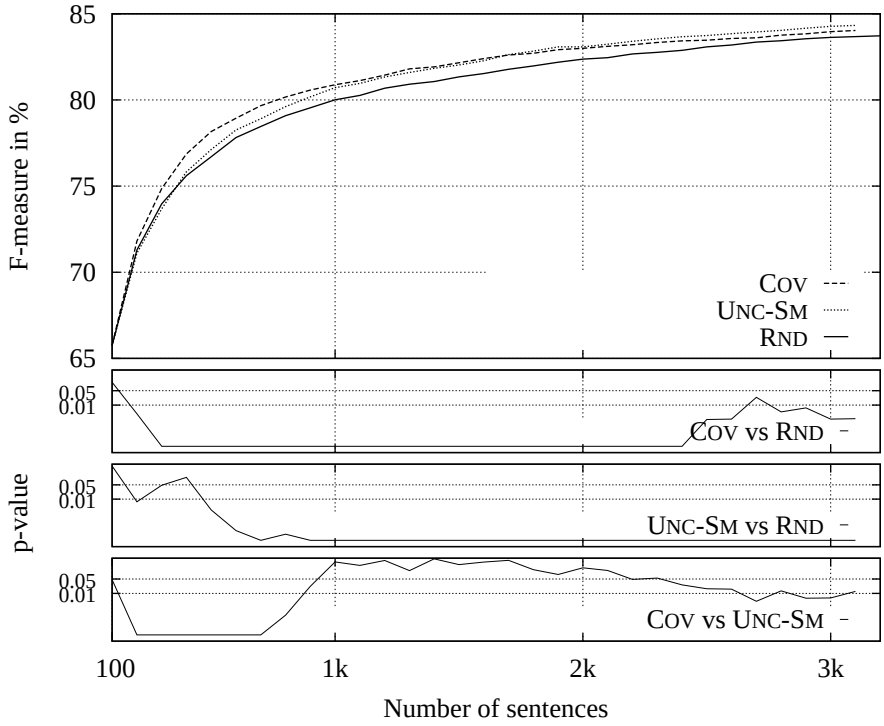

Figure 6.5: Comparing coverage-based selection with smoothed uncertainty-based selection (from 100 sentences)

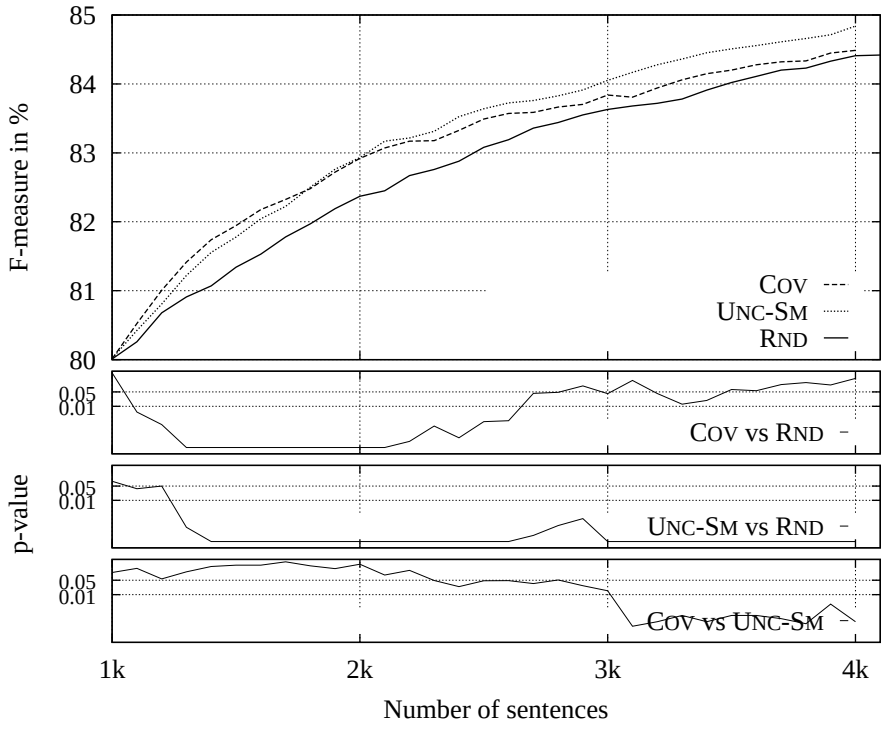

Figure 6.6: Comparing coverage-based selection with smoothed uncertainty-based selection (from 1000 sentences)

improvement over random sampling is significant throughout. The coverage-based method on the other hand has a remarkably steep rise in f-measure and is significantly better than random sampling after only a single iteration. However, this performance is not sustained through the experiment, and in later iterations f-measure converges towards random sampling.

In a direct comparison between the two methods, we find that the coverage-based method is significantly better than standard uncertainty sampling until 900 sentences have been annotated when starting with a small training set, and significantly worse only after ca. 2200 sentences. When starting with a large training set, the coverage-based method is as good as standard uncertainty sampling (or better) until ca. 2300 sentences.

This demonstrates that pursuing unparsable sentences to boost coverage is beneficial in sample selection at least in the early phases of active learning. When coverage converges to 100% however, such a method looses its impact and sample selection is increasingly based on random sampling.

Experiment 2 In the next experiment, we compare coverage- and uncertainty-based selection (COV-UNC) to the baseline of smoothed uncertainty-based selection (UNC-SM as above). The motivation for the coverage- and uncertainty-based selection method is that we would like to harness the advantages of early out-of-coverage selection and have a sustainable selection method for later phases. Results for this comparison are shown in Figures 6.7 and 6.8, again starting with a small and a large training set.

Both conditions consistently outperform random sampling in terms of f-measure for both starting points. However, the increase using smoothed uncertainty-based selection is slower and significantly better than random sampling only after a few iterations. By contrast, f-measure increases faster using coverage- and uncertainty-based selection and is significantly better after only a single iteration for both initial training set sizes.

When comparing both conditions against each other, we find that coverage- and uncertainty-based selection is consistently better than smoothed uncertainty-based selection. When starting at 100 sentences, this improvement is significant throughout all iterations. When starting at 1000 sentences, results are a little less pronounced and significance is achieved only in some iterations.

The difference between the two methods is that we turn off constraint relaxation in the sample selection stage. Apart from improved performance in terms of f-measure,

this has the added advantage that the sample selection phase can run considerably faster. When using a grammar trained on 100 sentences, parsing a pool of 1000 sentences takes two minutes with an unsmoothed parser, but more than seven hours with a smoothed parser. When using a grammar trained on 1000 sentences, parsing a pool of 1000 sentences takes 18 minutes with an unsmoothed parser, and more than two hours with a smoothed parser.

Experiment 3 Finally, we might ask what happens if we used a purely uncertainty-based selection in combination with an unsmoothed parser (UNC-NSM). This method implicitly disprefers unparsable sentences; hence we expect it to perform poorly. We contrast this method with the already discussed coverage- and uncertainty-based selection which prefers unparsable sentences (COV-UNC) in Figures 6.9 and 6.10.

The most striking result is found when starting with the small initial training set. F-measure for purely uncertainty-based selection is consistently worse than random sampling and significantly so from the first iteration until 1900 sentences have been sampled. Correspondingly, this method is also significantly worse than coverage- and uncertainty-based selection in all iterations. When starting from 1000 sentences, purely uncertainty-based selection increases at the same rate as random sampling in the first few iterations, and is significantly better than random sampling only after the third iteration. It is significantly worse than coverage- and uncertainty-based selection until 2900 sentences. This clearly demonstrates that failing to select unparsable sentences is harmful.

Uncertainty Sampling alone cannot reliably identify difficult sentences

The previous experiments demonstrated that the selection of unparsable sentences (with an unsmoothed parser) is beneficial. Also, we saw that selection using the entropy of a smoothed full-coverage parser alone is not a sufficient replacement for this simple method. Admittedly, this would be more elegant since we would run the parser with the same set of settings regardless of whether our purpose is testing or sample selection; and we use entropy as the single selection criterion.

This indicates that difficult unparsable sentences do not have a higher entropy once they are made parsable through smoothing. To verify this fact, we ran an experiment in which we recorded entropies of 1000 sentences when parsed with grammars trained on 100 and 1000 sentences. We show results in Table 6.1. With the small training set, 554 sentences require constraint relaxation to be parsable at all. Nevertheless,

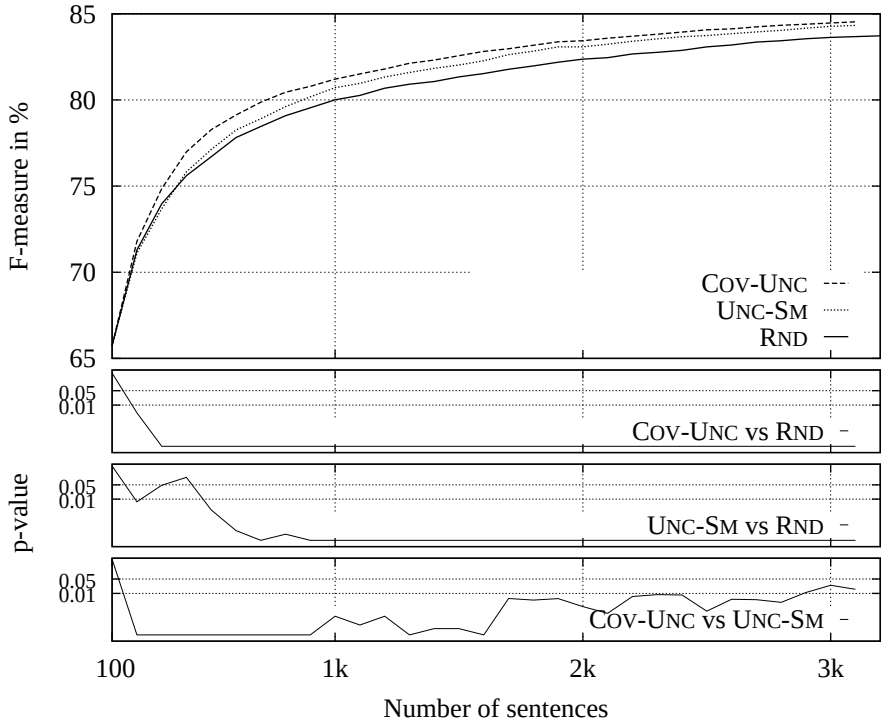

Figure 6.7: Comparing coverage- and uncertainty-based selection with smoothed uncertainty-based selection (from 100 sentences)

