Effective Active Learning for Complex Natural Language Processing Tasks

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Naturwissenschaften (Dr. rer. nat.) genehmigte Abhandlung

Vorgelegt von

Florian Laws

aus Stuttgart

Hauptberichter:	Prof. Dr. Hinrich Schütze
Mitberichter:	Prof. Dr. Benno Stein
Tag der mündlichen Prüfung:	30. November 2012

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2013

Abstract

Supervised machine learning is a widely used approach to natural language processing tasks. However, supervised learning needs large amounts of labeled training data, which needs to be annotated in a time-consuming and expensive process. Active learning is a strategy to reduce this annotation effort by setting up an interactive process in which the machine learning system iteratively selects data for annotation. By selecting only data that the system considers informative, this strategy promises a significant reduction of data that is needed for training.

In this thesis, we investigate the application of active learning to key natural language processing tasks. We investigate selection strategies for "informative" training examples for two key NLP tasks: named entity recognition and coreference resolution. We show that active learning can deliver a large reduction in annotation effort for these NLP tasks.

However, in cases of unfortunate initialization, active learning can suffer from slow learning progress on infrequent classes: the missed cluster effect. We show that active learning can be made resilient against this phenomenon by co-selecting examples that occur together in a natural context (e.g. a sentence). We also apply this strategy to selection of examples for coreference annotation and could demonstrate for the first time a successful active learning approach to coreference resolution.

We also monitor training progress during data annotation. We investigate a method to estimate performance without additional labeled test data. While this method is not reliable for stopping at a performance threshold, we can use it to define effective criteria to stop when performance for a given system and given dataset is close to optimal. Finally, we investigate crowdsourcing as a complementary cost reduction approach that aims to reduce the per-example cost by outsourcing annotation over the web. We propose strategies to mitigate the higher mistake rates of crowdsourcing annotators and present a successful combination of active learning with crowdsourcing.

Zusammenfassung

Überwachtes machinelles Lernen ist ein weitverbreiteter und sehr erfolgreicher Ansatz für Aufgaben der maschinellen Sprachverarbeitung. Überwachtes Lernen erfordert jedoch große Mengen an annotierten Trainingsdaten, die in einem oftmals teuren und zeitaufwendigen Prozess erstellt werden müssen. Eine Strategie, diesen Annotationsaufwand zu reduzieren ist das *aktive Lernen*: Ein interaktiver Prozess zwischen lernendem System und Annotator, in dem das System schrittweise informative Daten zur Annotation auswählt. Durch den Fokus auf für das System informative Daten soll die erforderliche Menge an Trainingsdaten reduziert werden.

In dieser Arbeit wird die Anwendung des aktiven Lernens auf zwei wichtige Aufgaben der maschinellen Sprachverarbeitung, Erkennung von Eigennamen und Koreferenzauflösung, untersucht. Es wird gezeigt, dass aktives Lernen eine erhebliche Reduktion der erforderlichen Datenmenge erzielen kann. Bei ungünstiger Initialisierung kann es jedoch zu einem verzögerten Lernen, speziell von wenig häufigen Klassen, kommen. Für Sprachdaten kann dieser "Missed-Cluster"-Effekt vermieden werden, indem mehrere Trainingsbeispiele aus einem natürlichen Kontext (z.B. einem Satz) gemeinsam ausgewählt werden. Mit dieser Strategie der Ko-Selektion kann erstmals auch die erfolgreiche Anwendung von aktivem Lernen auf die Annotation von Daten für die Koreferenzauflösung gezeigt werden.

Weiter wird untersucht, wie der Trainingsfortschritt des Systems während des Lernverfahrens überwacht werden kann. Wir stellen Stoppkriterien vor, mit denen der Auswahlprozess beendet werden kann, wenn ein für das gegebene System und die gegebenen Daten nahezu optimales Ergebnis erreicht wird.

Schließlich wird eine Kombination mit dem Crowdsourcing-Ansatz als komplementärer Strategie zur Senkung von Annotationskosten untersucht. Crowdsourcing verspricht durch Outsourcing der Datenannotation über das World Wide Web die Kosten pro annotiertem Trainingsbeispiel zu reduzieren, zieht jedoch oft fehlerbehaftete Annotationen nach sich. Es wird gezeigt, wie sich diese Fehler kompensieren lassen und dass sich somit Crowdsourcing und aktives Lernen erfolgreich verbinden lassen.

Acknowledgments

First of all, I would like to thank my supervisor, Prof. Dr. Hinrich Schütze. He challenged me over and over again and provided me with invaluable insight and guidance about experiments, scientific writing and all other aspects of research. I would also like to thank my second examiner, Prof. Dr. Benno Stein, for agreeing to referee this thesis and for his interesting comments in the late stages of writing this thesis. I'd also like to thank Prof. Dr. Bernhard Mitschang and Prof. Dr. Daniel Weiskopf for being on my examination committee.

Thanks to Google Europe for supporting my research with a Google European Doctoral Fellowship and for the exciting and interesting time I had as a research intern at Google Zurich in the summer of 2011. I would especially like to thank Katja Filippova, Keith Hall, Beate List, and the team of the NLP Research group at Google Zurich.

I'm indebted to Katrin Tomanek for allowing me to adopt her active learning software implementation, which provided a valuable foundation for running the experiments presented in some of the chapters. I am also very grateful to have had Katrin as a great co-author and source of inspiration.

There are many colleagues at the IMS that I would like to thank. Lukas Michelbacher for being a fantastic office mate. Sina Zarrieß and Alessandra Zarcone for always having time to discuss research and life over coffee. Sabine Dieterle for helping me with all the administrative things. I also greatly enjoyed the discussions with Anders Björkelund, Bernd Bohnet, Alex Fraser, Florian Heimerl, Wiltrud Kessler, Thomas Müller, Alexis Palmer, Sebastian Padó, Arndt Riester, Ines Rehbein, Christian Scheible, Jan Stöcklin, and many other colleagues at the IMS. For proofreading this thesis I would like to thank Andre Blessing, Wendy Brouwer, Thomas Müller, Christian Scheible, Charles Jochim, Kyle Richardson, and Michael Sappington.

I am grateful to my family and my friends. Without your constant support and encouragement I might not have completed this. And Hilda, I am happy to have met you and I am looking forward to starting an exciting new chapter of my life with you!

Contents

1 Introduction

The time we are living in is often referred to as the Information Age—an era where the abundant availability of information shapes our society. With the advent of computers, and especially personal microcomputers, information first became available in structured databases. However, with the mainstream success of the Internet, more and more information is contained in *unstructured text*, for example in digital collections of newspapers, scholarly articles, legal documents or—most recently—text that has been published directly to the Internet itself: in web logs (blogs), online discussion forums, or various forms of so-called user-generated content.

To make the information contained in this multitude of texts available for processing, indexing, and searching, it needs to be extracted from the text and converted into structured information. This automatic processing of information that has been expressed in a human language is the main task of *natural language processing* (NLP). In particular, the extraction of structured information from text is termed *information extraction*.

Information extraction systems are usually designed as a pipeline of processing stages that each solve subtasks on the way to the intended extraction result. While some of the subtasks are specific to the particular information one wants to extract, other such tasks have been found to be fundamental building blocks that are important to many information extraction tasks.

Among these tasks are: *sentence splitting*, the task of dividing a contiguous chunk of text into individual sentences, so that subsequent processing stages can focus on processing one sentence at a time; *part-of-speech tagging*—assigning the correct part of speech to every word—and *syntactic parsing* of the sentence structure; *named entity recognition* (NER), the detection

of names of, for example, persons, organizations or other real-world entities, and their assignment to types of entities; *coreference resolution*, determining which references to real-world entities refer to the same entity; *relation extraction*, which determines the relationship in which these extracted entities stand with respect to each other. We will focus on two of these important subtasks in this thesis: named entity recognition and coreference resolution.

Since the beginnings of the field of natural language processing in the 1960s, NLP systems have at first relied on elaborately hand-written rules and grammars to perform the tasks listed above. Development of these rules required significant effort where developers often had to be experts in both linguistics and the extraction task at hand.

From the early 1990s on, however, *machine learning* approaches to solving NLP tasks have become more and more popular. Machine learning infers patterns from *training data*, removing the need for hand-written rules. At the time of writing this thesis, machine learning approaches are the dominant approach to solving NLP problems. In particular, the *supervised* machine learning approaches have proven very successful. Supervised machine learning uses training data that has been *annotated* with the intended output (e.g. identified named entities or part-of-speech tags). The machine learning system can then learn a model which is used to predict this output on new data.

Machine learning relieves us of having to develop rules and grammars by hand, in turn presenting us with the need to provide *annotated training data*. This is a step forward from rule-writing, but for good performance large amounts of training data are needed. The effort to create the amount of training data that is required to train a supervised learning system can be prohibitive for applying NLP systems to new and diverse text domains, such as the various styles of user-generated content on the World Wide Web. This is called the *data acquisition bottleneck*.

Strategies against the data acquisition bottleneck

Active Learning One technique that has been developed to deal with the data acquisition bottleneck is *active learning* (AL), which aims to reduce the amount of training data needed for learning by focusing on data that is "maximally useful". In this learning scenario, data annotation is performed in an interactive way, such that the learning system chooses the data that should be annotated. By selecting only "useful" data instances, the amount of annotated data that is needed for good performance can be reduced significantly.

The bulk of active learning research has been on how to choose this data, often using a notion of the system's confidence in its decisions in the *uncertainty sampling* framework or the disagreement between different versions of the system in the *query-by-committee* framework. However, active learning using these simple strategies is not always robust and can lead to slow learning progress for rare classes. For complex NLP tasks, like coreference resolution, data selection strategies specifically tailored to the task are needed.

For the practical application of active learning in NLP, data selection is not the only problem that needs to be solved. Users want to be able to monitor progress during an annotation campaign. They want to know if their system's performance is sufficient for use, and they want to be sure that they don't waste their effort on annotating examples that no longer improve a system's performance.

Crowdsourcing A complementary strategy for alleviating the data acquisition bottleneck is reducing annotation costs by using *crowdsourcing*. Crowdsourcing, the outsourcing of data annotation tasks over the web to a large crowd of potential workers, has recently become interesting as a low-cost way of performing NLP data annotation. Combining active learning, which promises low cost by reducing the amount of work that needs to be performed, with crowdsourcing, which promises low cost per work item, has the potential to yield significant reductions in data annotation costs.

However, annotation by crowdsourcing to effectively anonymous and untrained annotators has the challenge of a high rate of annotation errors. This requires us to develop strategies for handling annotation errors in an active learning process—strategies that may also be applied to other human annotation settings, which are never completely error-free.

The remainder of this introduction section will briefly describe the contributions of this thesis and give an outline of the thesis structure.

1.1 Contributions

We study the application of the active learning paradigm to two important NLP tasks: named entity recognition and coreference resolution, and present a number of contributions.

- We demonstrate the effectiveness of active learning using uncertainty sampling for the named entity recognition task. An important point is resilience of the selection procedure against weak initialization. We show that selection of entire sentences can increase robustness for NER active learning and avoid the missed cluster effect that slows learning for small classes thanks to the *co-selection* effect: a natural way for exploring the sample space using entities that co-occur in the selected examples.

- For coreference resolution, we show that standard uncertainty sampling approaches to active learning are not successful. We instead propose a novel active learning strategy based on the combination of query by committee with a class balancing strategy that uses bootstrapping. This strategy again exploits co-selection by aggregating several coreference decisions into a natural unit, the neighborhood. With this strategy we could for the first time demonstrate successful active learning for coreference annotation.

- Further, we show how to monitor progress during data annotation. We investigate a method of estimating performance during annotations that does not require extra annotated data for classifier testing. We also propose effective criteria to stop the data annotation process close to the peak of performance that can be achieved with a given system on a given dataset.

- Finally, we investigate a complementary approach to tackling the data acquisition bottleneck: obtaining low-cost data annotation by crowdsourcing it from workers on the Internet. This approach of low cost per example is complementary to the active learning that aims to reduce the number of examples. By combining both we can attain further cost savings in data annotation. However, crowdsourced annotations have a higher rate of mistakes than those provided by trained annotators. We demonstrate strategies to mitigate this and present a successful combination of active learning with crowdsourcing.

1.2 Structure of Thesis

Chapter 2 This chapter contains background information necessary to understand the remainder of this thesis. Section 2.1 will introduce supervised machine learning and the necessary formal background. We will also briefly describe the learning models that we are using for experiments, decision trees in section 2.1.2 and maximum entropy models in section 2.1.3. Next, in section 2.2 we will introduce active learning, with a focus on the uncertainty sampling (section 2.2.2) and query-by-committee (section 2.2.3) strategies for example selection, which will be the basis for our application of AL to natural language processing tasks.

Chapter 3 This chapter introduces the two NLP tasks that we will apply active learning to: named entity recognition, which we will describe in section 3.1, and coreference resolution, described in section 3.2. We will introduce the problem that these tasks are going to solve, describe common machine learning solutions and describe how to evaluate the performance of a system solving these tasks.

Chapter 4 In this chapter, we investigate active learning strategies for the two example NLP tasks named entity recognition and coreference resolution. We will compare selection strategies for NER in section 4.1 and demonstrate the effectiveness of AL for this task. In section 4.2, we investigate one challenge for active learning: the *missed cluster effect* that can lead to slow learning progress for classes that are sparsely represented in the data. We present how this problem can be avoided by choosing an appropriate size of the data examples to be annotated, thereby exploiting the *co-selection effect*.

Finally, in section 4.3 we apply AL to coreference resolution. We present a novel example selection strategy that is specifically tailored to the challenges of coreference annotation and demonstrate the first successful application of AL for coreference resolution. The contents of this chapter are based on Laws and Schütze (2008), Tomanek et al. (2009), and Laws et al. (2012).

Chapter 5 We then turn to monitoring the progress of data annotation. In section 5.1, we investigate an approach that estimates the performance of a machine learning classifier trained on a dataset collected by active learning without having to annotate extra labeled data for evaluation. We also show in section 5.2 that it is advisable to stop data annotation when peak performance is reached and present criteria to stop annotation at that point in order to avoid unnecessary annotation. The contents of this chapter are based on Laws and Schütze (2008).

Chapter 6 In this chapter, we will combine data selection by active learning with low-cost annotations obtained by crowdsourcing. A software system that

manages annotation workflow and AL example selection on a popular crowdsourcing platform is presented in section 6.2. Strategies dealing with the fact that crowdsourced annotations are likely to contain mistakes are presented in section 6.3. We show the effectiveness of the system with these strategies in section 6.4. We further investigate the interplay of AL example selection and annotation quality in sections 6.6 and 6.7. The contents of this chapter are based on Laws et al. (2011).

Chapter 7 In the conclusion, we will summarize the contributions of this thesis and will end with an outlook into questions for further research.

2 Background

This chapter will give a brief introduction on two important techniques for this thesis: supervised machine learning and active learning.

2.1 Supervised Machine Learning

Machine learning is concerned with designing systems whose performance on a task improves with experience (Mitchell, 1997). In prediction learning, a system learns to return a prediction y when given some input data x. The experience comes in the form of *training data* which is presented to the learning procedure beforehand. Somewhat more formally, the learning procedure finds a function $f : \mathcal{X} \mapsto \mathcal{Y}$ that, given the input data x, returns the prediction y.

In natural language processing, we are usually interested in predictions that assume a number of discrete categories (e.g. parts-of-speech). This is called classification learning, and the outputs, $y \in \mathcal{Y}$, are called *classes* or *labels*. If the number of possible classes is 2, we call the classification *binary classification*, otherwise we call it *multiclass classification*. Some classifier types can be directly employed for multiclass classification, while others are only suitable for binary classification. Binary classifiers can, however, be combined to solve multiclass problems (Manning et al., 2008, p. 282).

Supervised machine learning finds the classification function f by looking at *labeled training data*, which is a set of input data together with the desired output labels $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{m}$. The *learning algorithm* uses this data to find the classification function that matches the training data. Often we apply the

term for a learning algorithm (e.g. *decision tree*) to the resulting classifier as well, depending on the context.

The classification function can often be expressed as a mathematical function or procedure, the *model*, that gets instantiated with a set of *model parameters* θ. Usually the developer of a natural language processing system chooses a particular class of functions as the appropriate model. The learning algorithm then just needs to estimate the model parameters θ to fit the data. Common approaches to this are to estimate the parameters directly from the data (e.g. as in the Naive Bayes method, cf. Russell and Norvig, 2003, p. 718) or to optimize the parameters for some criterion, for example to find the minimal error of the resulting classification function on the training set.

2.1.1 Feature Extraction

Classification methods do not work directly on language data. Instead, the data needs to be preprocessed by a *feature extraction* function F. For most classifiers, this is a function $F : \mathcal{X} \mapsto \mathbb{R}^k$ that maps an input example to a vector that encodes one feature per position, where every feature describes some property of the input data. The output space $\mathbb{R}^k$ is called the *feature space*. The choice of features is an important step in designing a successful classifier for a natural language processing task.

As an example, for a named entity recognition task, where we want to decide whether words are a part of names, we split sentences into words and then define feature functions over the words. For the word $w =$ "James", possible features could be

$$f_j(w) = \begin{cases} 1 & \text{if the word equals "James"} \\ 0 & \text{else} \end{cases}$$

or

$$f_j(w) = \begin{cases} 1 & \text{if the word starts with an uppercase letter} \\ 0 & \text{else} \end{cases}$$

These individual features are then combined into a feature vector: $\vec{x} = (f_1(w), \ldots, f_k(w))$. The range of the individual feature functions f depends on the classifier that is used. For example, a decision tree classifier can deal with arbitrary categories as values of a feature function, while linear classifiers require numeric values. In fact, some learning methods work best with binary features that assume only the values 0 and 1, so it is quite common to only use binary features. Depending on the learning method, it may be important to constrain the number of features or to check that features do not correlate strongly with each other (e.g. for the Naive Bayes method). In this case *feature selection* techniques are often applied to reduce the number of features before training (cf. Manning et al., 2008, ch. 13.5). Other learning methods handle large numbers of features more gracefully (e.g. the maximum entropy method, see section 2.1.3).

2.1.2 Decision Trees

An early machine learning method is the use of decision trees (Breiman, 1984). A decision tree classifier is structured as a tree, where all non-leaf nodes contain a test of one data attribute, or feature. Starting from the root node, we check the specified feature of the example we want to classify. Depending on the feature value, we then follow the edge labeled with the matching value to the next node. We then repeat this process until we reach a leaf node. The leaf node contains the label with which the example should be classified. In classification learning, this label is a discrete class label. Decision trees can also be used for estimating conditional probabilities by allowing leaf nodes to contain instances with differing class labels. The probability is then simply the ratio of instances carrying a particular label.

A decision tree subdivides the feature space into a hierarchical partitioning whose boundaries are aligned with the axes of the feature space. Unlike linear models (such as the maximum entropy model, see section 2.1.3), which define a linear function over the feature space, decision trees are capable of model-

ing functions that are non-linear over the feature space, so they can model classification functions that have non-linear boundaries between the classes.

A nice property of decision trees is that their decisions are easily interpretable: the path taken from root node to leaf node is simply a conjunction of the feature values that were used to make that decision. For NLP problems, decision trees have fallen out of favor in recent years, but they are still quite competitive in certain tasks such as part-of-speech tagging (Schmid and Laws, 2008) and coreference resolution (Kobdani et al., 2011). We will use decision trees as part of the coreference resolution system of Kobdani et al. (2011) in section 4.3.

During training, decision trees are constructed top-down by starting with a single root node and assigning the entire training set to it. One attribute is then chosen to split the set into several regions according to the values of the attribute. These regions are assigned to one child node each. The regions are then in turn further subdivided until a *stopping criterion* is reached.

The choice of the attribute that is used to split a region is determined by a *splitting criterion* that determines how well an attribute can be used to classify the current region of the training set. A common splitting criterion is *information gain*, which is based on the information theoretic measure of entropy

$$H(S) = \sum_{y \in \mathcal{Y}} -p_S(y) \cdot \log p_S(y)$$

where S a subset of the training set and $p_S(y)$ is the proportion of the examples in S that have label y. $H(S)$ is 0 when all the examples in S have the same label, and is maximal when all the labels have the same proportion in S. The information gain is then defined as the reduction of entropy when picking a feature A

$$\mathit{InformationGain}(S, A) = H(S) - \sum_{v \in V(A)} \frac{|S_{A=v}|}{|S|} \cdot H(S_{A=v})$$

where $V(A)$ is the set of possible feature values for A and $S_{A=v}$ is the subset of S in which all instances have the value v for feature A. In other words, this

splitting criterion tries to find an attribute that partitions S into subsets in which as many instances as possible have the same label.

If features with categorical (instead of just binary) values are used, the information gain criterion prefers attributes with a large set of values, which may lead to bad generalization power. Quinlan (1986) therefore proposes to normalize the information gain measure with a measure of how widely an attribute would split a training portion as the *gain ratio* measure:

$$GainRatio(S, A) = \frac{InformationGain(S, A)}{SplitInformation(S, A)} = \frac{InformationGain(S, A)}{-\sum_{v \in V(A)} \frac{|S_{A=v}|}{S} \log \frac{|S_{A=v}|}{S}}$$

If the tree grows too large and has too fine-grained attribute tests, its generalization performance suffers from overfitting the training data. However, it has been found that sometimes several splits that yield no apparent error reduction have to be performed, but then a next split yields a substantial reduction in error (Bishop, 2006, p. 665). Therefore, it is common to grow a large tree and prune it back afterwards using a criterion that balances prediction error against model complexity.

2.1.3 Maximum Entropy Models

Simple maximum entropy models

The *maximum entropy* (MaxEnt) classifier, also known as *multiclass logistic regression*[1] or the *multinomial logit model*, is a probabilistic classifier. Probabilistic classifiers model the data by a probability distribution. In the case of MaxEnt models, this is the conditional probability distribution $P(y|x)$. During classification, the classifier makes a prediction by choosing the class with the highest probability:

$$f_\theta(x) = \underset{y' \in \mathcal{Y}}{\operatorname{argmax}} P_\theta(y'|x)$$

[1] See Mount (2011) for a derivation that demonstrates the equivalence of both models.

The model parameters θ are estimated during training. Compared to other probabilistic classifiers such as Naive Bayes (Manning et al., 2008, ch. 14), maximum entropy models have the advantage that they don't assume statistical independence of the features that are used for describing the data. This allows one to engineer rich feature sets without having to worry about correlated features. The following description of the MaxEnt model is based on the tutorial of Berger et al. (1996).

The fundamental paradigm of parameter estimation in the MaxEnt model is based on the principle that if there is no other evidence, the model should be as unbiased as possible. In terms of probability distributions, that means the model should aim for the most uniform distribution that is consistent with the evidence found in the training data. Using entropy as the measure of the uniformity of a distribution, the best probability distribution $P_\theta(y|x)$ is the distribution that maximizes the conditional entropy

$$H_\theta = \sum_{(x',y') \in \mathcal{X} \times \mathcal{Y}'} P_\theta(y', x') \log P_\theta(y', x')$$

under the condition that P_θ is consistent with the evidence in the training data. This consistency requirement is modeled as a set of constraints based on feature functions and their expectations. In the case of MaxEnt models, the feature functions include both instance and class, for example:

$$f_j(x, y) = \begin{cases} 1 & \text{if the word } x \text{ equals "James" and } y \text{ is class "Person"} \\ 0 & \text{else} \end{cases}$$

The empirical expected value of f_j on the training data can be obtained by simply counting the proportion of examples for which f_j equals 1:

$$\tilde{E}(f_j) = \frac{1}{\mathcal{L}} \sum_{(x',y') \in \mathcal{L}} f_j(x', y')$$

We can then compute the model's expected value of f_j from

$$E_\theta(f_j) = \frac{1}{\mathcal{Y}} \sum_{x' \in \mathcal{L}} \sum_{y' \in \mathcal{Y}} P_\theta(y'|x') f_j(x', y')$$

For the consistency requirement, we now require that the model's expected values of the feature functions equal the empirical expected values from the training data:

$$E_\theta(f_j) = \tilde{E}(f_j)$$

As a final constraint, we require that $P_\theta(y|x)$ always fulfills the properties of a proper probability distribution. We then get a constrained optimization problem that is the core of the maximum entropy method: select

$$P_* = \text{argmax}\, H(P_\theta)$$

such that P_θ are probability distributions that fulfill the above-mentioned set of constraints.

Using Lagrange multipliers, this constrained optimization problem can be transformed into a corresponding dual unconstrained optimization problem. It turns out that this dual problem has the same solution as maximizing the log-likelihood function on the training data (Della Pietra et al., 1997):

$$\ell(\mathcal{L}, \theta) = \sum_{(x', y') \in \mathcal{L}} \log P_\theta(y'|x')$$

where P_θ has the form

$$P_\theta(y|x) = \frac{1}{Z_\theta(x)} \exp\left(\sum_i (\lambda_i f_i(y, x))\right)$$
$$Z_\theta = \sum_y \exp\left(\sum_i (\lambda_i f_i(y, x))\right)$$

To maximize log-likelihood, the partial derivatives of $\ell(\mathcal{L}, \theta)$ are needed, which can be computed as

$$\frac{\partial \ell(\mathcal{L}, \theta)}{\partial \lambda_i} = \tilde{E}(f_j) - E(f_j)$$

which brings us back to the constraints of the maximum entropy formulation.

The optimization problem can be solved using a version of iterative scaling, as proposed with the original formulation of the MaxEnt model (Berger et al., 1996), or by more recently proposed methods involving conjugate gradient or limited memory BFGS, which are faster (Daumé III, 2004).

Regularization

Chen and Rosenfeld (1999) point out that plain maximum entropy methods can suffer from overfitting. A strategy to avoid overfitting is to discourage complex models by *regularization*. Chen and Rosenfeld (1999) use a Gaussian prior for regularization. Instead of optimizing for maximum likelihood, we optimize for *maximum a posteriori* and maximize the function

$$\ell(\mathcal{L}, \theta) = \underbrace{\sum_{(x', y') \in \mathcal{L}} \log P_\theta(y'|x')}_{\text{log-likelihood}} - \underbrace{\sum_{j=1}^{k} \frac{\lambda_j^2}{2\sigma^2}}_{\text{regularizer}}$$

The partial derivatives of $\ell(\mathcal{L}, \theta)$ then become

$$\frac{\partial \ell(\mathcal{L}, \theta)}{\partial \lambda_i} = \tilde{E}(f_j) - E(f_j) - \frac{\lambda_j}{\sigma^2}$$

Conditional random fields

In many tasks, such as part of speech tagging or named entity recognition, it is useful to not just make a single prediction, but a sequence of predictions, e.g. one prediction per word for the entire sequence of words in a sentence. This way, the predictions of the surrounding context can be taken into account

and the best prediction for the whole input sequence can be found. For this purpose, sequence prediction models have been introduced.