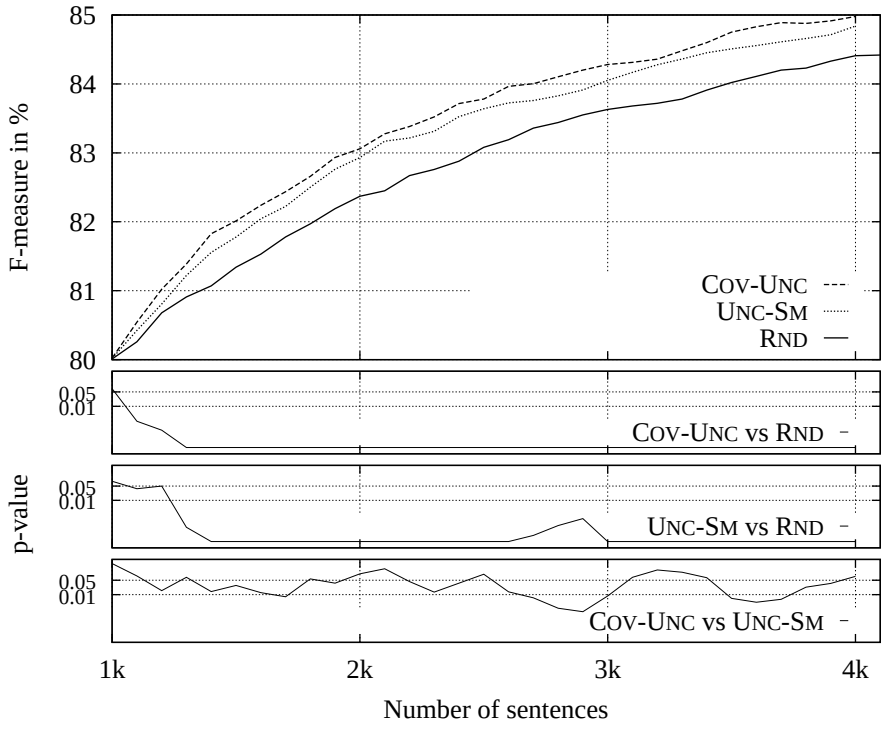

Figure 6.8: Comparing coverage- and uncertainty-based selection with smoothed uncertainty-based selection (from 1000 sentences)

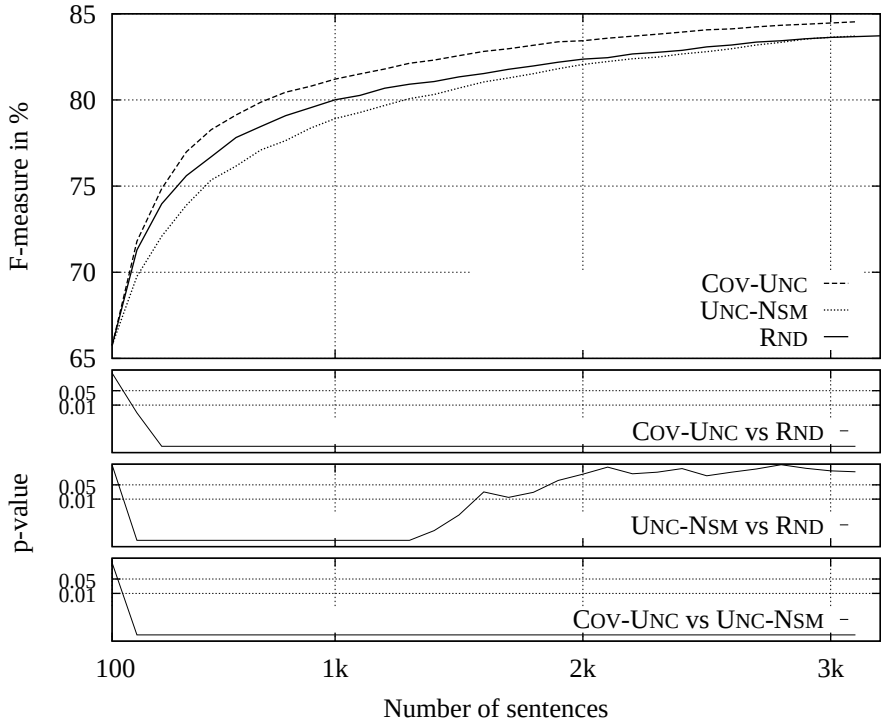

Figure 6.9: Comparing coverage- and uncertainty-based selection with unsmoothed uncertainty-based selection, excluding unparsed sentences (from 100 sentences)

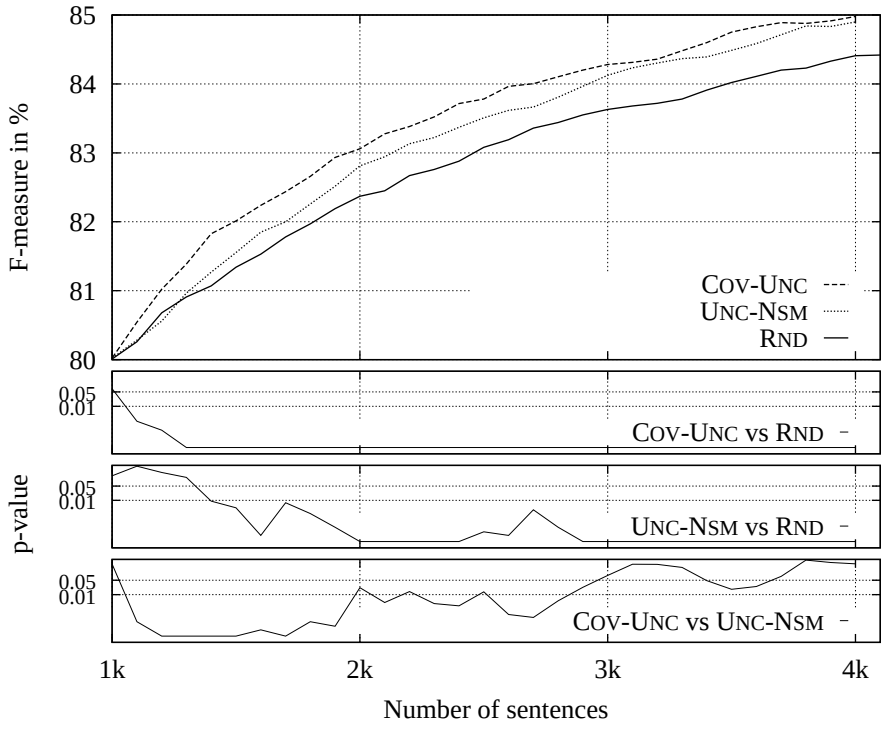

Figure 6.10: Comparing coverage- and uncertainty-based selection with unsmoothed uncertainty-based selection, excluding unparsed sentences (from 1000 sentences)

	average entropy		number of instances		
#sentences	smoothed	unsmoothed	#smoothed	#unsmoothed	(other)
100	0.47	0.45	554	432	(14)
1000	0.52	0.54	57	937	(6)

Table 6.1: Average entropies for easy and difficult examples

the average entropy for these sentences (0.47) is only marginally higher than for the 432 sentences which do not require smoothing (0.45). 14 sentences were not parsable at all due to time-outs or genuine unparsability. With the large training set only 57 sentences required constraint relaxation to be parsable. Their average entropy (0.52) is even lower than for the large majority of sentences which required no smoothing. Here, 6 sentences were not parsable even with constraint relaxation.

In short, entropy is not a good indicator for the selection of difficult sentences which can only be parsed with a smoothed parser. This clearly supports the notion that an uncertainty-based sample selection method should also use parsability information.

6.2.2 Summary

We have looked at aspects of selecting out-of-coverage sentences in the context of standard uncertainty sampling.

Only selecting out-of-coverage sentences from the pool, without any further sources of information, is a surprisingly effective method to increase f-measure at a higher rate than random sampling, at least in the early rounds of active learning. This method implies switching off the standard smoothing in order to identify unparsable sentences. On the other hand, failing to select out-of-coverage sentences, even when uncertainty sampling is employed, is detrimental and can result in performance worse than random sampling.

Best performance in a single learner setting is achieved with a combination of out-of-coverage selection and uncertainty sampling. In particular, we preferably select out-of-coverage sentences (according to an unsmoothed parser) and fill the batch with high-entropy sentences. This finding is another manifestation of the central idea of this thesis that unreliable parameters should be explicitly targeted. By contrast, uncertainty sampling, as standardly applied, does not incorporate such mechanisms and yields inferior results. We also see that the optimal smoothing regime for sample selection may well be different than for testing.

6.3 Query-By-Committee for Parsing

We have seen in the two previous chapters that using QBC can result in substantial improvements over single learner uncertainty sampling by identifying unreliable parameters based on infrequent parsing events. This section is concerned with finding optimal parameter settings for QBC with the aim of improving over the best uncertainty sampling results in this chapter. We compare and optimise the performance of ensembles along the following dimensions:

- **Divergence Metric:** Jensen-Shannon divergence; vote entropy
- **Smoothing:** On; Off
- **Ensemble Size:** Small; Large

In preliminary experiments, we found that using either bagging or Dirichlet sampling as sampling methods resulted in very similar results. We will use bagging as the sampling method.

Assigning scores for partially unparsable sentences

Throughout the following experiments, we preferably select sentences with high disagreement measured either as JS-divergence or as vote entropy. This raises the question how to apply these metrics in cases where a sentence is unparsable for some or all ensemble parsers due to coverage or timeout problems.

A possible solution is to ignore unparsable sentences in such situations and compute divergence only for the remaining analyses. However, in addition to not working well in practice, this is unsatisfactory for a number of reasons. Most importantly, this approach does not allow us to properly distinguish between cases with many and few unparsable situations even though, intuitively, we would like to prefer sentences where the ensemble has many unparsable cases. Also, we would have to pay attention to a number of special situations: i) in cases where no ensemble member can analyse a particular sentence, both divergence metrics are ill-defined; instead we may want to assign a high score; ii) in cases where only one member can analyse a sentence, the current definitions for divergence metrics would suggest complete agreement which is counter-intuitive; again we may want to assign a high score.

To remedy such problems, we employ the following strategy:[3] We replace every unparsed result in an ensemble with a single and unique analysis (associated with a probability of 1.0). This addresses the main problem with the previous solution since this way we assign larger divergences in case of larger numbers of unparsed results. At the same time, we deal with all mentioned special cases: i) when no ensemble parser can analyse a sentence, we have a maximal number of unique analyses and will assign a maximal score for both JS-divergence and vote entropy; ii) when only one member can analyse a sentence we again have a maximal number of unique analyses since the one genuine analysis will be different from all artificially assigned analyses, and we assign a maximal score in this case.

6.3.1 Impact of Smoothing on Parser Ensembles

In the previous section, we have established that it is beneficial to use an unsmoothed parser for uncertainty sampling while applying parsability as a selection criterion. The first set of experiments in this section will look at the question whether this also holds true when using QBC as a sample selection metric. We contrast ensembles of smoothed parsers with ensembles of unsmoothed parsers. Because active learning using smoothed parsers is extremely time consuming, we compare performances only for a minimal ensemble containing two ensemble members here. We make these comparisons for the following set of conditions:

- Bagging, JS-divergence
- Bagging, vote entropy

Bagging, JS-divergence

In our first comparison between smoothed and unsmoothed ensembles, we use bagging as a sampling method and JS-divergence as a divergence metric. Figures 6.11 and 6.12 show results when starting with initial training sets of 100 and 1000 sentences.

The unsmoothed method (QBC-NSM) performs significantly better than random sampling; following the second iteration when starting with a small training set, and right from the start when starting with a large set.

The smoothed method (QBC-SM) performs worse than random sampling when starting with a small training set. When starting with a large training set, the smoothed

[3] We are grateful to David Talbot who suggested this approach.

method performs better than random sampling; but the difference is significant only in some of the iterations.

Overall, the unsmoothed method clearly outperforms the smoothed method; significantly through all iterations for the small training set and until 3200 sentences have been sampled for the large training set.

Bagging, Vote Entropy

Next, we compare smoothed and unsmoothed ensembles when using bagging as a sampling method and vote entropy as a divergence metric. Figures 6.13 and 6.14 show results when starting with initial training sets of 100 and 1000 sentences.

We first note that the performance of smoothed and unsmoothed ensembles is much closer than in the two previous experiments using Jensen-Shannon divergence. Performance is almost identical when starting early; almost all pairwise comparisons are not significant. Performance of an unsmoothed of an unsmoothed ensemble is just a little bit better when starting late; and improvements are significant only in some iterations.

In comparison to random sampling, we find that both smoothed and unsmoothed ensembles perform significantly better when starting early; and mostly significantly better when starting late. However, we find that the improvement of using an unsmoothed ensemble over a smoothed ensemble is not as good as in the previous experiments.