Conditional random fields (CRFs) (Lafferty et al., 2001) are a variant of maximum entropy models that have been introduced for general relational learning, i.e. the prediction of a vector of labels $\vec{y} \in \mathcal{Y}^n$ from a vector of inputs $\vec{x} \in \mathcal{X}^n$, of which sequence prediction is a special case.

In sequence prediction, a special case of *linear-chain* CRFs is usually used. In this case, the probability of the output sequence $\vec{y} = (y_1, \ldots, y_n)$ is modeled as

$$P_\theta(\vec{y}|\vec{x}) = \frac{1}{Z_\theta}(\vec{x}) \prod_{i=1}^{n} \Psi_i(\vec{x}\vec{y})$$
$$Z_\theta(\vec{x}) = \sum_{\vec{y} \in \mathcal{Y}^n} \prod_{i=1}^{n} \Psi_i(\vec{x}\vec{y'})$$

where n is the length of the sequence and Z_θ is a normalization term to ensure that the distribution sums up to 1. This is very similar to the form of the distribution used in the simple MaxEnt model but one which is adapted to sequences: the probability now is the product over all states in the sequence. The functions Ψ_i are the *factors* of the distribution and have the form

$$\Psi_i(\vec{x}, \vec{y}) = \exp\left(\sum_{j=1}^{k} \lambda_j f_i(y_{i-1}, y_i, \vec{x}, i)\right)$$

The important difference is that in CRFs feature functions f_i can now depend on the whole input sequence $\vec{x}$ and, in addition to the output label on the current position y_i, also on the preceding output label y_{i-1}. Since this definition of Ψ_i allows one preceding output label, this model is a *first-order* linear-chain CRF. The modeled distribution can now be written as

$$P_\theta(\vec{y}|\vec{x}) = \frac{1}{Z_\theta(\vec{x})} \cdot \prod_{i=1}^{n} \exp\left(\sum_{j=1}^{k} \lambda_i f_i\left(y_{i-1}, y_i, \vec{x}, i\right)\right)$$

Training of the CRF is again done by maximizing the log-likelihood function on the training data with a regularization term to avoid overfitting. As in the simple maximum entropy model, the derivatives of the objective function contain the empirical expected value $\tilde{E}(f_j)$ of a feature and its model expectation $E(f_j)$ (see above). In CRFs, however, $E(f_j)$ must be calculated using dynamic programming with a variant of the Forward-Backward algorithm (Sutton and McCallum, 2006) that was introduced for *hidden markov models* (HMM) by Rabiner (1989) (cf. Russell and Norvig, 2003, p. 546).

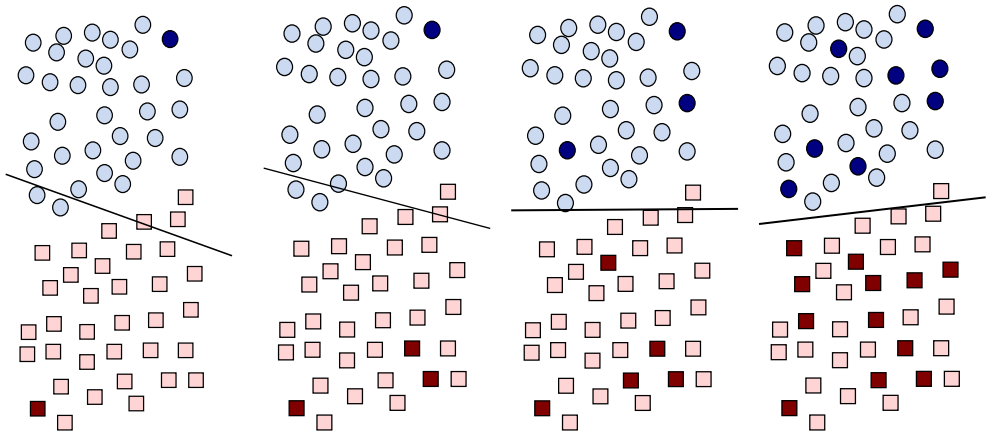
Just like for HMM sequence models, the *Viterbi* algorithm must be used for predicting the best label sequence $\vec{y}^*$ given a particular input sequence $\vec{x}$. For many active learning approaches it is also important to know the posterior probabilities of a label sequence $P_\theta(\vec{y}|\vec{x})$. Again, a variant the Forward-Backward algorithm is employed for this calculation (Sutton and McCallum, 2006; Mann and McCallum, 2007).

2.2 Active Learning

Supervised machine learning approaches work well for many natural language processing tasks, but they need large amounts of labeled training data. For many NLP tasks, this data is not readily available but needs to be created by annotation of language data by human annotators. This is a laborious task and can also be quite expensive, especially if domain experts are required.

Active learning (AL) is an annotation strategy that aims to reduce the number of training examples for classifier training. Unlike conventional, or *passive*, supervised learning, active learning does not assume a pre-existing training set but creates that training set with the help of human annotators in an interactive[2] annotation loop. Starting from an initially trained classifier, the learner

[2]Note: We are using the term "interactive" from the perspective of the learning system: Instead of getting fixed training input the learning system can now prompt the annotator for input and make different choices depending on the answer, thus allowing for some amount of interaction between system and annotator. From the more common definition of interactivity from the user's perspective, however, typical AL setups are not viewed as interactive systems, as the user has little choice but answering the prompted questions.



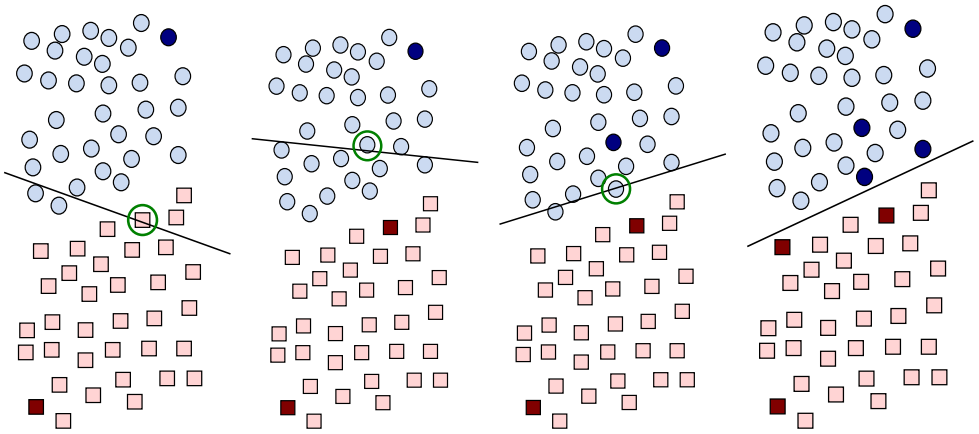
(a) 2 labels (b) 4 labels (c) 8 labels (d) 20 labels

Figure 2.1: Example illustration of random sampling

selects examples that are maximally informative (according to some criteria) and requests annotation for these examples. After annotation, the classifier is updated with the newly annotated information and further examples are chosen. By selecting only informative examples, less data needs to be annotated.

For an illustration of the general idea of (pool-based) active learning, let us imagine a simple annotation task where we want to train a classifier that classifies shapes into red squares and blue circles, as depicted in Figures 2.1 (for random sampling) and 2.2 (for active learning).

Figure 2.1 illustrates the progress of data annotation where labeled examples have been selected randomly (indicated by darker colors). A classifier that is trained on these labeled examples is visualized by the straight line, partitioning the space into the hypothesized region of circles above the line and the hypothesized region of squares below. Some random examples, for example those selected in the step from Figures 2.1(a) to 2.1(b), do not contribute much to the improvement of the classifier hypothesis. After 20 labels, the classifier still misclassifies some of the examples (e.g. the red square above the line in Figure 2.1(d)).



(a) 2 labels (b) 3 labels (c) 4 labels (d) 7 labels

Figure 2.2: Example illustration of active learning

The active learning approach does not draw examples randomly but selects informative examples based on the current classifier hypothesis. Figure 2.2 again illustrates the progress of an annotation task, this time using active learning, starting from the same two labeled examples as before. Now the current classifier is used to choose an especially informative example, highlighted by a green circle. In this illustration we defined "informative" examples as those that are closest to the separating line of the hypothesis. The examples chosen by this method lead to large updates to the hypothesis (see the large change to the line from e.g. Figures 2.2(a) to 2.2(b)) and rapid convergence to a good solution. After annotation of only 7 examples, the classifier correctly classifies all examples (Figure 2.2(d)).

How to select the "informative" example is what distinguishes the different active learning approaches. We now give a brief overview about the different approaches. Notation is adopted from the thesis of Tomanek (2010).

2.2.1 Selective Sampling

The first approach to active learning was *query construction* (Angluin, 1988). Here, the example to be annotated is not selected from existing examples but instead synthetically created by the active learning system. This approach has, however, been rejected for NLP tasks because it has been shown that it is very hard for human annotators to label synthetic examples (Baum and Lang, 1992).

Instead, the preferred approach is *selective sampling*. In this approach it is assumed that a large source of unlabeled data is available at little to no cost. The learner then picks informative examples from this source. There are two main branches of selective sampling that differ in the source of the data. In *stream-based* or *online* active learning, the learner sees each unlabeled example only once in a stream of examples. The learner can then decide to pick the example for annotation or to ignore it, but it cannot revert this decision.

In practice, it is often impractical to collect data just in time for stream-based active learning. Instead, unlabeled data gets collected beforehand and is then available for *pool-based* active learning. Here, a *pool* $\mathcal{P} \subset \mathcal{X}$ of unlabeled data is available where the learner can compare examples against each other and at each iteration select the optimal example from the pool.

As this selection progresses, a subset of the pool is labeled $\mathcal{L} \subset \mathcal{X} \times \mathcal{Y}$, which becomes the training set for classifier training. As the goal of active learning is to reduce annotation effort, an optimal strategy would be the strategy that finds the minimal subset $\mathcal{L}$ so that a model trained in this set would reach maximum performance, or, given a comparison performance, would reach this performance with minimal $\mathcal{L}$. In some annotation scenarios, the number of examples $|\mathcal{L}|$ is not an adequate measure for annotation cost (Hachey et al., 2005). In this case, the goal of active learning would be to minimize some cost function that models the annotation cost. In most publications on AL, however, the cost is simply the number of examples, $|\mathcal{L}|$.

Finding the global optimum for $\mathcal{L}$ is usually intractable, so active learning iteratively selects examples that optimize a local selection criterion in each

iteration, with the assumption that this will yield a good approximation of the globally optimal solution.

The criterion that is optimized at each selection step is usually some *utility measure* that quantifies the expected usefulness of an example if it is labeled and added to the training set, based on the current model. At each step, the best example according to the utility measure is selected.

This example is then used as a *query* and a human annotator is asked to provide the correct annotation. In experiments that evaluate new AL strategies, the annotator is often replaced by an *oracle* that always gives the correct answer (e.g. by uncovering a previously hidden label). We call this kind of setup a *simulated* AL experiment. The labeled example is then added to $\mathcal{L}$, and the model is retrained. Then, the next example can be selected. For computational efficiency, often a *batch* of the n best examples is selected in one iteration.

2.2.2 Uncertainty Sampling

Different active learning approaches differ mainly in the choice of the examples that are selected in each round. A pioneering approach that is still among the most widely used for active learning is the *uncertainty sampling* heuristic proposed by Lewis and Gale (1994). It can be used with any classifier that returns posterior class probability estimates or confidence estimates along with its predictions.

The original uncertainty sampling method by Lewis and Gale (1994) was proposed for binary classes only and picked the example whose probability was closest to the threshold between positive and negative classes in binary classification, usually 0.5.

For multiclass classification several other utility measures based on uncertainty sampling have been proposed. We formulate these measures as *confidence measures* M, such that higher values mean examples where the classifier is confident in its decision and the selective sampling procedure selects examples with *smaller* values. We will also use the term *uncertainty measure* interchangeably, even though this term hints in the other direction.