6.3.2 Increasing Ensemble Size

As we saw in the previous chapter, using larger ensemble sizes in Query-by-Committee can improve active learning performance. We now examine if this is also the case when applied to parsing. To this end, we contrast small ensembles of two parsers with larger ensembles of five parsers. Having established that it is not beneficial to use smoothed parser ensembles, we only use unsmoothed ensembles for this experiment. Again, we make these comparisons for the set of conditions:

- Bagging, JS-divergence
- Bagging, vote entropy

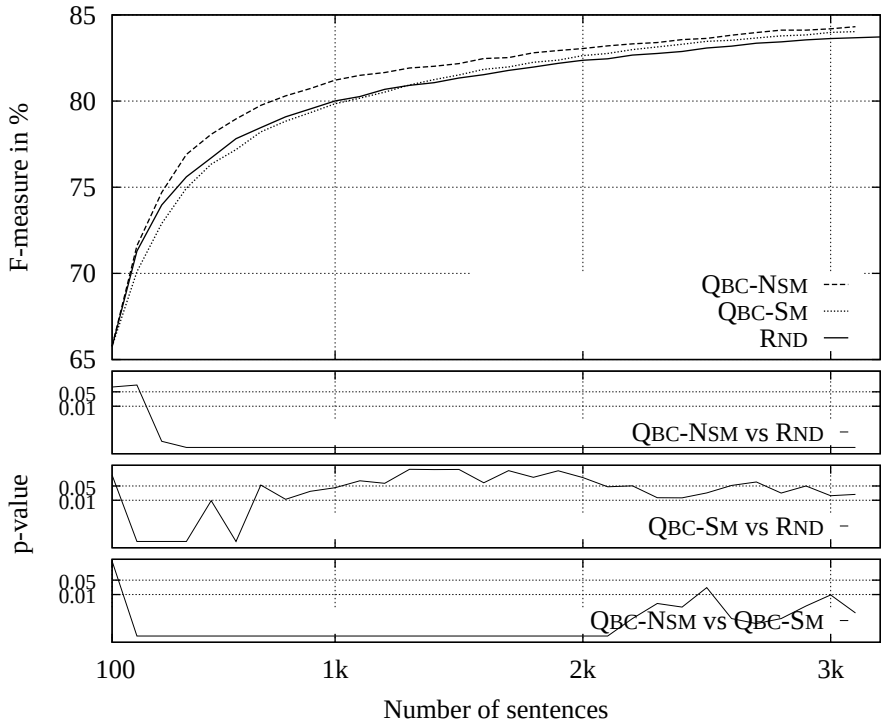

Figure 6.11: Smoothing settings for bagging/JS-divergence (from 100 sentences)

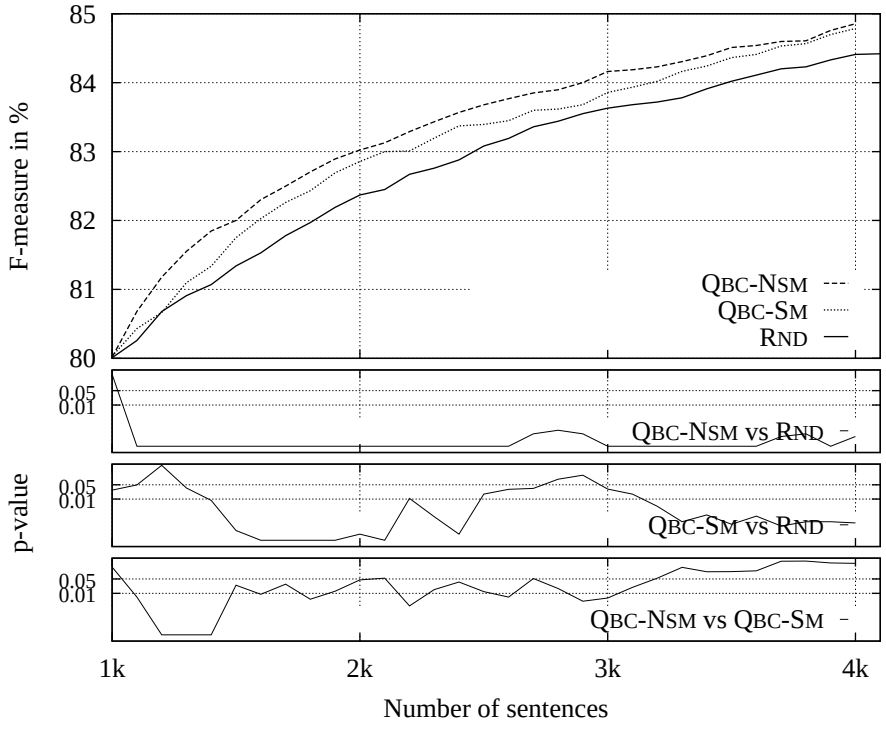

Figure 6.12: Smoothing settings for bagging/JS-divergence (from 1000 sentences)

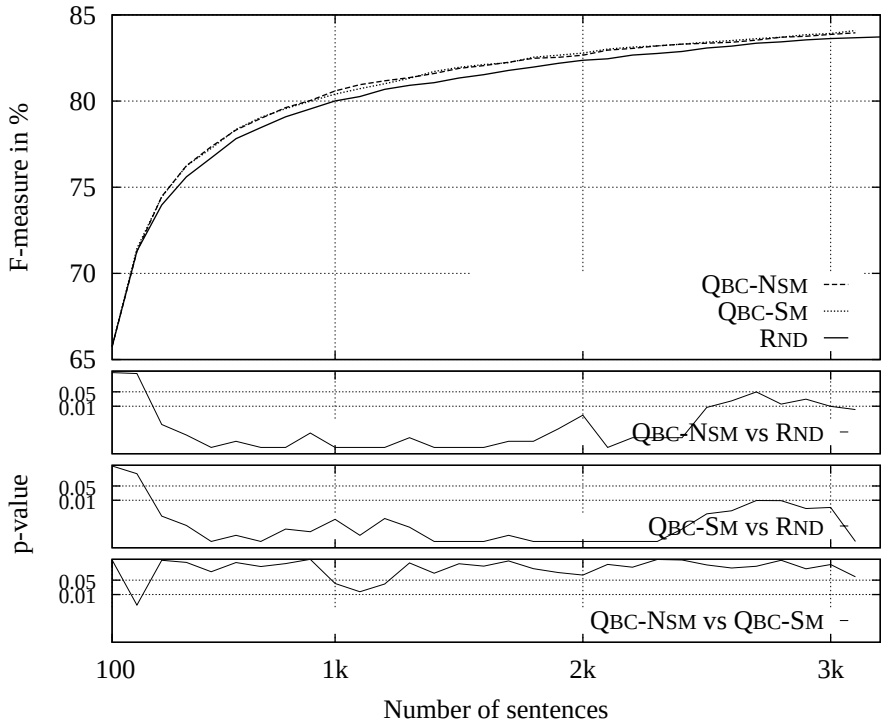

Figure 6.13: Smoothing settings for bagging/vote entropy (from 100 sentences)

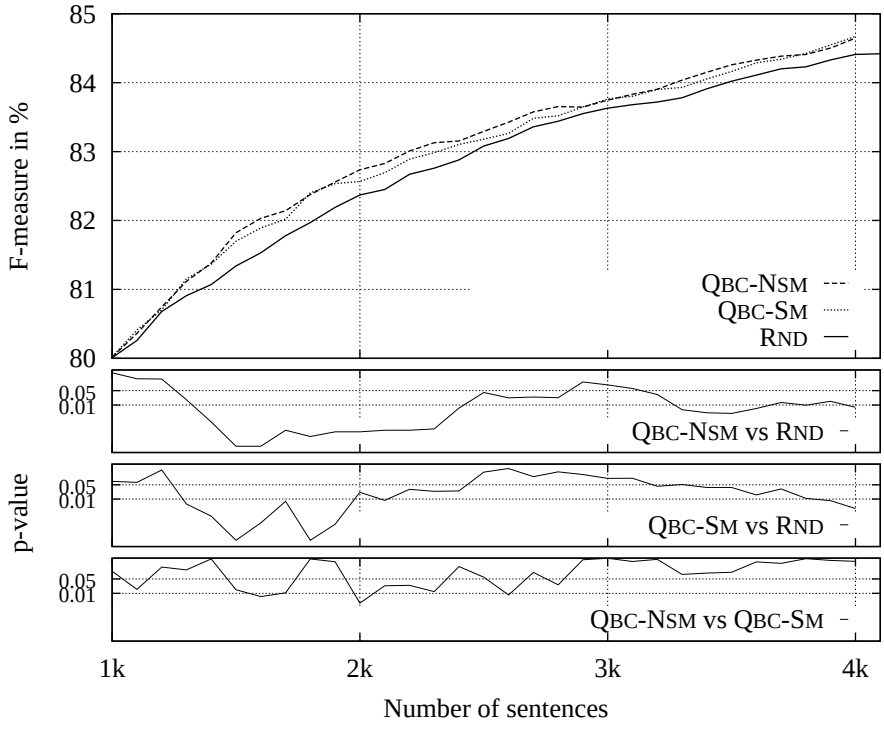

Figure 6.14: Smoothing settings for bagging/vote entropy (from 1000 sentences)

Bagging, JS-divergence

In our first comparison between small and large ensembles, we use bagging as a sampling method and JS-divergence as divergence metric. In Figures 6.15 and 6.16, we see that using a large ensemble results improves performance over using a small ensemble. When starting early, this improvement is significant after 12 iterations; when starting late, improvements are significant after only four iterations.

Bagging, Vote Entropy

Next, we compare small and large ensembles when using bagging as a sampling method and vote entropy as divergence metric. In Figures 6.17 and 6.18, we see again that using a large ensemble results improves performance over using a small ensemble. Improvements are significant throughout, both when starting early and late.

JS-divergence versus Vote Entropy

When comparing performance for large ensembles between divergence metrics, we find vote entropy and JS-divergence to be almost identical when starting early. (Results not included here.) When starting late, JS-divergence performs slightly better but not significantly so. For the remaining comparisons, we will use large ensembles with a combination of JS-divergence and bagging.

6.3.3 Comparison with Uncertainty Sampling

Having identified a setting for Query-by-Committee which results in best improvements over random sampling, it is interesting to see how much we gain over the best results for uncertainty sampling, see results in Figures 6.19 and 6.20. When starting early we find that uncertainty sampling and Query-by-Committee perform almost identical. When starting late improvements of using Query-by-Committee over uncertainty sampling are significant only from iterations 3 to 11. Given the computational cost of running an ensemble as opposed to a single learner method may suggest the preferred use of single learner uncertainty sampling over Query-by-Committee.