The simplest is the *MinMax* or *least-confidence* method, which is based on the probability of the most likely class and is basically an extension of the measure proposed by Lewis and Gale (1994) to multiple classes or structured outputs (Culotta and McCallum, 2005). Especially for complex structured classification this has the advantage of requiring only the probability of the best output, rather than probabilities of n best outputs as the *margin* measure or the entire probability distribution over all possible outputs as the *entropy* measure.

$$M_{\text{MinMax}}(x, \theta) = \max_{y \in \mathcal{Y}} P_\theta(y|x) \tag{2.1}$$

Other uncertainty measures take the probabilities of the remaining classes into account. The *Margin* uncertainty measures the difference between the most likely class y and the second-most likely class y'. A small margin means that a distinction between two likely classes is difficult.

$$M_{\text{Margin}}(x, \theta) = P_\theta(y|x) - P_\theta(y'|x) \tag{2.2}$$

The *1-Entropy* measure takes all classes into account. It is based on the Shannon entropy and favors examples where the classifier assigns similar probabilities to all classes.

$$M_{\text{1-Entropy}}(x, \theta) = 1 - H(P_\theta(.|x)) \tag{2.3}$$

$$= 1 + \sum_{y \in \mathcal{Y}} P_\theta(y|x) \log P_\theta(y|x) \tag{2.4}$$

2.2.3 Query by Committee

The second important family of utility functions is based on a committee of several classifiers $\mathcal{C} = \{\theta_1, \ldots, \theta_e\}$. The members of the committee are trained to be different from each other, and the selection procedure picks the examples where the predictions of the committee members disagree. Query-by-

committee (QbC) approaches differ in the way the members of the committee are created and in the way disagreements among predictions are quantified.

Agreement measures

The example on which the committee members disagree most is then selected for annotation. We therefore need an *agreement measure* to quantify this. The first two measures are constructed quite similar to the uncertainty measures, but they are computed over the number of votes of the committee members instead of class probabilities of a single classifier. The notation of these measures is again adopted from the *disagreement* measures of Tomanek (2010) but adapted to correspond to the smaller-is-preferred nature of the confidence measures for uncertainty sampling.

Abe and Mamitsuka (1998) proposed a disagreement measure akin to the *margin* uncertainty measure, which can be thought of as the "vote margin": the example with the maximum number of disagreeing decisions between the most popular and second-most popular class among the committee members is selected. Like the margin uncertainty measure, this favors examples where the distinction between two classes is hard. However, this measure has not been widely adopted by the AL research community.

More widely used is the *vote entropy* measure proposed by Engelson and Dagan (1996) that takes all classes into account. This measure takes the entropy of the distribution over the predicted classes of the committee members $\mathcal{C}$

$$M_{\text{1-VE}}(x, \mathcal{C}) = 1 + \sum_{y' \in \mathcal{Y}} \frac{V(y', x))}{|\mathcal{C}|} \log \frac{V(y', x))}{|\mathcal{C}|}$$

where $V(y', x)$ is the number of committee members that predicted y' for example x. Again, similar to the entropy uncertainty measure, this favors examples where the votes are evenly split across all classes. To our knowledge, an analogue to the least-confidence measure for QbC has not been proposed in the literature.

The above measures look at just the labels without taking into account the confidence of the decisions of the individual committee members. When using probabilistic classifiers, the committee member's predictions of the class probabilities can be incorporated as well in order to make finer distinctions. To this end, McCallum and Nigam (1998) propose the *KL divergence to the mean* measure. It is based on the Kullback-Leibler divergence D, which quantifies the difference between two probability distributions P and Q:

$$D(p||q) = \sum_{y' \in \mathcal{Y}} p(y') \log \frac{p(y')}{q(y')}$$

D is non-symmetrical, so q needs to be some sort of reference distribution to which all committee members' class distributions can be compared. McCallum and Nigam (1998) choose the mean of the committee members' class distributions as the reference:

$$p_{\text{avg}}(y'|x) = \frac{\sum_{\theta_i \in \mathcal{C}} p_{\theta_i}(y'|x)}{|\mathcal{C}|}$$

The utility function based on the KL divergence to the mean then becomes:

$$M_{\text{1-KLM}}(x, \mathcal{C}) = 1 - \frac{1}{|\mathcal{C}|} \sum_{\theta_i \in \mathcal{C}} D(p_{\theta_i}(y||x)) p_{\text{avg}}(y||x)$$

Other possible divergence methods involve selecting examples for which the F-Score differs (Ngai and Yarowski, 2000), and adaptations of the various uncertainty measures for committees of probabilistic classifiers (Körner and Wrobel, 2006).

Committee creation

Finally, for the query-by-committee approach a way to create the members of the committee is needed. This method needs to ensure diversity of the member classifiers, so they in fact give answers that disagree for important examples.

Seung et al. (1992) pioneered the query-by-committee approach for stream-based active learning. In their approach, the members of the committee are sampled directly from the *version space*, which is the space of all models that correctly classify the training data $\mathcal{L}$. When examples are added to $\mathcal{L}$, the number of models that correctly classify $\mathcal{L}$ potentially gets smaller. By choosing examples where the committee members disagree, the version space can be reduced considerably, up to halving the version space in each iteration. Since the version space can be quite large, especially when there are many features for the classifiers, sampling directly from the version space is infeasible. Gilad-Bachrach et al. (2005) therefore propose to intersect the version space with the version space of just the current unlabeled example x to obtain a smaller version space for sampling. Argamon-Engelson and Dagan (1999) and McCallum and Nigam (1998) create the committee members by sampling the individual model parameters independently from approximate distributions.

In practical applications of query-by-committee AL, computationally less expensive heuristic methods of committee construction are often employed. Abe and Mamitsuka (1998) introduced *Query-by-Bagging* and *Query-by-Boosting*, which are committee construction techniques inspired by earlier ensemble learning methods called *bagging* (Breiman, 1996) and *boosting* (Freund, 1995). Query-by-Bagging constructs the committee member by drawing $|\mathcal{C}|$ subsamples uniformly from the labeled data $\mathcal{L}$ and training one committee member on each subsample. Query-by-Boosting draws the subsamples according to a re-weighting distribution that incorporates the misclassification rate of each example. The resampling distribution as well as the weights of the committee members get chosen using the AdaBoost algorithm (Freund and Schapire, 1995).

While query-by-committee AL methods are computationally more expensive than uncertainty sampling approaches, the heuristic query-by-committee construction methods have the advantage that they can be applied to any base learning algorithm, even those that do not have reliable confidence estimates or even have no confidence estimates at all.

2.2.4 Other Active Learning Strategies

Other active learning strategies have been proposed in the literature that have interesting theoretical properties, but place some restrictions on the base learning methods they can be applied to or are too expensive computationally to be used in practice. For completeness, I will briefly summarize them here.

Expected model change

The first approach is based on selecting the example that would change the model the most if we had its label. Settles et al. (2008b) implement this approach as *expected gradient length* for models that are trained using gradient-based optimization. Here, the method selects the example that would result in a training gradient that has maximal length. Since the true label is unknown before selecting the example and issuing a query, the length gets calculated as an expectation over all possible labelings. Settles et al. (2008b) demonstrate good empirical performance, but this approach is expensive if the feature space and the number of classes are large. For the support vector machine (SVM) classifier, Tong and Koller (2002) show that this is equivalent to querying the instance that is closest to the separating hyperplane, which is in turn equivalent to the least-confidence uncertainty sampling strategy.

Expected error reduction

Another approach is to select the example that is most likely to reduce the model's error the most. For each example and all its labels (x, y), the expected error of a model $\mathcal{L} \cup (x, y)$ is estimated on the remaining unlabeled instances $\mathcal{P} \setminus x$. The example that minimizes the expected error then gets selected for annotation. The expected error can be computed using different loss functions. When using the expected log-loss, the selection becomes equivalent to reducing the expected entropy over labelings of $\mathcal{P}$, as proposed by Roy and McCallum (2001). Zhu et al. (2003) combine this approach with semi-supervised learning, with good results. However, the big drawback is that this approach is very ex-

pensive. In each iteration, a separate model must be trained for every possible label of every instance in the pool. Even when sampling from the pool (Roy and McCallum, 2001; Zhu et al., 2003) or using approximate training (Guo and Greiner, 2007), this approach is usually impractical.

A simplification of it is *variance reduction*, where the expected future error is decomposed into its noise, bias, and variance terms. The selection algorithm then tries to minimize only the variance term. Cohn (1996) and Cohn et al. (1996) introduced this approach, and Schein and Ungar (2007) and Settles and Craven (2008) extended it to logistic regression and conditional random fields, respectively. Still, these approaches are much more expensive than uncertainty sampling or even query by committee. We will therefore focus on uncertainty sampling and query by committee in this thesis.

2.3 Summary

In this chapter, we have introduced supervised machine learning for classification and active learning. Supervised learning learns a classification function from labeled training data. We have briefly presented three important learning techniques, which are used for the experiments in the remainder of this thesis: decision trees, maximum entropy models, and conditional random fields.

We also gave an introduction into the active learning technique, which is central to this thesis. We focus on pool-based active learning, where examples that are useful for classifier training are iteratively sampled for annotation from a large collection of unlabeled examples, the pool. Pool-based active learning approaches differ in their notion of example utility. The most common approaches are uncertainty sampling and query by committee. Uncertainty sampling approaches use confidence estimates from the classifier. Query-by-committee approaches use a committee of several diverse classifiers and select examples where the predictions of these committee members differ.

3 Applications for Machine Learning in NLP

This chapter introduces the two main tasks that we will use in our active learning experiments: named entity recognition (NER) and coreference resolution (CR).

The first task, NER, concerns recognizing entities mentioned in a text and categorizing them; the second task, CR, looks at identifying which of these entities share the same referent. Both steps are a necessary prerequisite for further processing tasks, such as relation extraction, and are therefore fundamental components in most information extraction pipelines. We will define the tasks, introduce the standard machine learning approaches to solve them, and describe the standard ways of evaluating the performance of a system for these tasks.

3.1 Named Entity Recognition

Named entity recognition is the task of identifying the names of real-world entities such as persons or companies that are mentioned in a text. For example, in the sentence[1]

> Media tycoon Barry Diller$_{\text{PER}}$ on Wednesday quit as chief of Vivendi Universal Entertainment$_{\text{ORG}}$, the entertainment unit of French$_{\text{GPE}}$ giant Vivendi Universal$_{\text{ORG}}$ whose future appears up for grabs.

[1] Sentence taken from the ACE-2005 corpus.

"Barry Diller" refers to an entity of type "person (PER)", "Vivendi Universal Entertainment" and "Vivendi Universal" refer to entities of type "organization (ORG)", and "French" refers to an entity of type "geo-political entity (GPE)".[2]

The named entity recognition task can therefore be loosely described as recognizing proper names in text and categorizing recognized mentions according to a set of predefined types. As Nadeau and Sekine (2007) point out, however, there is no formal definition of a named entity since specific tasks differ in terms of the entities or types that need to be recognized. For example, some tasks require identifying temporal expressions (e.g. "June 2012"), whereas others concern numeric expressions that might relate to money or other standard units of measurement. Following conventions set by a series of competitions associated with the Message Understanding Conference (MUC), proper name expressions are called "enamex", temporal expressions are called "timex", and numerical expressions "numex". In this thesis, we focus on proper names, i.e. the "enamex" set of MUC, but we use more recent datasets from the ACE and CoNLL-2003 corpora.

Named entity recognition as a separate NLP subtask grew out of the Message Understanding Conferences. These conferences featured *shared tasks*, competitions where several research teams evaluated their system's performance on a common dataset. These shared task were concerned with *information extraction* (IE), where structured information about, for example, company activity is mined from unstructured texts such as news reports. The MUC organizers recognized that identifying the companies, people, etc., mentioned in the texts is a major building block for further extraction processing. The MUC-6 competition (Grishman and Sundheim, 1996), therefore, for the first time defined a named entity recognition subtask, consisting of the recognition of "numex" and "enamex" expressions with a set of "person"/ "organization" / "location" entity types. Together with the follow-up conference MUC-7, (Chin-

[2]In the ACE task definition (Doddington et al., 2004), a geo-political entity is a location with a government, e.g. a country. In other NER task definitions like MUC or CoNLL-2003, there is no distinction between locations (LOC) with or without a government.

chor, 1999), it defined the "classical" set of entity types for general-purpose named entity recognition. Other notable shared evaluations of named entity recognition include CoNLL-2002 and -2003 (Tjong Kim Sang and De Meulder, 2003), which included evaluation on several languages, and ACE (Doddington et al., 2004), which featured several different text genres. Recently, research interest in named entity recognition has also come from the biomedical field, where the goal is to identify named entities of very different types, such as proteins or genes, for example in the BioCreative shared evaluations (Hirschman et al., 2005; Smith et al., 2008).

In more general text domains, there has also been recent research interest in moving from the rather coarse MUC-style categories to a more finer-grained set of categories. Some approaches simply use extended sets tailored to specific needs (e.g. Maynard et al., 2001), while others go for open sets of classes that are themselves mined from the data, thereby blurring the border between NER and relation extraction (Etzioni et al., 2005; Zhu et al., 2005).

3.1.1 Approaches to Named Entity Recognition

The first approaches to implementing named entity recognition systems often consisted of simply matching name lists (*gazetteers*) with the text. While simple, this approach suffers from several drawbacks. List-based approaches cannot generalize to unknown names and inflected forms of names, so recall will be low. Furthermore, when assigning found entities into types, list-based approaches cannot distinguish to which class an ambiguous name belongs. For example, a list-based system cannot tell if "London" refers to the English capital city or to Jack London the author. This is a problem especially for biomedical NER, where abbreviations are often highly ambiguous (Shen et al., 2003). The next step involved rule-based approaches, where typical patterns of names in text were encoded as a set of detection rules. These rules can generalize to unknown names and can also help to solve the entity type disambiguation problem. However, rule-based systems need to be developed by domain experts

with additional knowledge of linguistics, which makes development a difficult task.

Most of the current approaches to NER use supervised machine learning techniques, where a classifier is trained to detect and categorize entities. Development of an NER system then consists of developing an appropriate classifier and creating training data which is used to train the classifier's model. In principle, the classifier can stay the same for different domains and even languages; only the model needs to be retrained on annotated data. These annotated data, however, need to be created for each domain. Active learning can be used to minimize the amount of annotated data.

NER as sequence tagging

Machine learning approaches usually cast the NER problem as a *sequence tagging* problem. For each word in a sentence, a trained classifier tries to predict a *tag* that encodes whether the word is part of a named entity and—if so—the type of the entity.

The most common set of tags follows the BIO encoding scheme (Jurafsky and Martin, 2008, p. 487): a word that is the first word of an entity of type T gets tagged with a "B-T" (begin) tag. The following words that continue the same entity have "I-T" (inside) tags. Words that are not part of any entity get tagged with "O" (outside) tags. An entity ends where the tag sequence changes from B or I tags to O or to another B tag. An example of tagging with the BIO scheme is shown in Figure 3.1. The BIO scheme is the most widely used encoding for NER, but Ratinov and Roth (2009) suggest that extended encodings may deliver improved performance. On the other hand, it is often possible to reduce the tag set to just I and O tags, with the beginning implied by a change from O to I or from I-T1 to I-T2. The reduced I/O-tag space helps with easier annotation and faster classifier training at the expense of being unable to model directly adjacent entities of the same type. However, when these are rare, this simplification can be a useful trade-off.

Media$_{\text{O}}$ tycoon$_{\text{O}}$ Barry$_{\text{B-PER}}$ Diller$_{\text{I-PER}}$ on$_{\text{O}}$ Wednesday$_{\text{O}}$ ...

Figure 3.1: Example of BIO encoding for named entity tags

The tag sequences in the training data are then used to train a statistical classifier. Different approaches to classifier implementation have been proposed in the NLP literature. Bikel et al. (1997) pioneered the use of sequence tagging for NER using a hidden markov model (HMM). Borthwick et al. (1998) introduced maximum entropy models for NER. Currently, conditional random fields seem to be the most popular approach for NER (McCallum and Li, 2003; Finkel et al., 2005), while structured perceptrons offer an interesting alternative (Ciaramita and Altun, 2006; Ratinov and Roth, 2009).

Features

For training, every token in the input is converted into a feature vector. Common features for named entity recognition according to Jurafsky and Martin (2008) are:

Lexical item The token itself, sometimes also a stemmed version of the token.

Shape An abstraction of common capitalization and orthography patterns. Possible values could be "Lowercased", "Capitalized", "Mixed Case", or "Contains hyphen", "Contains digits", etc., or some summarized encoding of the word's capitalization pattern (e.g. the word "Barry" would get a shape encoding of "Aa", or "x-ray" would get "a-a". This encoding is due to Collins, 2002).

Affixes Suffixes and Prefixes of the token.

Part-of-speech The part-of-speech tag of the token (if available).

Name lists Presence of the token in a list of known named entities.

Context features All of these features can additionally be applied to tokens that precede or follow the current token in a certain window for additional information about the context.

NER approaches used in this thesis

In this thesis, we use two NER systems. The first system is used for active learning experiments where we select single tokens for annotation (sections 4.1, 4.2 and 5). With annotations on single tokens, we do not get complete annotated sequences, and it is therefore very unnatural to train a sequence model on it. For some experiments, such as the performance estimation we investigate in section 5.1, we also need probability estimates on the labels of individual tokens, not entire sequences. For these experiments we use an NER system based on a logistic regression (a.k.a. maximum entropy) classifier. We use the Bayesian Binary Regression (BBR) package (Genkin et al., 2007) as the classifier implementation. More details about the systems can be found in section 4.1.2.

For experiments where we select entire sentences (sections 4.2 and 6), we use a named entity recognizer that is based on a CRF and is due to Tomanek et al. (2007). It uses a standard baseline feature set and the MALLET library (McCallum, 2002) for the core CRF implementation.

3.1.2 Evaluation of NER Systems

Evaluating the performance of an NER system is usually done on a separate test dataset. This is a random sample of the data that has been annotated and has not been used for any training or tuning of the system. This data is classified by the system, and the entities that are found by the system (the response) are compared to the reference entities from the annotation.

For (part-of-speech) tagging problems, the common performance measure is *accuracy*, defined as the percentage of correct tags. Although NER is usually cast as a tagging problem, this measure is not useful for evaluating NER performance. The reason for this is that only a small portion of the words in a text

are named entities, so the majority of tokens have "O" tags. A system that was evaluated using the accuracy measure could achieve high performance figures by predicting only "O" tags —i.e., no entities at all—but this system would not be very useful.

Because of this, NER is usually evaluated using the information-retrieval-inspired *precision* and *recall* measures.[3] *Precision* (P) is the fraction of entities in the response that are in fact correct entities according to the reference.

$$\text{Precision} = \frac{\#(\text{correct entities in response})}{\#(\text{all entities in response})}$$

Recall (R) is the fraction of correct entities that are found by the system.

$$\text{Recall} = \frac{\#(\text{correct entities in response})}{\#(\text{entities in reference})}$$

These measures can be computed by counting correct answers and errors in a contingency table:

	in reference	not in reference
in response	true positives (TP)	false positives (FP)
not in response	false negatives (FN)	true negatives (TN)

Using these counts, precision and recall can then be computed as follows:

$$P = TP/(TP + FP)$$
$$R = TP/(TP + FN)$$

These two measures trade off against each other: a conservative system that outputs only entities for which it is very confident is likely to have high precision but low recall. In the extreme case, a system that proposes *no* entities will get 100% precision but 0% recall. Conversely, a system that is less careful

[3]The following description of precision, recall, and F follows Manning et al. (2008).

and eagerly suggests entities is likely to get high recall at the expense of low precision. A system that proposes everything as the response will get 100% recall but 0% precision (except for boundary errors and wrong entity types; see below).

A combined measure that trades off both measures is the *F-measure*, a weighted harmonic mean of precision and recall

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \qquad \text{where} \quad \beta^2 = \frac{1-\alpha}{\alpha}$$

where $\alpha \in [0,1]$ or $\beta \in [0,\infty]$ control the weighting of recall versus precision. Small values of β prefer precision and large values of β prefer recall. At $\beta = 1$, the F measure is evenly balanced. This is the most common way of using the F-measure and is used throughout this thesis. In this case, the formula simplifies to

$$F_1 = F_{\beta=1} = \frac{2PR}{P+R}$$

There are two additional considerations about scoring for named entity recognition. The first is that common NER task definitions require not only being able to identify entities but also being able to assign the correct class to them. So how should we score an entity that has been identified correctly but has been assigned the wrong class? In this case, the response gets scored as a double negative: since the response lacks the entity of the correct class A but contains a spurious entity of the incorrect class B, it gets penalized with both a false negative (FN) and a false positive (FP).

The second question arises when an entity is identified and assigned to the correct class but the response entity does not have all the tokens that make up the reference entity, or when the response has some extra tokens compared to the reference. For example, consider the organization "Vivendi Universal Entertainment" from the example sentence on page 37. An NER system's response might only contain "Vivendi Universal". This case is called a partial match and is handled differently depending on the scoring scheme. Some schemes, like the CoNLL-2003 scoring scheme, are strict and penalize partial matches,

again with both FN and FP. Other schemes give partial credit for partial matches, like the ACE evaluation score (NIST, 2004) or left/right boundary matching of the JNLPBA shared task (Collier and Kim, 2004). We adopt the CoNLL-2003 strict scoring scheme.

3.2 Coreference Resolution

Another important component for natural language understanding systems is coreference resolution (CR). Coreference resolution is the task of deciding which phrases in a text refer to the same real-world entity. As an example, consider the following sentence:[4]

> [Rolls-Royce Motor Cars Inc.]$_1$ said [it]$_1$ expects its [U.S.]$_2$ sales to remain steady at about 1,200 cars in 1990. [The luxury auto maker]$_1$ last year sold 1,214 cars in the [U.S.]$_2$. [Howard Mosher]$_3$, [president and chief executive officer]$_3$, said [he]$_3$ anticipates growth for [the luxury auto maker]$_1$ in Britain and Europe.

"Rolls-Royce Motor Cars Inc.", "it", and "the luxury auto maker" are coreferent phrases that refer to the same company; "U.S.", and "U.S." are coreferent phrases referring to the same country. Likewise, "Howard Mosher", "president and chief executive officer", and "he" are coreferent phrases referring to the same person.

More formally, following van Deemter and Kibble (1995), we define, given two phrases α_1 and α_2

> *α_1 and α_2 corefer if and only if Referent(α_1) = Referent(α_2)*

where *Referent*(α) is a function that returns the entity referred to by α.

This way, the coreference relation is defined as a binary equivalence relation between two phrases. Any phrase that can enter a coreference relation is called

[4] Sentence taken from the SemEval-2010 dataset.

a *mention* or sometimes a *markable*. Usually these mentions will be nouns, pronouns or complex noun phrases. If two mentions are coreferent, we say they form a *coreferent pair*, or *coreferent link*. The first mention of such a pair is called the *antecedent*. The transitive closure of coreferent pairs forms a set of coreferent mentions that is called a *coreference cluster*, or a *coreference chain*.

Coreference resolution has been the subject of extensive research in the NLP community and has been the topic of several shared evaluations of coreference resolvers. These shared evaluations have contributed a great deal toward a common definition of the task and have led to the development of common datasets for evaluation and standard evaluation metrics.

As in named entity recognition, the Message Understanding Conferences on information extraction broke new ground in defining coreference resolution as an important information extraction subtask. MUC-6 (MUC-6, 1995) was among the first shared evaluations that defined the coreference resolution task in NLP, and the definitions introduced above come mostly from this shared evaluation. Other shared evaluations for CR include the ACE (Automatic Content Extraction) competitions (NIST, 2004, 2008) and the recent SemEval-2010 (Recasens et al., 2010) and CoNLL-2011 (Pradhan et al., 2011) shared tasks. The SemEval-2010 and CoNLL-2011 shared tasks are based on the OntoNotes corpus (Hovy et al., 2006), a large corpus with diverse linguistic annotations, including coreference, for various genres.

3.2.1 Approaches to Automatic Coreference Resolution

The earliest approaches to automatic coreference resolution were based on linguistic rules. For example, Hobbs (1986) describes a procedure for pronoun resolution that searches for plausible antecedent noun phrases (NPs) in a parse tree.

Current coreference resolvers, however, are predominantly based on machine learning approaches. One seminal paper that proposed machine learning for CR is by Soon et al. (2001). The model proposed in this paper is often called the *mention-pair* model. A *link classifier* classifies pairs of mentions whether

they are coreferent or non-coreferent. The algorithm proceeds over all mentions in the document from left to right and pairs the current mention with all possible antecedents to its left for prediction. As soon as a mention pair is determined as coreferent, the algorithm moves on to the next mention. This way a linear coreference chain is created. This way of creating a coreference chain from coreferent mention pairs is called *closest-first clustering*.[5] Apart from the classification algorithm, an important contribution of Soon et al. (2001) is a method to balance the data used for training, which we will revisit in detail in section 4.3.

Ng and Cardie (2002) improve on this algorithm by introducing a different clustering step. It uses the confidence rating of the classifier to select the *most probable* antecedent that was classified as coreferent. The resulting coreferent clusters are a *best-first clustering*.

The mention-pair model has some weaknesses (Luo et al., 2004). For example, it only determines if a candidate antecedent is likely to be coreferent given the anaphoric NP and does not take into account the other candidates. Furthermore, the information in just two paired NPs might not be sufficient to make a good coreference decision, e.g. when one of the NPs lacks information about gender. Other models have been proposed to address these issues—for example, the entity-mention model (Luo et al., 2004; Culotta et al., 2007). For an overview of different approaches to coreference resolution we refer to Ng (2010).

Nevertheless, the mention-pair model is still very much competitive with more sophisticated models, as evidenced by the recent SemEval-2010 (Recasens et al., 2010) and CoNLL-2011 (Pradhan et al., 2011) shared tasks. For this reason, we use in this thesis an implementation of the mention-pair model that is due to Kobdani et al. (2011). This implementation uses decision trees as its best performing link classifier and a mixture of best-first and closest-first clustering for the construction of the coreference clusters.

[5]The term "clustering" refers to collecting the set of coreferent mentions from individual coreferent pairs and is only vaguely related to the unsupervised machine learning technique of the same name.

3.2.2 Evaluation of Coreference Resolvers

Evaluation of the performance of a coreference system is done by comparing the coreference clusters that are returned by the system to a reference or *gold* clustering. Like in NER evaluation, scores are given as a recall measure, indicating if the system is able to identify existing coreference relations, and a precision measure that indicates if the relations proposed by the systems are correct. How these scores are computed is, however, not trivial, and several measures have been proposed, which we will now describe.

According to Luo (2005), a coreference scoring metric should fulfill two important properties:

1. **Discriminativity** The metric should be able to differentiate a good system from a bad one.

2. **Interpretability** The metric should provide an intuitive sense of how good a system is when the metric suggests that a certain percentage of coreference results are correct.

While these requirements sound trivial, early coreference evaluations used a graph-based scoring scheme that reportedly produced very unintuitive results (Vilain et al., 1995) and, from the MUC-6 conference on, was supplanted by the MUC scoring scheme proposed by Vilain et al. (1995). Even the more recent ACE scoring scheme is criticized as being difficult to interpret by Luo (2005).