6.3.4 Summary

In this section, we have looked at different aspects of ensemble methods, including the role of smoothing, different sampling methods, different divergence metrics and

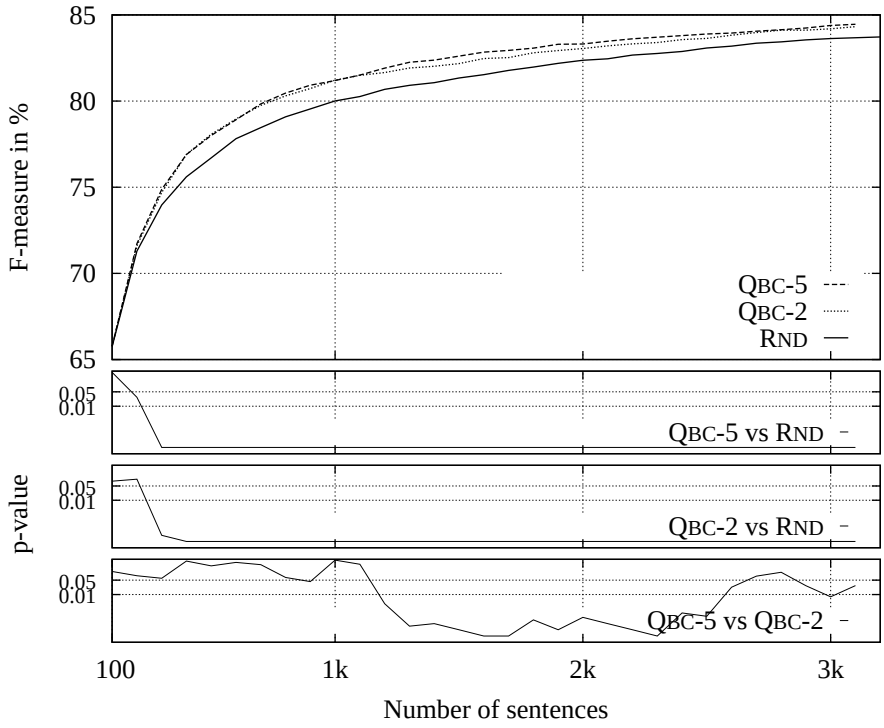

Figure 6.15: Ensemble sizes for bagging/JS-divergence (from 100 sentences)

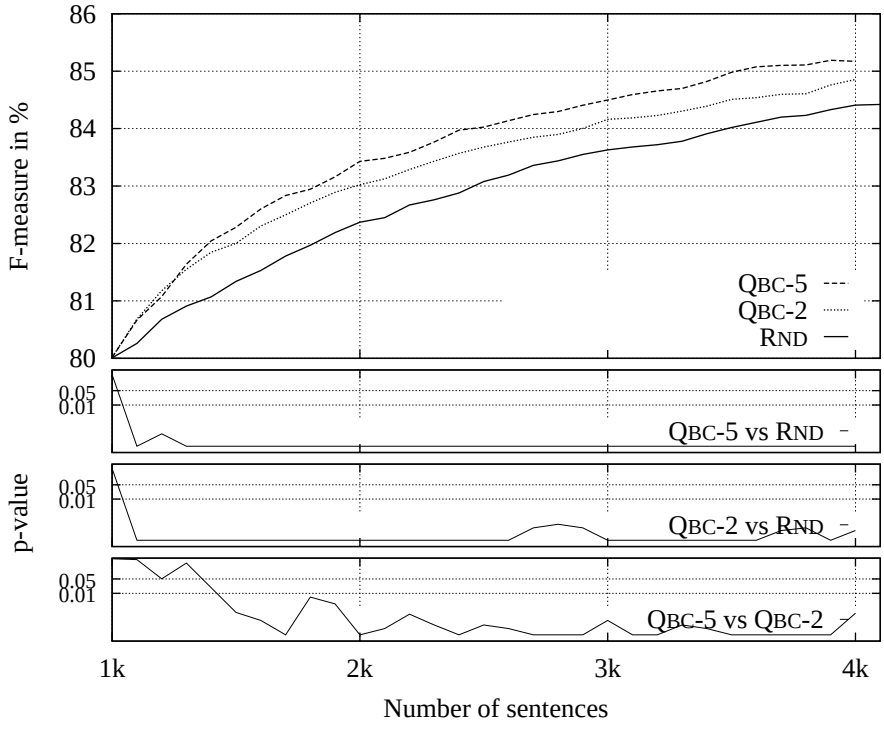

Figure 6.16: Ensemble sizes for bagging/JS-divergence (from 1000 sentences)

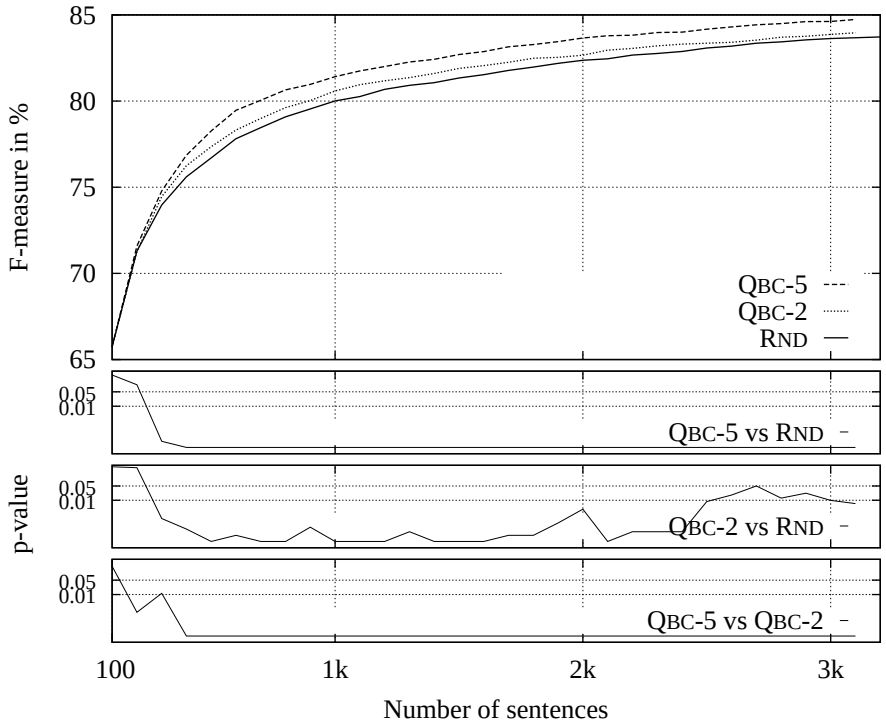

Figure 6.17: Ensemble sizes for bagging/vote entropy (from 100 sentences)

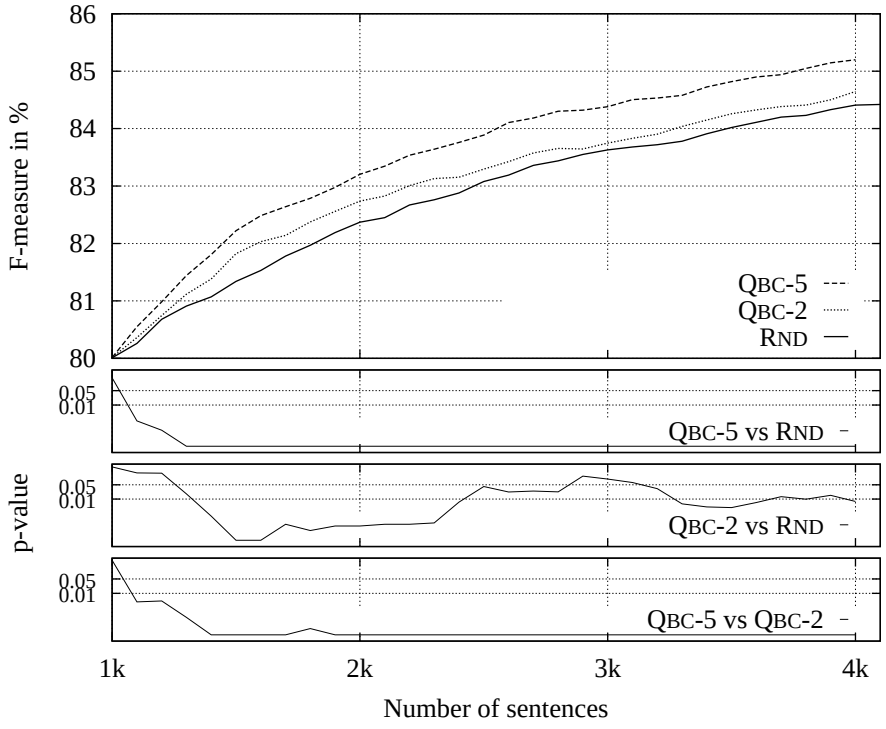

Figure 6.18: Ensemble sizes for bagging/vote entropy (from 1000 sentences)

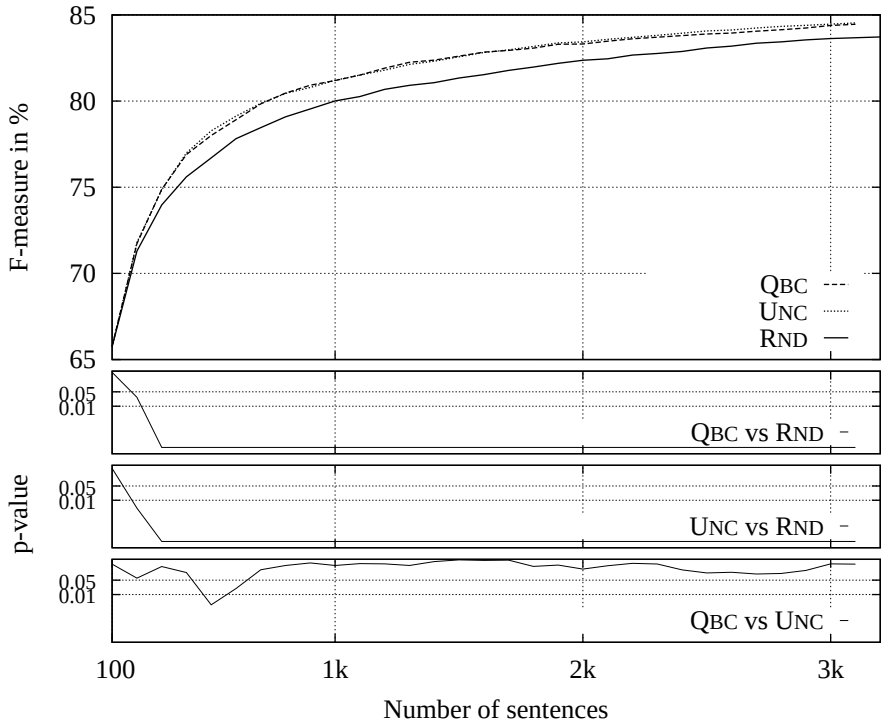

Figure 6.19: Best settings for uncertainty sampling and QBC (from 100 sentences)

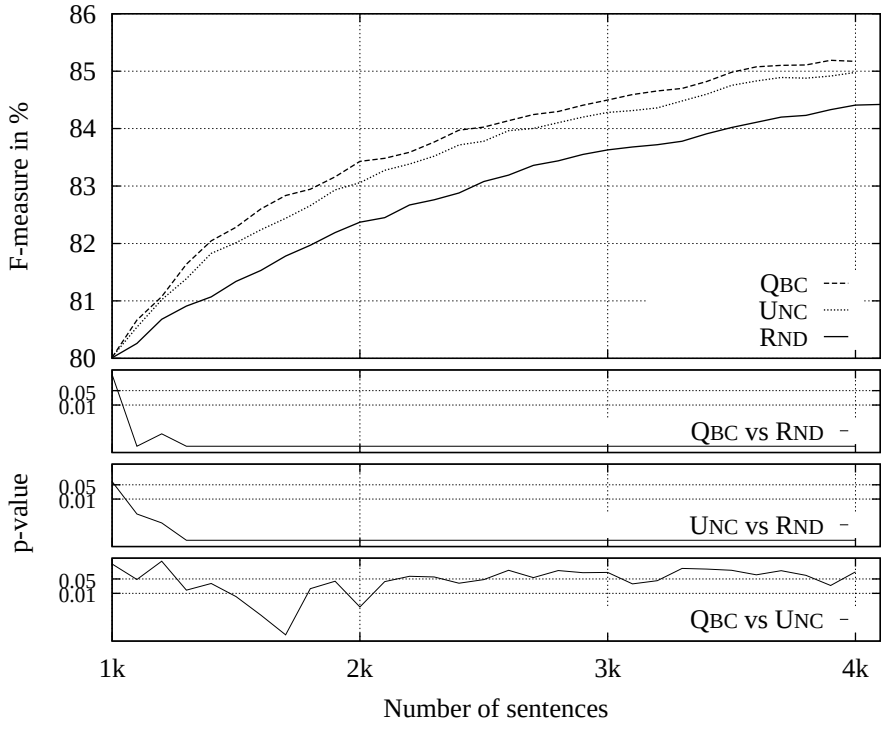

Figure 6.20: Best settings for uncertainty sampling and QBC (from 1000 sentences)

ensemble size. Best results are achieved with ensembles of unsmoothed parsers in combination with Jensen-Shannon divergence as a measure of disagreement. Using smoothed parsers ensembles in combination with this divergence metric results in a decrease of performance, and may even perform worse than random sampling.