After the introduction of the MUC scoring scheme, several other performance measures for coreference have been proposed that are intended to fix several shortcomings of the MUC scoring scheme and its successors. To date, no single measure has come to be the universally accepted scoring measure. We will briefly introduce the three most important scoring metrics here.

The MUC scoring metric

The MUC scoring scheme (Vilain et al., 1995) is based on the observation that a coreference cluster, i.e. an equivalence class of coreferent mentions, can be

represented as a *minimal spanning tree* of coreference links. The size $c(S)$ of a minimal spanning tree is always the minimal number of coreference links that are needed to completely define a cluster, which in turn is directly related to the number of elements in the cluster: $c(S) = (|S| - 1)$.

For each coreference cluster $S \in \{S_1, \ldots, S_k\}$ of the reference, recall can then be computed as follows: for each cluster, compute the partition $p(S)$ as the intersection of S with each cluster from the system's response $R_1, \ldots, R_m$ that overlaps with S. Recall errors are then defined as the missing links $m(S)$ that have to be added to the response to create a spanning tree that covers S. The recall for a single cluster S can then be easily expressed via the number of elements per cluster and the number of partitions:

$$Recall_{\text{MUC},S} = \frac{c(S) - m(S)}{c(S)} = \frac{(|S| - 1) - (|p(S)| - 1)}{(|S| - 1)} = \frac{|S| - |p(S)|}{|S| - 1} \quad (3.1)$$

The recall over the entire document (or corpus) is computed by summing over all clusters in the reference:

$$Recall_{\text{MUC}} = \frac{\sum_i^k |S_i| - |p(S_i)|}{\sum_i^k |S_i| - 1} \quad (3.2)$$

Precision is simply computed by switching reference and system answer:

$$Precision_{\text{MUC}} = \frac{\sum_i^m |R_i| - |p(R_i)|}{\sum_i^m |R_i| - 1} \quad (3.3)$$

As an aggregate score, the common F_1 score is used, the harmonic mean of precision and recall.

The B^3 scoring metric

The MUC score was the one of first coreference scores that produced interpretable results, but it has two important shortcomings which led to the development of other scoring methods. The first disadvantage is that since the MUC score is based on links between coreferent mentions, no credit is given to

the correct detection of *singleton mentions*, i.e. the mentions are not coreferent with any other mention in the text. The second issue is that the MUC score treats all errors equally, but some errors have worse effects than others. For example, Bagga and Baldwin (1998) argue that it is a worse error if two large coreference chains are joined by a spurious link than if small coreference chains get joined.

The B^3 metric ("B-Cubed"; Bagga and Baldwin, 1998) solves these problems by looking at each mention individually. Let $p(S)$ again be the partition of a reference cluster S with respect to the system's response. Its parts are $p(S) = \{P_1, \ldots, P_l\}$, and each $P_j \in p(S)$ is a subset of S. Recall error is then computed by counting the number of missing mentions $m_j(S)$ that are missing from each P_j. For an entire coreference cluster, the recall can then be computed by

$$Recall_{\mathrm{B}^3,S} = 1 - \frac{1}{|S|} \sum_{P_j \in p(S)} \sum_{e \in P_j} \frac{m_j(S)}{|S|} \tag{3.4}$$

$$= 1 - \frac{\sum_{P_j \in p(S)} \sum_{e \in P_j} |S| - |P_j|}{|S|^2} \tag{3.5}$$

To get a recall score for the entire document, the scores for each coreference cluster are combined using a weighted average. The full version of B^3 allows for different weights depending on the type of entity, but usually uniform weights are used and the document-wide recall is a weighted average over all clusters, which are weighted by size:

$$Recall_{\mathrm{B}^3} = \sum_{i}^{k} \frac{|S_i|}{\sum_{j=1}^{k} |S_j|} Recall_{\mathrm{B}^3,S_i} \tag{3.6}$$

As in the MUC score, precision can be computed with reference and system output flipped, and the F_1 score is again used as an aggregate score.

The CEAF scoring metric

The B^3 metric addresses the shortcomings of the MUC metric but has its own disadvantages. According to Luo (2005), systems that produce many singletons

in the response get rewarded with inordinately high precision values, while systems that tend to produce a few very large clusters get high recall values. Furthermore, recall values can be disproportionately high if the gold clustering contains many singletons.

Luo (2005) therefore proposes the CEAF measures. (Constrained Entity Alignment F-Measure) These measures define precision and recall based on an optimal one-to-one alignment $g^*(R)$ of system clusters to reference clusters, subject to a similarity function. The similarity function measures the similarity of two coreference clusters.

The optimal alignment is then used to compute precision and recall scores

$$Precision_{\text{CEAF}} = \frac{\sum_{R_j \in R^*} \text{sim}(R_j, g^*(R_j))}{\sum_i^m \text{sim}(S_i, S_i)} \tag{3.7}$$

$$Recall_{\text{CEAF}} = \frac{\sum_{R_j \in R^*} \text{sim}(R_j, g^*(R_j))}{\sum_i^m \text{sim}(R_i, R_i)} \tag{3.8}$$

where R^* is the subset of the system's responses for which $g^*(R)$ is maximal. (R^* might be a proper subset of the response clusters because it might be the case that some clusters proposed by the system cannot be aligned to reference clusters in a one-to-one alignment, e.g. if the system had proposed too many clusters.) Again, precision and recall are aggregated by the F_1 score.

The optimal alignment $g^*(R)$ is computed with respect to a similarity function, for which Luo (2005) proposes two variants. For *entity-based CEAF*, the similarity function is $\text{sim}_e(R, S) = \frac{2|R \cap S|}{|R|+|S|}$. The F-Score of entity-based CEAF can be interpreted as the ratio of entities that the system got correct. For *mention-based CEAF*, the similarity function is simply $\text{sim}_f(R, S) = |R \cap S|$. Mention-based CEAF's F-Score can be interpreted as the ratio of mentions that are in the correct entities.

CEAF addresses the weaknesses of B^3 with regard to all-singletons or single-entity solutions. However, it shares with B^3 the surprisingly high scores for singleton-rich reference datasets. CEAF in turn has some weaknesses of its own, as discussed by Denis and Baldridge (2009). Recasens and Hovy (2011) therefore propose a new coreference scoring measure, BLANC, but so far this

measure has not been widely adopted by the coreference research community. In fact no single measure has been accepted as the preferred coreference scoring measure, so coreference systems are commonly evaluated reporting all three of the (link-based) MUC, (mention-based) B^3, and entity-based CEAF measures (Denis and Baldridge, 2009).

3.3 Summary

This section introduced the two main tasks used for experiments throughout this thesis: named entity recognition (NER) and coreference resolution (CR). Named entity recognition is concerned with the identification of names of entities (i.e. persons, locations, or organizations) in a text and assigning these mentions to an entity class. The common approach to the NER tasks is to treat this as a sequence tagging problem, where each token gets assigned a tag indicating both if the token is part of a mention span and which class of entity it is part of. This tagging problem is then solved using classifiers like MaxEnt or CRFs. Evaluation is done using precision, recall, and F-Score over identified (and correctly classified) mentions.

Coreference resolution is the task of deciding which phrases in a sentence refer to the same real-world entity. The common approach to this task is the mention-pair model, which works in two stages: First, a classifier decides if two pairs of mentions are coreferent. The second stage then uses these pairwise decisions to group mentions into larger clusters of coreferring mentions. For coreference resolutions, there are several evaluation measures in use by the community, MUC, B^3, and entity-based CEAF, and therefore we will present results using all three of them.

4 Example Selection for Complex NLP Tasks

In this chapter we will investigate active learning strategies for the two NLP tasks described in the last chapter: Named entity recognition (NER) and coreference resolution (CR). We will start with NER and demonstrate the effectiveness of uncertainty sampling AL when applied to this task.

We will then investigate the missed cluster effect that slows learning for infrequent classes. We show that this issue can be avoided by choosing the appropriate size of selected examples, exploiting the information that is contained in named entities co-occurring in a sentence. We call this the *co-selection effect*.

For coreference resolution we find that simple uncertainty sampling approaches are not successful. We therefore propose a purpose-built example selection strategy for CR annotation that uses query by committee together with an explicit strategy that facilitates co-selection and class balancing. Experiments show that AL using this novel strategy is effective for CR annotation.

4.1 Comparison of Confidence Measures

We start with AL example selection strategies for named entity annotation. Since active learning is an interactive process, it is important that the time spent on retraining and example utility re-estimation is kept low. In standard non-parallel AL implementations, the annotator has to wait during retraining and utility re-estimation, so keeping this idle time low is an important factor in efficient annotation. Even in a concurrent AL scenario, which we consider

in section 6.2.2, fast retraining and re-estimation is helpful for having good utility estimates at all times.

For this reason, we focus on the uncertainty sampling class of AL selection strategies. Compared to committee-based approaches, uncertainty sampling-based approaches save on computational effort because only one classifier needs to be retrained for each cycle. Also, the computation of the uncertainty measure requires classification of the examples in the pool by only one classifier for each cycle.

The choice of the uncertainty measure is relatively straightforward for binary classification. For multiclass problems, however, there is a choice of several uncertainty measures. Schein and Ungar (2007) point out that the various uncertainty measures perform differently depending on the task, so we evaluate several uncertainty measures for NER with multiclass logistic regression (also known as maximum entropy classification).

We use the common sequence tagging approach to NER that assigns named entity tags to tokens using a simple I/O tag set (see section 3.1.1). For high sampling efficiency, we chose to select individual tokens from the unlabeled pool. This way, the active learning system can focus on exactly the token it is uncertain on. In an annotation user interface, a token selection approach would allow a very simple user interface. An annotator could simply click on one button to select the type of named entity that the token is part of (or "O" otherwise). Obviously, the token needs to be presented with some surrounding context.

For all the experiments in this chapter, we simulated annotation by querying the labels from a pre-annotated corpus.

4.1.1 Confidence Measures Recap

We now briefly review the measures used for uncertainty sampling. We formulate these measures as *confidence measures* M, so larger values denote examples that the classifier is more confident on. Consequently, the selection chooses

the examples with the *minimum* value at each cycle. We will compare three confidence measures, defined as follows:

1-Entropy

$$\begin{aligned} M_{i,\textit{1-Entropy}} &= 1 - H(P_\theta(.|x_i)) \\ &= 1 + \sum_{y \in \mathcal{Y}} P_\theta(y|x_i) \log P_\theta(y|x_i) \end{aligned}$$

where $P_\theta(y|x_i)$ is the current estimate of the probability of class y given the example x_i. (The θ represents the currently trained model used for example selection.) We use *1-Entropy* instead of entropy, so all three confidence measures will have lower values for less certain instances. *1-Entropy* favors examples where the classifier assigns similar probabilities to all classes.

Margin If y is the most-likely class and y' is the second-most-likely class, the margin is defined as follows:

$$M_{\textit{Margin}}(x, \theta) = P_\theta(y|x) - P_\theta(y'|x)$$

Margin picks examples where the distinction between two likely classes is hard.

MinMax

$$M_{i,\textit{MinMax}} = \max_{y \in \mathcal{Y}} P_\theta(y|x_i)$$

The rationale here is that a low probability of the selected class indicates uncertainty. *MinMax*, or *least-confidence*, is a measure that is more directly based on the classifier's decision for a particular example whereas the other two measures also take into account the classifier's assessment of classes that were not chosen for the unlabeled example.

4.1.2 Experiments

Setup

We compare the performance of the three uncertainty metrics by conducting a simulated active learning experiment. A simulated experiment means that instead of employing an actual human annotator, annotation is simulated by uncovering labels from a previously annotated corpus.

We used the newswire section of the ACE 2005 Multilingual Training Corpus (Walker et al., 2006) (128 documents, 66,015 tokens) as the dataset for the experiments. A subset of the documents was randomly sampled into an evaluation set that consists of 6,301 tokens. The remaining documents serve as the uncertainty sampling pool.

We start with a seed set of ten consecutive tokens randomly selected from the training pool and label it. This seed set constitutes the initial labeled training set that the initial version of the classifier is trained on. In each round of AL we select the ten examples with the smallest value of the confidence measure $M_{i,X}$ (where $X \in \{\textit{1-Entropy}, \textit{Margin}, \textit{MinMax}\}$) from the remaining pool. We then remove these examples from the pool, label them, and add them to the labeled training set. The classifiers are retrained with the new training set and the AL loop repeats. We performed 20 runs of the experiments, each with the same sampling pool but with a different seed set, randomly selected as described above.

Classifier setup We use the BBR package (Genkin et al., 2007) for binary logistic regression (also known as maximum entropy classification, see section 2.1.3) as our base classifier, with default values for all of BBR's parameters. Since we use binary classifiers but have several classes of named entities (NEs), we train separate classifiers for each NE class (PER, ORG, LOC, GPE, FAC, VEH, and WEA) and another one for the class "O" ("not an NE"). For each token we normalize the output probability of the individual classifiers so they sum up to 1 and then select for each token the class with the highest probability.