Using vote entropy results in lower performance, in particular when used with small ensembles. This is in contrast to our findings with respect to prepositional phrase attachment and part-of-speech tagging, but serves to illustrate the fact that divergence metrics should be carefully selected for the task at hand.

Using either bagging or Dirichlet sampling does not make much of a difference. This is a finding of practical relevance since generally bagging of the training set is simpler to implement than Dirichlet sampling.

Increasing the ensemble size has a positive influence on performance. However, even with an ensemble of a moderate size it can be difficult to outperform uncertainty sampling in combination with out-of-coverage selection.

6.4 A Novel Two Stage Method

Acquiring the correct analysis of a sentence for which the predicted analysis was selected on the basis of infrequent parsing events may well be informative. A simple but effective method is to eliminate some infrequent events from the parsing model. Simply bagging the current training set, and retraining the parser on this set allows to identify such examples for labelling.

The proposed method operates in two stages. First, we parse all pool sentences with a parser which has been trained on a bootstrap replicate of the training set. Second, we parse the pool sentences with a parser trained on the full training set. In both stages, we deliberately allow the parser to fail by not applying constraint relaxation. The selection of sentences for manual annotation proceeds as follows. As in the coverage-based methods, we first select sentences which are unparsable according to the fully trained parser. From the remaining sentences, we select sentences which are unparsable according to the bagged parser. Finally, we select those sentences with the highest entropy according to the fully trained parser.

We can express this formally as follows:

$$f^{two}_{M,M'}(s,\tau) = \max(\mathit{failure}(s,M,2c), \mathit{failure}(s,M',c), f^{te}_{M}(s,\tau_M)) \tag{6.1}$$

where f^{te}_{M} is the entropy according to a fully trained model M, defined in Equation 2.21.

The function $failure(s, M, c)$ returns a very large constant c when sentence s is unparsable given parser model M and 0 otherwise. M denotes a fully trained model, and M' a bagged model.

This method operates very similar to the coverage- and uncertainty-based method described in Section 6.2. The difference is the extra layer with the bagged parser which is designed to additionally identify sentences of which the analysis is based on infrequent events.

Experiments

In the following, we will compare the two-stage method against earlier best results from uncertainty sampling and QBC.

Figures 6.21 and 6.22 show results when comparing the two-stage method against best uncertainty sampling, namely the coverage- and uncertainty-based selection method. We find that the two-stage method does not achieve an improvement over uncertainty sampling when starting early. However, when starting late we find a significant improvement right from the beginning.

These two methods are identical in their uncertainty sampling component and differ only in the way how out-of-coverage sentences are identified initially. The coverage- and uncertainty-based method selects unparsable sentences according to a fully trained parser; the two-stage method selects them according to both a fully trained parser and a bagged parser. While the former is good at identifying examples with unseen parsing events, the latter identifies examples with both unseen and infrequent parsing events. Clearly, the improved performance of the two-stage method seen in Figure 6.22 is attributable to this fact.

Figures 6.23 and 6.24 show the results when comparing the two-stage method against QBC at the best setting, namely QBC with an ensemble of 5 members, using bagging and JS-divergence. We find that the two-stage method is always as good as or better than QBC, and never significantly worse for both starting points.

As argued before, QBC can be good at identifying examples from which we can learn both unseen and infrequent parsing events. The almost equivalent performance seen in the last experiment indicates that the two-stage is as good in this respect as QBC.

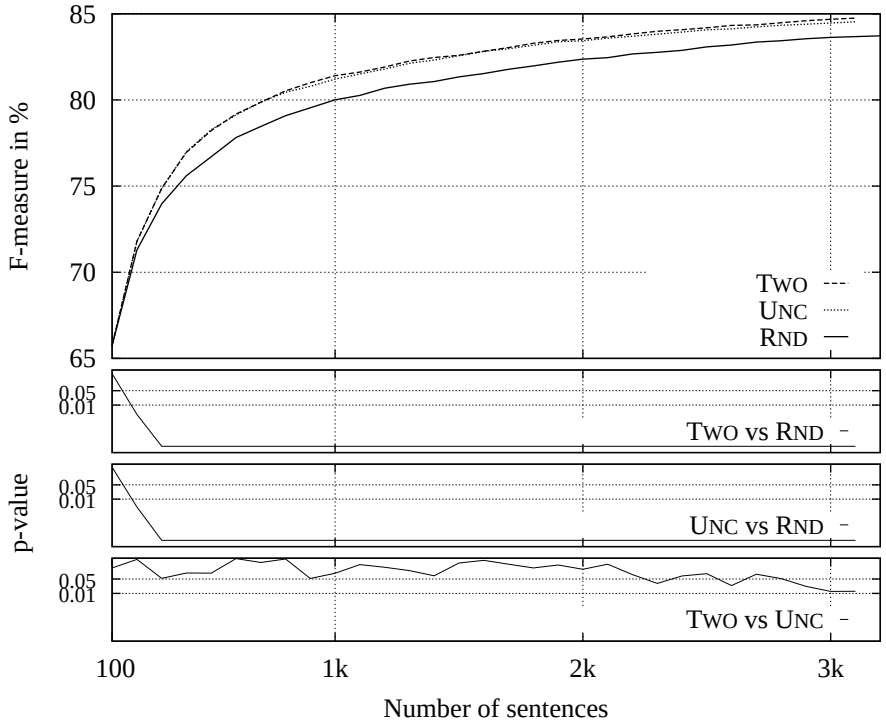

Figure 6.21: Comparing two-stage method with best uncertainty sampling (from 100 sentences)

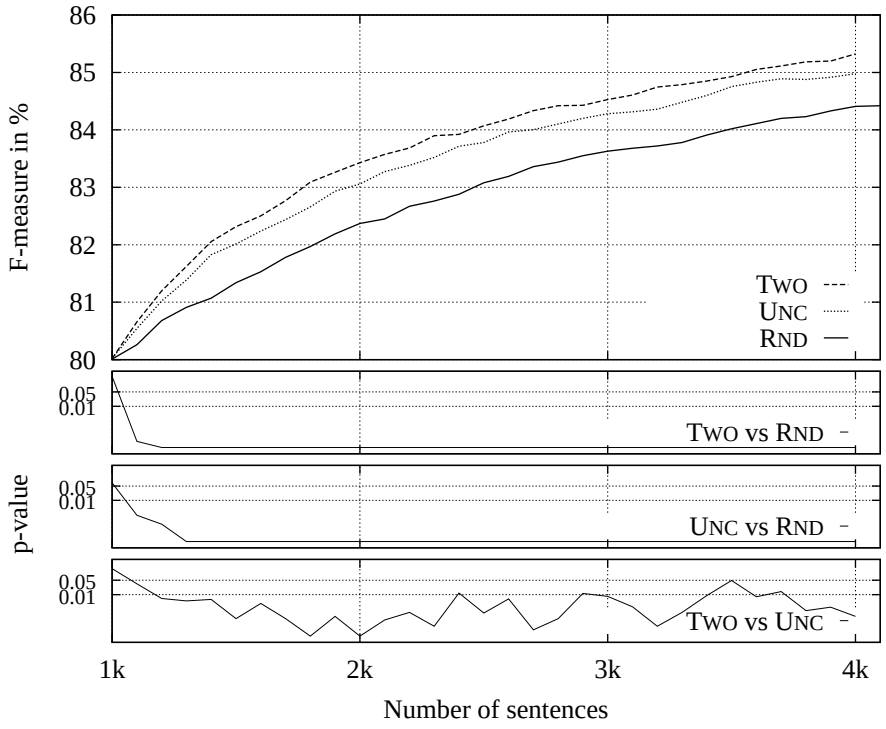

Figure 6.22: Comparing two-stage method with best uncertainty sampling (from 1000 sentences)

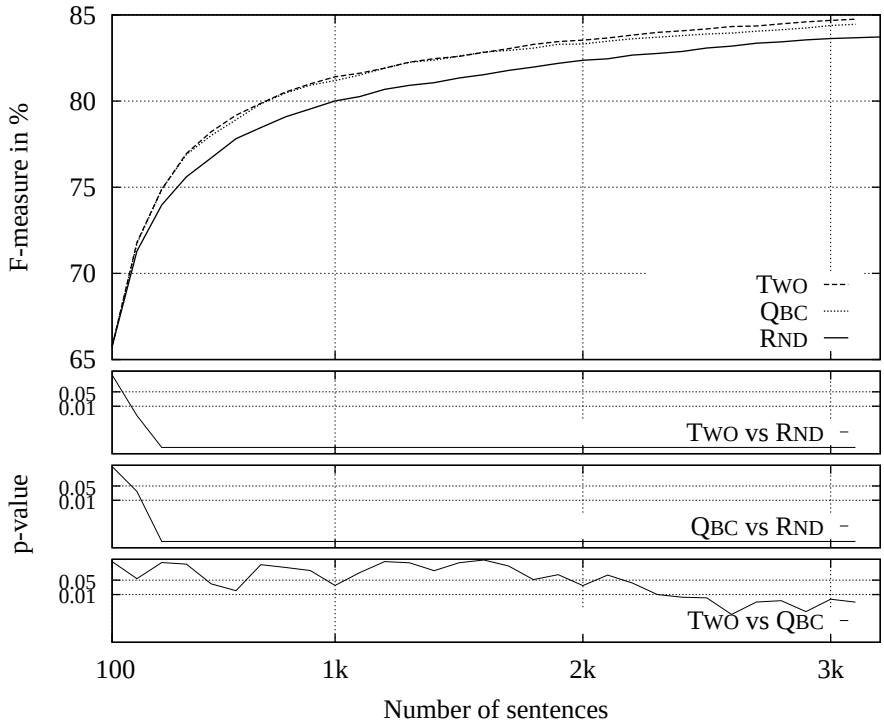

Figure 6.23: Comparing two-stage method with best QBC (from 100 sentences)

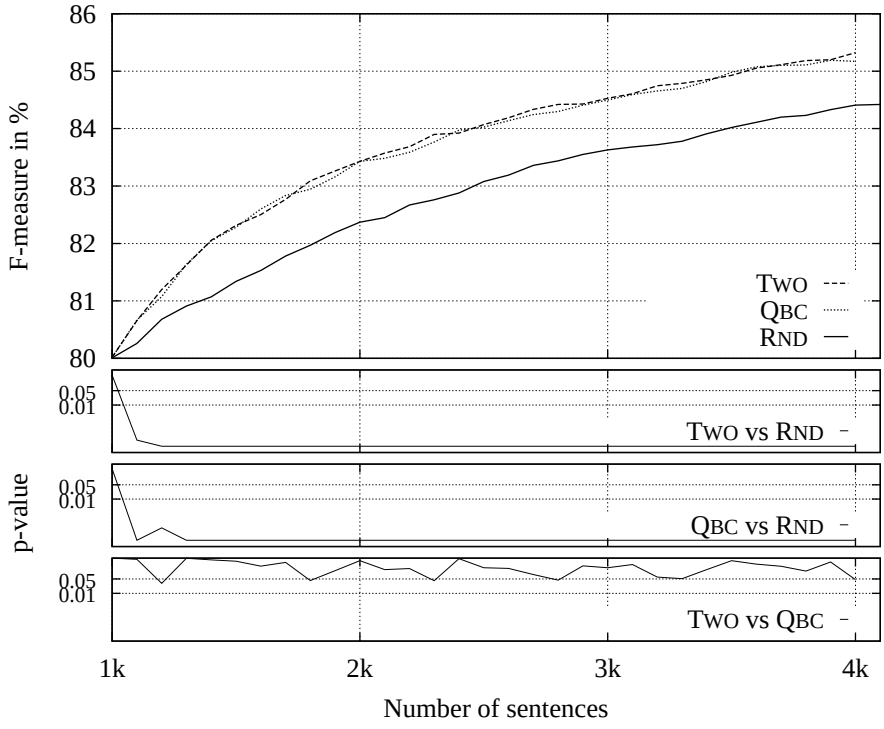

Figure 6.24: Comparing two-stage method with best QBC (from 1000 sentences)

A Baseline

The two-stage method uses bagging to eliminate low frequency events from the training set. This raises the question if we can achieve comparable results using a simple cutoff instead of bagging to create a reduced parser. To this end we perform a baseline experiment where we replace the bagging component of the two-stage method with a frequency cutoff, removing all parse events which occur fewer than n times. We found little difference in the choice of n for reasons that we will explain below. For the following experiment, we set $n = 2$, removing all parse events which occur only once.

Figures. 6.25 and 6.26 show that such a baseline method initially performs well but eventually converges towards the performance of random sampling. This behaviour is very similar to that of coverage-based sampling which we display in the same graph for demonstrative purposes. The significance test for the pairwise comparison of this baseline method against coverage-based sampling shows that both methods are virtually indistinguishable.

To explain the difference between the baseline method and the two-stage method, we look at the coverage of the reduced parser for both methods, that is the bagged parser and the cutoff parser, on the pool. Figures 6.27 and 6.27 show that coverage of a bagged parser converges towards 99%, starting from below 30% when starting early and from 90% when starting late. By contrast, a cutoff parser has very little coverage, starting from almost 0% towards 20% coverage when starting early and towards 25% when starting late. This means that the majority of the pool sentences would be marked for preferable selection. In a situation where we fill a batch which is considerably smaller than the pool, this effectively results in random selection in the second selection phase, and since the third phase using uncertainty sampling is never reached we observed random selection behaviour throughout after the first phase. This explains the similarity in behaviour with coverage-based sampling. Using higher cutoff values for n only aggravates the problem, and results in even lower levels of coverage.

In summary, we find that the bagged component of the two-stage method is indeed essential and cannot be replaced with a simple cutoff method.

Summary

We introduced a novel two-stage method which primarily selects unparsable sentences according to a fully trained parser and a bagged parser (both without constraint relaxation). This technique identifies both unseen and infrequent parsing events. This

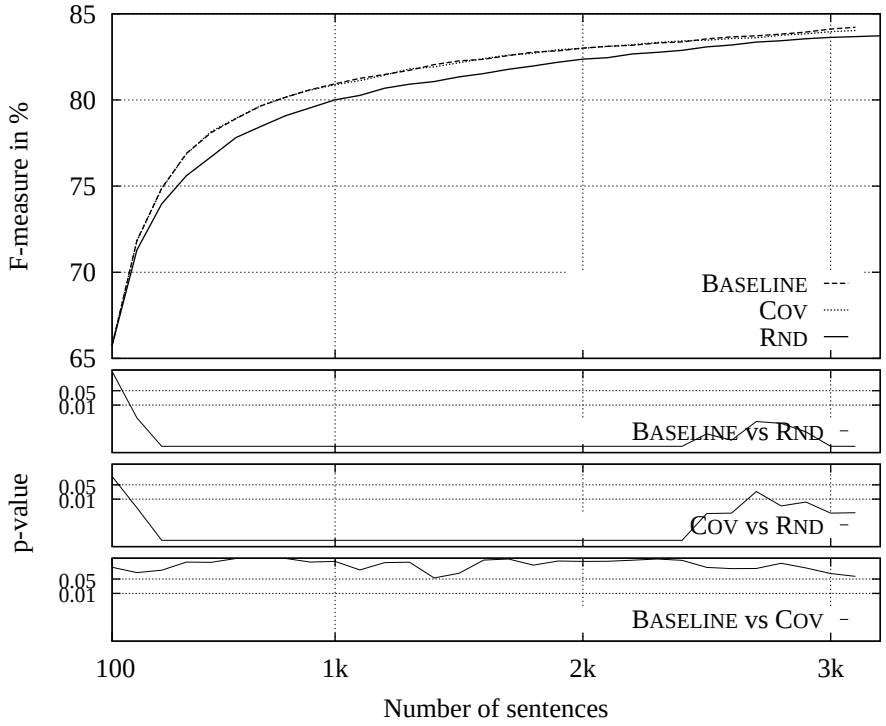

Figure 6.25: Comparing baseline method with coverage-based selection (from 100 sentences)

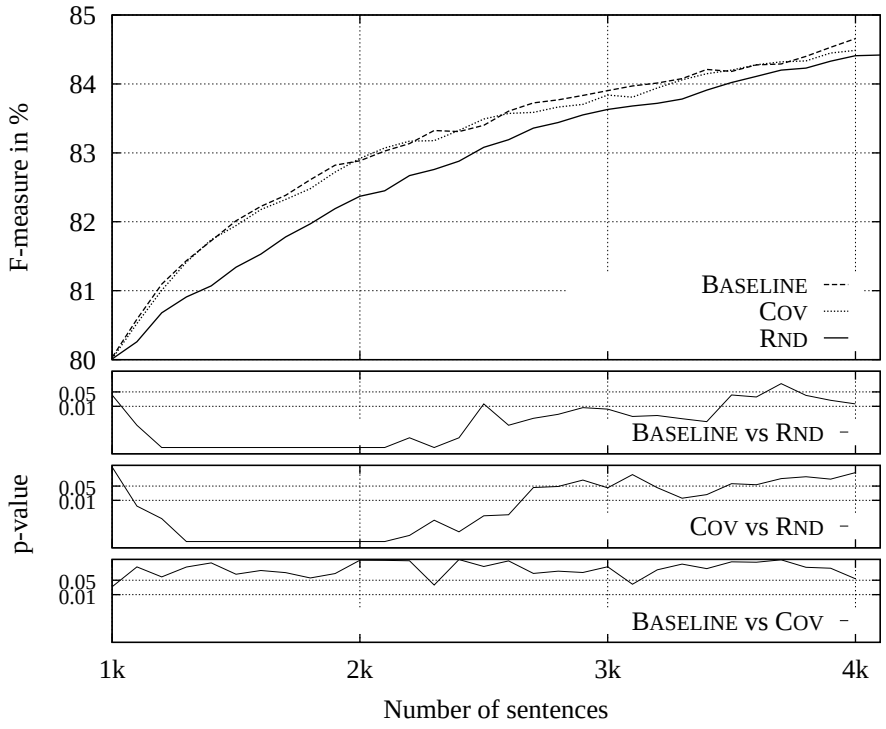

Figure 6.26: Comparing baseline method with coverage-based selection (from 1000 sentences)

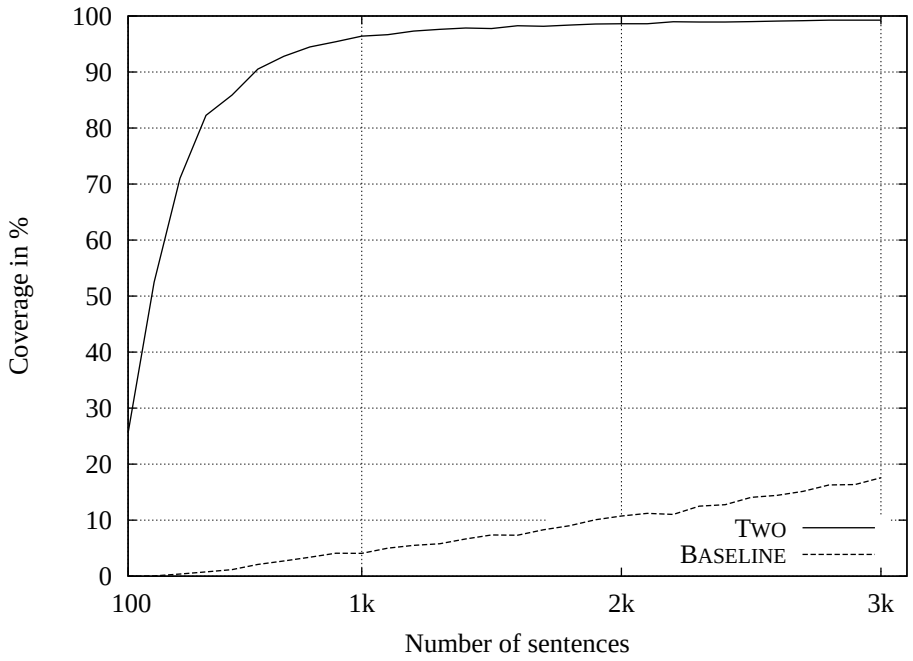

Figure 6.27: Comparing coverage on the pool between baseline and two-stage method (from 100 sentences)

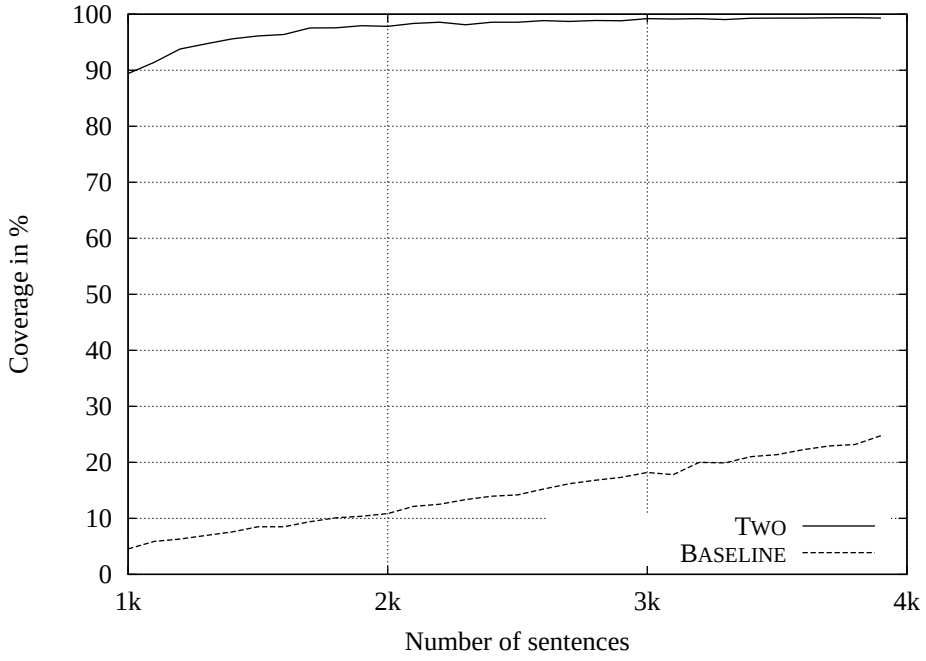

Figure 6.28: Comparing coverage on the pool between baseline and two-stage method (from 1000 sentences)

two-stage method performs as well as or better than the best uncertainty sampling method, and just as well as the best QBC method, while being considerably cheaper to run.