All classifiers use the same feature set, which is a standard baseline feature set. It contains the following features, which are defined over each input token:

- First character of the token is capitalized.
- Token consists of capital letters only.
- Token contains some uppercase letters, but is not made up exclusively of them.
- Token consists of digits only.
- Token consists of special characters, e.g. "&" or "-". Special characters are the "non-word" characters as defined by the Perl-compatible regular expression engine, i.e. all characters except letters, digits, and the underscore.
- Token contains the dash "-".
- Token ends with a dot.
- Token has length x in characters.
- Lexical feature: the token itself.
- Prefix feature: the first three characters of the token, set to lowercase.
- Suffix feature: the last three characters of the token, set to lowercase.

These features are generated not only for the current token but also for the preceding 2 and the following 3 tokens. For example, a feature "-2_lex_city" means that the token two positions to the left of the current token was "city", while "1_sngram_ent" means that the suffix of the following token was "ent".

Using *all* labeled training data as our fully supervised baseline results in a performance of 78.7% F-Score, which is comparable to similar studies on AL, such as Vlachos (2008). More recent publications on NER (Ratinov and Roth, 2009) report higher performance, but these systems are (i) evaluated on different datasets (e.g. the CoNLL-2002/2003 shared task data, which has fewer classes), not on the ACE dataset, and (ii) use very highly tuned complex feature sets, which often contain features that are induced from word lists or other sources of external lexical knowledge. We believe that for the comparison of active learning methods, it is more interesting to forgo the use of

external knowledge and focus on the performance differences that result from the annotation of the data in the pool.

Evaluation criteria

The goal of active learning as an annotation strategy is to reduce annotation effort. Therefore, the standard way to evaluate active learning is a comparison of the annotation effort relative to classification performance. One way to view this is *sampling complexity*, i.e. the annotation effort that is required to attain a given performance level. If experiments are done on an already existing dataset, as in our case, one particular performance level that is often used for comparison is often the performance of the *full supervised baseline*, i.e. of the system that is trained on all available data.

Conversely, one can compare classifier performance at a set amount of annotation effort. Commonly, the performance relative to the annotation effort is presented as a *learning curve*, where the x-axis shows the progress of the annotation with progressively larger labeled sets (or effort) and the y-axis shows classifier performance. Learning curves are also often compared to the learning curve of the *random sampling baseline*, which shows classifier performance trained on a progressively larger labeled set selected by random sampling. For learning curve comparison with the random sampling baseline, Haertel et al. (2008) and Tomanek (2010) suggest computing relative improvement measures, but these measures have not yet been adopted by the community.

This leaves us with the question of how annotation effort should be quantified. For the experiments on NER in this chapter we use as a cost measure the number of tokens that were queried and annotated. Recent research has called into question whether the assumption of a unit cost per annotated token is a realistic measure of annotation effort. For example, it has been claimed that human annotators take longer to annotate some sentences as opposed to other sentences (Becker et al., 2005; Hachey et al., 2005; Baldridge and Palmer, 2009). Thus, the use of wall-clock annotation time as a cost measure has been proposed. However, for simulation experiments timing information is usually

not available, with the notable exception of the timing metadata of Tomanek and Hahn (2010). For certain scenarios, such as some crowdsourced annotation experiments, there also may be a direct relation to monetary cost, which can then be used as a realistic cost measure. We will discuss crowdsourcing and associated annotation costs in greater detail in Chapter 6.

In the absence of this kind of metadata, a token-based cost measure is still the established measure of annotation effort.

4.1.3 Results

Table 4.1 shows the performance of the AL selection strategies. We find that active learning is quite successful in this experiment. After only 7% of the training data, the performance of the AL systems matched the performance of the fully supervised baseline that uses the whole dataset.

Selection	Baseline		Peak performance	
	F_1	tokens	F_1	tokens
1-Entropy	78.7	2139 (7.1%)	80.8	3460 (11.5%)
MinMax	78.7	2108 (7.0%)	80.8	3650 (12.1%)
Margin	78.7	2019 (6.7%)	81.2	3694 (12.3%)

Table 4.1: Percentage of data needed by AL to reach baseline or peak performance

We also found that the AL performance even surpasses the fully supervised baseline: at about 12% of the data, AL performance ranges between 80.8 and 81.2 F_1, which is about 2.1 to 2.5 points better than the fully supervised baseline. After reaching this peak, performance decreases again. Therefore, once the peak has been reached, the AL process should stop, even if the annotation budget has not yet been used up. We will therefore investigate methods to control and stop the AL process in Chapter 5.

Comparison of selection functions

Comparing the different selection functions, we found little difference between their performances. *Margin* performs significantly better (Student's t-test, $\alpha = 0.05$), but the difference is small ($< 1\%$ F-Score). Also, *Margin* finds a training set that yields a slightly better classifier at peak performance, with 0.4 points F improvement, compared to the alternatives, *1-Entropy* and *MinMax*.

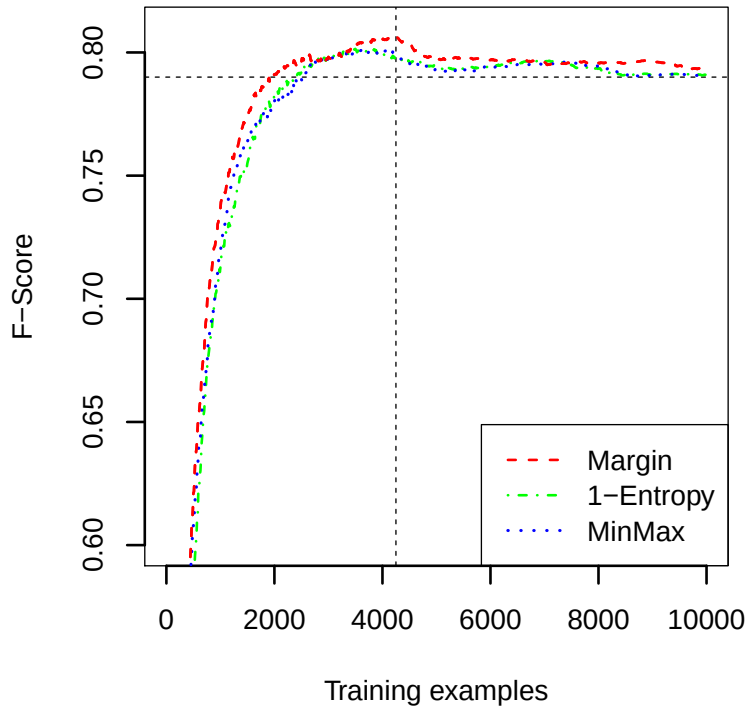


Figure 4.1: Performance as a function of number of labeled training examples used. (For better view of the important part of the curve, the x-axis is truncated at 30% of the pool.)

The ability of *Margin* to find a better peak is consistent over variation of seed sets. If we compare two AL processes (say *Margin* and *1-Entropy*) that were started with the same pool and seed set and stop both processes when they each reach their respective peak performances, *Margin* has a better peak performance of 0.3% F-Score on average (significant at $\alpha = 0.05$).

The differences between *1-Entropy* and *MinMax* are not statistically significant, except for a short start-up phase. The curves of the two measures are very similar (see Figure 4.1).

We conclude that active learning is a very efficient sampling strategy for named entity recognition. It is successful for all three selection functions we tested, with the *Margin* function slightly outperforming the alternatives. Active learning can create datasets that even surpass the performance of the full dataset. Strategies for detecting when to stop in order to reach peak performance will be discussed in Chapter 5.

Peak behavior

Encountering a performance peak that surpasses the performance we would get when labeling and using the entire data pool is a bit surprising, as one would generally expect better performance when using more data. We therefore checked in a separate experiment that this observation is not merely a result of overfitting.

We therefore conducted an experiment based on 10-fold cross-validation. We split the data into three different parts each: the sampling pool, a development test set, and another test set. We ran active learning on the pool as described above, starting from a seed set drawn from that pool. In each iteration, we measure performance on the development test set to obtain a learning curve. We observe that, again, in every fold there is a peak point where using the current selected set yields performance that is greater than using the entire pool. However, in different folds this peak point occurs with selected sets of different size. This might still be an effect of overfitting on the development test set of the respective fold, so we also evaluated the performance using that selected set on the second test set. Again, we observed that performance exceeded performance of the fully supervised baseline, so we believe that the peak behavior is not just an effect of overfitting. Schohn and Cohn (2000) and Vlachos (2008) also report similar performance peak behavior for other tasks. The authors speculate that this phenomenon is caused by noise or other

inconsistencies in the training data that are not present when only the AL-selected subset is used.

Better performance with less data means that the combination of the classifier plus the active sample selection strategy yields a system that is more favorable to the task than just the classifier alone using unbiased sampling; the combined system found a favorable bias-variance trade-off toward higher bias, which is plausible given that active learning is by design a biased sampling strategy.

This biased sampling by active learning can also be seen as a form of instance weighting in which uninformative examples get down-weighted to zero weights. By that analogy, there are other scenarios in which instance-weighting techniques are used to down-weight or even remove examples from datasets to achieve higher performance, for example down-weighting examples of the majority class in a class imbalanced dataset (Zadrozny et al., 2003).

Named entity recognition has in fact a mild class imbalance: in the ACE newswire dataset, entity tokens account for only approx. 20% of the examples. Since Ertekin et al. (2007) suggest that active learning can implicitly perform re-balancing of classes, we measured the ratios of classes in the selected set over the active learning progress. We observed that in all folds, at dataset sizes when the peak performance occurs, the class distribution in the selected set is in fact quite different than the original distribution: entity tokens make up between 40% and 60% of the examples in the selected set.

However, class balancing alone is not a sufficient explanation for the peak performance behavior as peaks in the learning performance curves do not coincide with peaks in the class distribution curves. The characterization of this "serendipitous sampling bias" (Settles, 2012, p. 52) is thus still a question for future research.

4.2 The Influence of Sampling Granularity on Selection Robustness

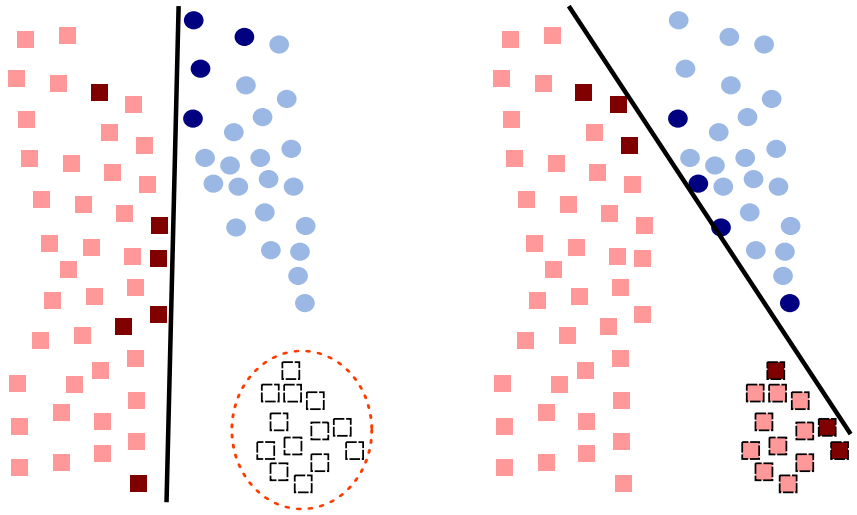
4.2.1 Missed Clusters and Missed Classes

A fundamental characteristic of AL is the fact that it constitutes a biased sampling process. Examples are not drawn independently and identically distributed (i.i.d.) from the sample space but are drawn selectively to optimize example utility with respect to (the current version of) the classifier. This is so by design, but the bias can have an undesirable consequence: partial coverage of the instance space so that entire regions are never explored by the active learner.

As a result, classes or clusters of examples within classes may be completely missed, resulting in low recall or slow learning progress. This has been called the *missed cluster effect* (Schütze et al., 2006; Dasgupta and Hsu, 2008). While AL has been studied for a range of NLP tasks, the missed cluster problem has hardly been addressed.

A special case of the missed cluster problem is the *missed class effect*, where complete classes are overlooked by an active learner. Both effects often occur because AL in its pure form is an exploitative strategy: given an initial hypothesis, it is eager to select examples that are useful to refine this hypothesis. However, there might be examples that are misclassified, but for which the classifier is very confident in its decision. These examples are selected very late in the selection progress or are not even selected at all. The classifier then fails to learn some aspect of a target concept (missed cluster), or even an entire class (missed class).

As an illustration of this, see Figure 4.2. Figure 4.2(a) illustrates how a region of the sample space can be missed by an active learner. If the seed set gets initialized with just examples from the left "squares" region and the right "circles" region, the learner may focus on examples close to the boundary between these regions (indicated by the line). Only when all examples close to this boundary are labeled may the learner discover the cluster of "square" ex-



(a) Missed cluster (bottom right) due to insufficient exploration

(b) Better exploration can avoid the missed cluster effect

Figure 4.2: Illustration of the missed cluster effect in active learning

amples to the bottom right, which may be very late in the annotation process. With better exploration of the sample space, for example through an initialization that also contains an example from the bottom right region, the system can learn a much better boundary between the two classes (Figure 4.2(b)).

The missed class effect is the result of insufficient exploration of the sample space before or during the mainly exploitative AL process. It is aggravated if there is severe *class imbalance*, i.e. one class occurs very rarely, or if the missed clusters are small. In standard uncertainty sampling or query-by-committee-based AL, exploration of the sample space is only performed at the very beginning, with the choice of the seed set that is used to train the initial version of the classifier(s). Poor seed sets that lack examples of some classes then give rise to the missed class effect.