6.5 Conclusion

In this chapter, we demonstrated various ways of dealing with unreliable parameter problems in the case of parsing. A particular important form of unreliable parameters are unseen head- and modifier-generation events, that is, events which cannot be learned from the current training set. Parsing failures due to such unknown events impair both recall and f-measure.

Coverage problems can be addressed in different ways. A standard way is to apply more aggressive smoothing regimes. In a random sampling experiment, we demonstrated that the application of constraint relaxation in Bikel's parser is indeed a successful way of bringing coverage close to 100%. This causes a strong increase in recall, especially for small training sets, while only minimally impairing precision. Correspondingly, we observe a highly significant increase in f-measure for all training set sizes.

In the context of active learning, we can explicitly target unparsed sentences and thus learn new events, rather than applying smoothing to fix such problems after the fact. This requires that the parser be run with a lesser degree of smoothing during sample selection. We show that selecting unparsed sentences as the only sample selection method is successful in its own right; this method combines gracefully with other active learning methods such as uncertainty sampling. By contrast, the use of a fully smoothed parser in uncertainty sampling yields inferior results; this is because sentences which are difficult to parse and hence would be useful to annotate do not show higher entropy than other sentences.

The observation that it may be necessary to apply less smoothing in the sample selection phase in order to have the full benefit of an out-of-coverage selection strategy is a novel and important contribution of this chapter. While it has been recognised before that out-of-coverage examples should be selected preferably when applying active learning to parsing, for instance (Thompson et al., 1999) and (Hwa, personal communication, 2003), the importance of this question has not received any attention.

Identifying infrequent (rather than unseen) events is in the domain of QBC. By randomly perturbing parsing models, parameters based on infrequently observed events

will show higher variance and hence higher disagreement. In our experiments, we show that it is beneficial to select sentences which require such parameters for their decoding. We optimised sample selection performance for QBC by exploring relevant experimental parameters. We achieve best results using bagged ensembles with JS-divergence. Furthermore, we find that using larger ensembles further improves performance. As with uncertainty sampling, using unsmoothed parsers is beneficial for ensemble-based methods.

Finally, we introduced a novel two-stage method which primarily selects unparsable sentences according to a fully trained and a bagged parser. This technique identifies both unseen and infrequent parsing events in cases where they were deleted during bagging. In order to fill the batch we apply uncertainty sampling to the remaining sentences using a fully trained parser. This two-stage method performs as well as the best QBC methods, and is considerably cheaper to run.

We have demonstrated the importance of addressing unreliable parameters when applying active learning to parsing. New parsing events are effectively learned by selecting out-of-coverage sentences; parameter estimates of infrequently observed events can be improved using the two-stage method. These are the novel contributions of this chapter.

Chapter 7

Conclusion

This thesis concerns the proper treatment of sparse data problems when applying active learning to natural language processing tasks. Sparse data problems are ubiquitous in natural language processing due to the Zipfian nature of language. When randomly sampling a training set from a population, sparse events will appear either infrequently or not at all. Both cases are problematic for the supervised learning of statistical models and can result in degraded classification performance. Infrequently observed events cause high variance when estimating model parameter, and missing events can cause the model structure to be incomplete such that a trained classifier may not be able to predict any label for some input.

Active learning is a class of methods which are supposed to reduce the amount of manually annotated data necessary for the supervised training of classifiers to reach a given performance level. However, the two most popular active learning methods in the literature, uncertainty sampling and QBC, have severe shortcomings with regard to sparse data situations. Neither method has a principled way to deal with missing events as they are both defined to refine probability estimates within a given model structure. In view of these shortcomings of active learning, we stated the **Unreliable Parameter Principle**:

> Active learning should explicitly and additionally address unreliably trained model parameters in order to optimally reduce classification error. In order to do so, we should target both missing events and infrequent events.

In this thesis, we demonstrated how this principle applies to a variety of problems, namely prepositional phrase attachment, part-of-speech tagging, named entity recognition (NER) and syntactic parsing.

7.1 Contributions of the Thesis

The main contributions of this thesis fall into the following categories.

Comparison between Uncertainty Sampling and QBC

Uncertainty sampling and QBC are very commonly used active learning methods. For this reason, we provided an extensive comparison of these methods for a variety of natural language processing tasks.

In a number of situations, we found that it is easy to misapply active learning when using unfortunate experimental parameters to the degree that active learning underperforms random sampling. However, we found that, in general, both methods perform better than random sampling as was expected. Furthermore, we found that QBC outperforms uncertainty sampling in most cases. Surprisingly, this is not the case in NER. We conducted some preliminary experiments which indicate that the relative performance of active learning methods may be related to the size of the tagset.

QBC is equipped with a number of experimental parameters which need to be set properly for good active learning results. In particular, we explored the use of ensemble creation method and divergence metric. With respect to ensemble creation method, we explored bagging (Abe and Mamitsuka, 1998) and Dirichlet sampling (McCallum and Nigam, 1998; Argamon-Engelson and Dagan, 1999) as popular methods. We found empirically that QBC using bagging performed as well or better than QBC using Dirichlet sampling in all applications. From a practical point of view, this is an expedient result for QBC, since bagging is considerably easier to implement than Dirichlet sampling in most cases. Bagging only requires the application of sampling with replacement to the training set, leaving the subsequent training and application of the classifier unchanged. By contrast, Dirichlet sampling requires the application of resampling techniques on the level of individual distributions within the model.

With respect to the choice of divergence metrics we get varied results. For PPA and part-of-speech tagging, we get best results with QBC using vote entropy, while best results for parsing were achieved with QBC using JS-divergence. McCallum and Nigam, 1998 found that JS-divergence works better than vote entropy but were careful to note that this finding is specific to their application of text classification. It seems that later work has overgeneralised their finding, for example (Melville and Mooney, 2004). Our findings indicate that the choice of appropriate divergence metric may well need to be established from domain to domain.

In general, these findings suggest caution when choosing active learning methods for novel applications. To the best of our knowledge, such a comprehensive comparison between uncertainty sampling and QBC across different tasks has not been presented in previous literature. Similarly, the exploration of the experimental parameter space for QBC is novel.

Explicitly Targeting Unreliable Parameters via Smoothing/Backing-Off

Choosing appropriate smoothing or backoff settings in active learning can be vital to allow for the targeted selection of out-of-coverage examples. We demonstrate this for prepositional phrase attachment and for parsing. For prepositional phrase attachment, assigning an appropriate backoff probability helps to select unknown prepositions and substantially improves coverage and accuracy for both uncertainty sampling and QBC. For parsing, switching off *constraint relaxation* (an effective smoothing method) helps to select out-of-coverage examples and substantially improves coverage and f-measure.

The observation that altered smoothing regimes in active learning can be beneficial is an important contribution of this thesis. This may seem counter-intuitive at first and goes against the practice in the field to use the same model for active learning and for later testing purposes (Baldridge and Osborne, 2004).

To provide an intuition, we offer the following analogy: Training a model from annotated data is like building a house, where we liken sparse data problems to cracks in the wall. To make a model usable in practice we need to apply smoothing; similarly, we make a house habitable after completion by papering cracks over with wallpaper. Staying in the analogy, active learning can be compared to building a house in stages. When acquiring data for the next stage, we should focus on structural problems such as missing events in the model. To expose such problems, we do not apply smoothing (or apply smoothing to a lesser degree). The application of smoothing in this stage would be like having to find cracks in the wall while the wallpaper is up already.

Explicitly Targeting Unreliable Parameters via Other Methodologies

Beyond adjusted smoothing, we introduced two novel methods in this thesis which explicitly target either out-of-coverage examples or variance in parameter estimates.

For sequence labelling tasks, we introduced a novel method which directly counts the number of unreliable parameters based on missing events which are needed for decoding a given sentence. This method in isolation is as good as or better than un-

certainty sampling. Furthermore, we established that it tends to select examples which are different from the examples that uncertainty sampling would select. This situation suggests the combination of the count-based method with uncertainty sampling and, in fact, the combination beats both methods in isolation and matches the best overall result from QBC. This method directly implements the missing events aspect of the Unreliable Parameter Principle and demonstrates that directly addressing unreliable parameters is a successful strategy.

For parsing, we introduced a novel two-stage method which primarily selects unparsable sentences according to a fully trained parser and a bagged parser (both without constraint relaxation). This technique identifies both unseen and infrequent rules in cases where they were deleted during bagging. In order to fill the batch we apply uncertainty sampling to the remaining sentences using a fully trained parser. This two-stage method performs as well as the best QBC methods, and is considerably cheaper to run. This method implements both the missing event and the infrequent event aspect of the Unreliable Parameter Principle.

7.2 Future Work

In this thesis, we have identified the need to deal with sparse data problems in the context of active learning and suggested a variety of methods appropriate for the respective tasks. We have not provided a formal or a unified treatment how to deal with sparse data problems in active learning. Devising such a treatment would be a worthwhile endeavour but is outside the scope of this thesis.

Other worthwhile avenues of research would be to investigate the combination of active learning methods suggested in this thesis with other work in the field. In particular, density estimation may help to iron out some of the worst problems we have identified for uncertainty sampling (McCallum and Nigam, 1998; Tang et al., 2002). The application of online choice algorithms (Baram et al., 2004) may obviate the need to commit to one particular primary active learning method. However, while each of this approaches could potentially further improve performance, we believe that they are orthogonal to our methods.

A more general remark concerns the question why active learning methods have been deployed so little in practical projects since their inception. Most active learning research assumes that an active learning method almost exclusively drives the sampling process, leading to a bias in the selected data. However, it should be acknowl-

edged that there is an inherent value in randomly sampled data, for example for corpus linguistics. Correspondingly, annotation projects may be rather unwilling to hand over control about how data are sampled to an active learning method. A line of work that we would like to follow in the future is to consider active learning as an add-on to existing annotation project, such that active learning is only employed late in the entire annotation process. Interesting research would concern the question when to start the active learning process. The use of the techniques suggested in this thesis may well help to further improve performance.

Bibliography

Abe, N. and Mamitsuka, H. (1998). Query learning strategies using boosting and bagging. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, Madison, WI, USA.

Altmann, G. T. and Steedman, M. (1988). Interaction with context during human sentence processing. *Cognition*, 30:191–238.

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

Baayen, H. and Sproat, R. (1996). Estimating lexical priors for low-frequency morphologically ambiguous forms. *Computational Linguistics*, 22(2):155–166.

Baldridge, J. and Osborne, M. (2003). Active learning for HPSG parse selection. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-03)*, Edmonton, AB, Canada.

Baldridge, J. and Osborne, M. (2004). Active learning and the total cost of annotation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, Barcelona, Spain.

Baram, Y., El-Yaniv, R., and Luz, K. (2004). Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291.

Becker, M., Hachey, B., Alex, B., and Grover, C. (2005). Optimising selective sampling for bootstrapping named entity recognition. In *Proceedings of the ICML-2005 Workshop on Learning with Multiple Views*, Bonn, Germany.

Becker, M. and Osborne, M. (2005). A two-stage method for active learning of statistical grammars. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005)*, Edinburgh, UK.

Bikel, D. M. (2004a). Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4):479–511.

Bikel, D. M. (2004b). *On the parameter space of generative lexicalized statistical parsing models*. PhD thesis, Philadelphia, PA, USA.

Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the HLT-91 Workshop on Speech and Natural Language*, Pacific Grove, CA, USA.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory (COLT-98)*, Madison, WI, USA.

Brants, S., Dipper, S., Lezius, S. H. W., and Smith, G. (2002). The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, Sozopol, Bulgaria.

Brants, T. (2000a). Inter-annotator agreement for a German newspaper corpus. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-00)*, Athens, Greece.

Brants, T. (2000b). TnT – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP-00)*, Seattle, WA, USA.

Brants, T., Skut, W., and Uszkoreit, H. (2003). *Building and using Parsed Corpora*, chapter Syntactic Annotation of a German newspaper corpus. Kluwer.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24(2):123–14.

Bresnan, J., editor (1982). *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, MA.

Brill, E. and Resnik, P. A rule-based approach to prepositional phrase attachment disambiguation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, Kyoto, Japan.

Campbell, C., Cristianini, N., and Smola, A. J. (2000). Query learning with large margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, San Francisco, CA, USA.

Carroll, G. and Rooth, M. (1998). Valence induction with a head-lexicalized PCFG. In *Proceedings of the 1998 Conference on Empirical Methods in Natural Language Processing (EMNLP-98)*, Granada, Spain.

Charniak, E. (1996). Tree-bank grammars. Technical Report CS-96-02, Brown University.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proceedings of the 1st Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-00)*, Seattle, WA, USA.

Charniak, E. (2001). Immediate-head parsing for language models. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-01)*, Toulouse, France.

Chinchor, N. (1992). The statistical significance of the MUC-4 results. In *Proceedings of the 4th conference on Message understanding (MUC-4)*, McLean, VA, USA.

Chinchor, N., Lewis, D. D., and Hirschman, L. (1993). Evaluating message understanding systems: an analysis of the third message understanding conference (MUC-3). *Computational Linguistics*, 19(3):409–449.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(2):113–123.

Church, K. W. (1988). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the 2nd Applied Natural Language Processing Conference (ANLP-88)*, Austin, TX, USA.

Cohen, P. R. (1995). *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA.

Cohn, D. A., Atlas, L., and Ladner, R. E. (1994). Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Cohn, D. A., Ghahramani, Z., and Jordan, M. I. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, Madrid, Spain.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

Collins, M. and Brooks, J. (1995). Prepositional phrase attachment through a backed-off model. In *Proceedings of the Third Workshop on Very Large Corpora (WVLC-95)*, Boston, MA, USA.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Dasgupta, S. (2004). Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS-04)*, Vancouver and Whistler, BC, Canada.

Dermatas, E. and Kokkinakis, G. (1995). Automatic stochastic tagging of natural language texts. *Computational Linguistics*, 21(2):137–163.

DeRose, S. J. (1988). Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39.

Dhillon, I. S., Mallela, S., and Kumar, R. (2002). Enhanced word clustering for hierarchical text classification. In *Proceedings of the Eighth International Conference on Knowledge Discovery and Data Mining (KDD-02)*, Edmonton, AB, Canada.

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

Francis, W. N. and Kučera, H. (1964). *Manual of Information to accompany a Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Brown University Press.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

Garside, R., Leech, G., and Sampson, G., editors (1987). *The computational analysis of English. A corpus-based approach*. Longman, London.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural Computation*, pages 1–58.

Grishman, R. and Sundheim, B. (1996). Message understanding conference - 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING-96)*, Copenhagen, Denmark.

Hakkani-Tür, D., Riccardi, G., and Tur, G. (2006). An active approach to spoken language processing. *ACM Transactions on Speech and Language Processing*, 3(3):1–31.

Harabagiu, S., Moldovan, D., Paşca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Gîrju, R., Rus, V., and Morărescu, P. (2001). The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-01)*, Toulouse, France.

Hindle, D. and Rooth, M. (1993). Structural ambiguity and lexical relations. *Computation Linguistics*, 19(1):103–120.

Hwa, R. 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*, Hong Kong.

Hwa, R. (2001a). *Learning Probabilistic Lexicalized Grammars for Natural Language Processing*. PhD thesis, Harvard University.

Hwa, R. (2001b). On minimizing training corpus for parser acquisition. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-01)*, Toulouse, France.

Hwa, R. (2004). Sample selection for statistical parsing. *Computational Linguistics*, 30(3):253–276.

Jelinek, F., Lafferty, J., Magerman, D., Mercer, R., Ratnaparkhi, A., and Roukos, S. (1994). Decision tree parsing using a hidden derivation model. In *Proceedings of the Workshop on Human Language Technology (HLT-94)*, Plainsboro, NJ, USA.

Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, Netherlands.

Johansson, S., Leech, G. N., and Goodluck, H. (1978). *Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for Use with Digital Computers*. University of Oslo.

Johnson, M., Geman, S., Canon, S., Chi, Z., and Riezler, S. (1999). Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*.

Jones, R., Ghani, R., Mitchell, T., and Riloff, E. (2003). Active learning for information extraction with multiple view feature sets. In *Proceedings of the ECML-03 Workshop on Adaptive Text Extraction and Mining*, Cavtat and Dubrovnik, Croatia.

Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(4).

Klein, D. and Manning, C. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-03)*, Sapporo, Japan.

Kučera, H. and Francis, W. N. (1967). *Computational analysis of present-day american english*. Brown University Press.

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *IEEE Transactions on Information Theory*, 22:79–86.

Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 148–156, New Brunswick, NJ, USA.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, Dublin, Ireland.

Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151.

Magerman, D. (1995). Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, Cambridge, MA, USA.

Manning, C. D. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press, Cambridge, MA, USA.

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

McCallum, A. and Nigam, K. (1998). Employing EM and pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, Madison, WI, USA.

McCallum, A., Rohanimanesh, K., and Sutton, C. (2003). Dynamic conditional random fields for jointly labeling multiple sequences. In *NIPS-03 Workshop on Syntax, Semantics, and Statistics*, Vancouver and Whistler, BC, Canada.

Melville, P. and Mooney, R. J. (2004). Diverse ensembles for active learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML-04)*, Banff, AB, Canada.

Mitchell, T. (1982). Generalisation as search. *Artificial Intelligence*, 18(2):203–226.

Moore, D. S. and McCabe, G. P. (1999). *Introduction to the Practice of Statistics*. W. H. Freeman, New York, 3rd edition.

Muslea, I. (2002). *Active learning with multiple views*. PhD thesis, University of Southern California.

Noreen, E. W. (1989). *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. John Wiley & Sons.

Oepen, S., Toutanova, K., Shieber, S., Manning, C., Flickinger, D., and Brants, T. (2002). The LinGO Redwoods treebank: motivation and preliminary applications. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, Taipei, Taiwan.

Osborne, M. and Baldridge, J. (2004). Ensemble-based active learning for parse selection. In *Proceedings of the 5th Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-04)*, Boston, MA, USA.

Pollard, C. and Sag, I. A. (1988). *Information-based syntax and semantics: Vol. 1: fundamentals*. Center for the Study of Language and Information, Stanford, CA, USA.

Prins, R. and van Noord, G. (2003). Reinforcing parser preferences through tagging. *Traitement Automatique des Langues*, 44(3):121–139.

Ramshaw, L. A. and Marcus, M. P. (1995). Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora (WVLC-95)*, Boston, MA, USA.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-96)*. Philadelphia, PA, USA.

Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994). A maximum entropy model for prepositional phrase attachment. In *Proceedings of the Workshop on Human Language Technology (HLT-94)*, Plainsboro, NJ, USA.

Roy, N. and McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, Williamstown, MA, USA.

Samuelsson, C. (1993). A morphological tagger based entirely on Bayesian inference. *Nordiska Datalingvistikdagarna*.

Samuelsson, C. and Voutilainen, A. (1997). Comparing a linguistic and a stochastic tagger. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL-97)*, Madrid, Spain.

Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn treebank project. Technical Report MS-CIS-90-47, Department of Computer and Information Science, University of Pennsylvania.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden Markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, Lisbon, Portugal.

Schohn, G. and Cohn, D. (2000). Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning (ICML-00)*, San Francisco, CA, USA.

Schröder, I. (2002). A case study in part-of-speech tagging using the ICOPOST toolkit. Technical report, Computer Science, University of Hamburg.

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

Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology*, 46(1):561–584.

Shieber, S. M. (1985). Evidence against the context-freeness of natural language. *Linguistics and Philosophy*, 8:333–343.

Skut, W., Krenn, B., Brants, T., and Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, Washington, D.C., USA.

Tang, M., Luo, X., and Roukos, S. (2002). Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, Philadelphia, PA, USA.

Taylor, A., Marcus, M., and Santorini, B. (2003). *Building and using Parsed Corpora*, chapter The Penn Treebank : an overview. Kluwer.

Thompson, C. A., Califf, M. E., and Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, Bled, Slovenia.

Thurmair, G. (1990). Parsing for grammar and style checking. In *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, Finland.

Tjong Kim Sang, E. F. (2002). Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL-02)*, Taipei, Taiwan.

Tjong Kim Sang, E. F. and De Meulder, F. (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings*

of the Conference on Computational Natural Language Learning (CoNLL-03), Edmonton, AB, Canada.

Tong, S. and Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.

Toutanova, K. and Manning, C. D. (2002). Feature selection for a rich hpsg grammar using decision trees. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-02)*, Taipei, Taiwan.

van der Beek, L., Bouma, G., Malouf, R., , and van Noord, G. (2002). The Alpino dependency treebank. In Theune, M., Nijholt, A., and Hondorp, H., editors, *Computational Linguistics in the Netherlands 2001. Selected Papers from the Twelfth CLIN Meeting*.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Viterbi, A. (1967). Error bounds for convolutional codes and an asymmetrically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13:260–269.

Volk, M. (2000). Scaling up. Using the WWW to resolve PP attachment ambiguities. In *KONVENS 2000 / Sprachkommunikation, Vorträge der gemeinsamen Veranstaltung 5. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS), 6. ITG-Fachtagung "Sprachkommunikation"*, Ilmenau, Germany.

Voutilainen, A. (1995). *Morphological Disambiguation*. Natural Language Processing. Mouton de Gruyter, Berlin and New York.

Yamada, K. and Knight, K. (2001). A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-01)*, Toulouse, France.

Zipf, G. K. (1935). *The Psychobiology of Language*. Houghton-Mifflin, New York.

Appendix A

Dynamic Programming of Expected Backoff

Given a hidden Markov model, θ, we calculate the expected back-off count as:

$$\mathrm{E}_\theta[c(\mathbf{o})] = \sum_{\mathbf{s}} p_\theta(\mathbf{s}|\mathbf{o}) c(\mathbf{s},\mathbf{o}) \tag{A.1}$$

where $c(\mathbf{s},\mathbf{o})$ counts the number of trigrams in $\mathbf{s}$ that are not present in the labelled training sample used to estimate the model θ. This measure can be factorised into sub-sequences of length three, and thus (A.1) can be written as:

$$\sum_{t,s_{t-1},s_t,s_{t+1}} p_\theta(s_{t-1},s_t,s_{t+1}|\mathbf{o}) c_3(s_{t-1},s_t,s_{t+1}) \tag{A.2}$$

where $c_3(\cdot)$ here is a function returning one when the given trigram is present in the training sample and zero otherwise. This can be efficiently computed using the forward and backward matrices α and β, using the trigram marginal probabilities:

$$p_\theta(s_{t-1},s_t,s_{t+1}|\mathbf{o}) = \frac{\alpha_t(s_{t-1},s_t)\beta_t(s_t,s_{t+1})}{p_\theta(o)}$$

where $p_\theta(o)$ is the observation probability, which is given by $\sum_{s_{t-1},s_t} \alpha_T(s_{t-1},s_t)$.