Some researchers have proposed avoiding these issues by better creation of the initial seed set (Kang et al., 2004; Hu et al., 2010), while others propose to balance exploration and exploitation (e.g. Baram et al., 2003; Dasgupta

and Hsu, 2008; Cebron and Berthold, 2009). However, Attenberg and Provost (2011) note that even these approaches may fail for difficult concepts.

4.2.2 The Co-Selection Effect

In Tomanek et al. (2009), we explore an effect that mitigates the missed class effect for a number of natural language processing tasks: the *co-selection effect*. Many NLP tasks, such as POS tagging or NER, can be expressed as sequence learning problems. If one wants to use sequence classification methods (e.g. CRFs) for these tasks, the unit on which these sequence classifiers are trained are usually sentences, so active learning should consequently select sentences for annotation. Furthermore, the sentence has also been proposed as an annotation unit that naturally provides the context necessary for annotation (Ringger et al., 2007; Tomanek et al., 2007).

Within such sequences, instances of different classes often co-occur. Thus, an active learner that selects uncertain examples of one class gets examples of a second class as an unintended, yet positive side effect. We call this the co-selection effect.[1].

As a result, AL for sequence labeling is not "pure" exploitative AL but implicitly comprises an exploratory aspect which can substantially reduce the missed class problem. In scenarios where we cannot hope for such a co-selection, we are much more likely to have slower AL progress due to missed clusters or classes.

4.2.3 Experiments

We ran several experiments to investigate how the sampling granularity, i.e. the size of the selection unit, influences the missed class effect on a named

[1] Very recently, Li et al. (2012) proposed a different class-balancing AL technique for a sentiment classification task under the name "co-selecting". Besides the name this technique has very little similarity with our approach, but is instead inspired by co-testing (Muslea et al., 2006)

entity recognition task. AL based on token selection (T-AL) is compared to AL based on sentence selection (S-AL) on synthetic and real datasets.

Classifiers and active learning setup

For T-AL, we use the classifier setup described in section 4.1.2, while for S-AL we used a linear-chain conditional random field as described in Tomanek et al. (2007). In both cases, active learning is performed using uncertainty sampling with the margin metric as the uncertainty measure.

For T-AL, the sampling granularity is the token, while in S-AL complete sentences are selected. For S-AL, the margins of all tokens in a sentence are averaged and the aggregate margin is used to select sentences. This uncertainty measure is the most straightforward extension of the margin measure to multiple items (in our case, tokens). In either case, examples (tokens or sentences) with a small margin are preferred for selection. In every iteration, a batch of examples is selected: 20 sentences for S-AL, 200 tokens for T-AL.

Datasets

We used three datasets in our experiments. Two of them (ACE and PBIO) are based on standard datasets, while the third dataset SYN is a synthetic dataset that we constructed. The ACE dataset is based on the newswire section of the ACE-2005 corpus (see section 4.1.2). For simplicity, we only consider two entity classes, one majority class (MAJ) and a minority class (MIN). In ACE, we choose the "person (PER)" class as MAJ and "organization (ORG)" as MIN. All other classes are mapped to OUTSIDE (O).

The PBIO dataset is based on the annotations of the PENNBIOIE corpus for biomedical entity extraction (Kulick et al., 2004). Again, we used one MAJ class (a combination of the "gene-*" classes), one MIN class (a combination of the "variation-*" classes), and OUTSIDE.

The SYN dataset was constructed by mixing sentences from the original ACE and PENNBIOIE corpora. The "person" class constitutes the minority class; the very similar classes "malignancy" and "malignancy-type" were merged to

	PBio	Ace	Syn
sentences (all)	11,164	2,642	13,804
sentences (MAJ)	7,075	767	5,667
sentences (MIN)	2,156	974	974
MIN-MAJ ratio	1 : 3.3	1 : 1.3	1 : 5.8
tokens (all)	277,053	66,752	343,773
tokens (MAJ)	17,928	2,008	18,959
tokens (MIN)	4,079	1,822	2,008
MIN-MAJ ratio	1 : 4.4	1 : 1.1	1 : 9.4

Table 4.2: Characteristics of the datasets; "sentences (MAJ)", e.g. specifies the number of sentences containing mentions of the majority class.

form the majority class. All other class labels were set to OUTSIDE. Syn's construction was motivated by the following characteristics of the new dataset, which would make the appearance of the missed class effect very likely for insufficient exploration scenarios:

- Absence of inner-sentence entity class correlation to ensure that sentences contain either mentions of only a single entity class or no mentions at all.
- Marked entity class imbalance between the majority and minority classes.
- Dissimilar surface patterns of entity mentions of the two entity classes with the rationale that class similarity will be low.

Table 4.2 summarizes characteristics of the datasets. While Syn exhibits high imbalance (e.g. 1:9.4 on the token level), PBio and Ace are moderately skewed. In PBio, the number of sentences containing any entity mention is relatively high compared to Ace or Syn. For further details on the datasets, including statistics of inner-sentence entity co-occurrence, the reader is referred to (Tomanek et al., 2009). For our experiments, the corpora were randomly split into a pool for AL and a test set for performance evaluation.

Seed sets

Selection of an appropriate seed set for the start of an AL process is important to the success of AL. This is especially relevant in the case of imbalanced classes because a typically small random sample will possibly not contain any example of the rare class. We constructed different types of seed sets (whose naming intentionally reflects the use of the entity classes from above) to simulate different scenarios of ill-managed initial exploration. All seed sets have a size of 20 sentences. The RANDOM set was randomly sampled, the MAJ set is made of sentences containing at least one majority class entity, but no minority class entity. Accordingly, MIN is densely populated with minority entities. Finally, OUTSIDE contains only sentences without entity mentions.

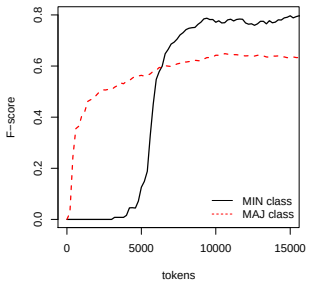
One could think of the OUTSIDE and MAJ seed sets of cases where a random seed set selection has unfortunately produced an especially bad seed set. MIN serves to demonstrate the opposite case. For each type of seed set, we sampled ten independent versions to calculate averages over several AL runs.

4.2.4 Results

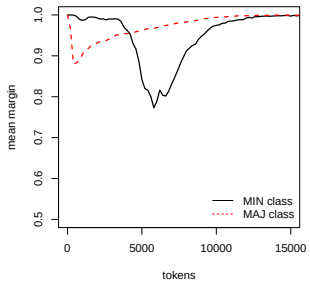
Results on SYN corpus

First, we show the results of the experiments on the Syn corpus in Figure 4.3. Token selection AL (T-AL) is shown in the upper row, sentence selection AL (S-AL) in the lower row. For all graphs, the x-axis shows the number of annotated tokens. For S-AL, the number of tokens on the x-axis is the total number of tokens in the sentences labeled so far.

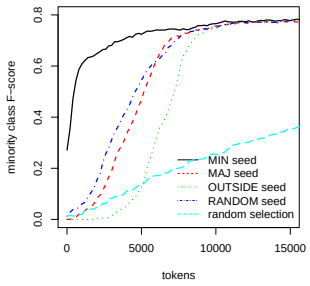
Figures 4.3(a) and 4.3(d) show the learning progress for minority and majority class, for a single AL run initialized with the OUTSIDE seed set. This problematic seed set causes the learner to focus on just the majority class. While learning progresses as expected for the majority class, for the minority class there is almost no learning progress on the minority class for the many iterations. Only at around 4,000 tokens for T-AL or 30,000 for S-AL, the learner



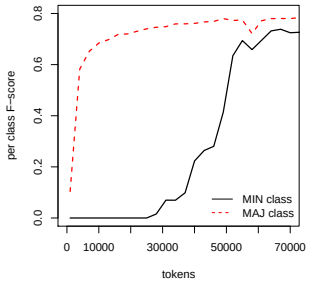
(a) T-AL learning curve, single run with OUTSIDE seed



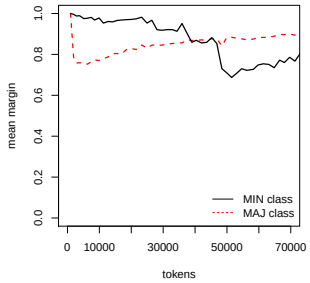
(b) T-AL mean margin curve, single run with OUTSIDE seed



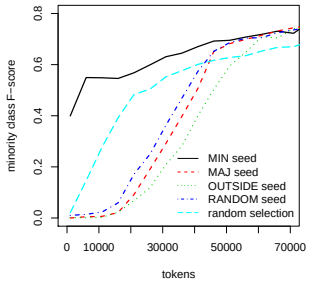
(c) T-AL learning curves, minority class, all seeds, 10 runs



(d) S-AL learning curve, single run with OUTSIDE seed



(e) S-AL mean margin curve, single run with OUTSIDE seed



(f) S-AL learning curves, minority class, all seeds, 10 runs

Figure 4.3: Results on SYN corpus for token selection (a,b,c) and sentence selection (d,e,f)

discovers the minority class and then quickly selects examples to learn the minority class as well.

Figure 4.3(b) and 4.3(e) show the mean confidence over the examples in the pool, broken down by class, during the learning progress. Examples of the minority class have high confidence ratings during the first iterations. Therefore, the minority class examples are misclassified with high confidence. Only after discovering some of the minority class examples does the margin drop sharply for this class, prompting the learner to sample more examples of this class and finally learn the concept.

Figures 4.3(c) and 4.3(f) compare learning performance with the different types of seed set. On S-AL all types of seed set except the MIN dataset are problematic for learning the minority class. Even the popular choice of a randomly drawn seed set is problematic when the minority class is as rare as in our SYN dataset. On T-AL, AL started with bad seed sets outperforming random selection, but this is because random selection is a weak baseline on the SYN dataset with token selection, due to the token distribution. There is still a marked difference between good and bad seed sets.

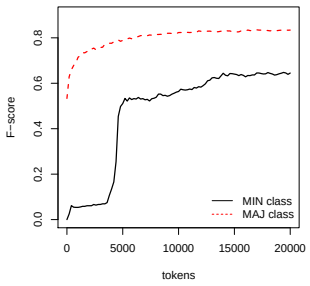
As we intended with the construction of the SYN dataset, both T-AL and S-AL exhibit weak learning progress on the minority class.

Results on PBIO corpus

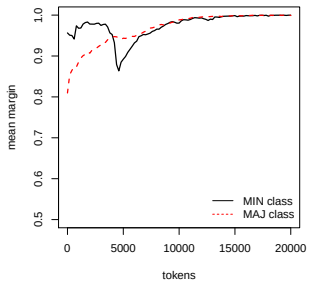
We show learning progress and confidence plots for experiments on the PBIO corpus in Figure 4.4. The AL process was initialized with the MAJ seed set, a seed set that was constructed to model the pathological case that the seed set contained no examples of the minority class. For T-AL we find almost no learning progress over many iterations (Figure 4.4(a)). Again confidence for minority class examples is too high (Figure 4.4(b)), so minority class examples get confidently misclassified and do not get selected for annotation and learning.

For S-AL, we find that the learning process is much more robust, as shown in Figure 4.4(e). Even with the pathological seed set, learning of minority class examples progresses steadily right from the beginning. Confidence values for these examples are low right from the start (Figure 4.4(e)), so minority class examples are frequently selected.

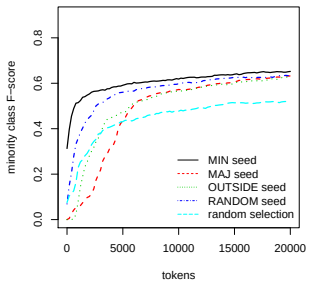
Comparing different kinds of seed sets confirms that S-AL is consistently more robust against seed set variation than T-AL (Figures 4.4(f) vs. 4.4(c)).



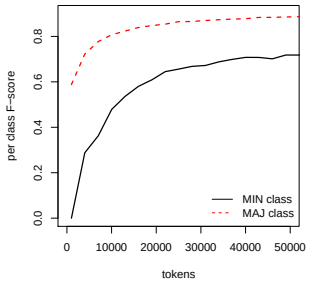
(a) T-AL learning curve, single run with MAJ seed



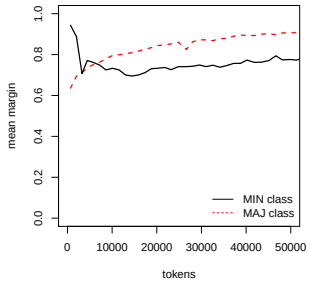
(b) T-AL mean margin curve, single run with MAJ seed



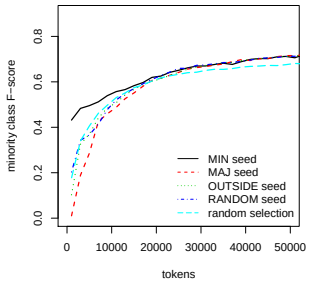
(c) T-AL learning curves, minority class, all seeds, 10 runs



(d) S-AL learning curve, single run with MAJ seed



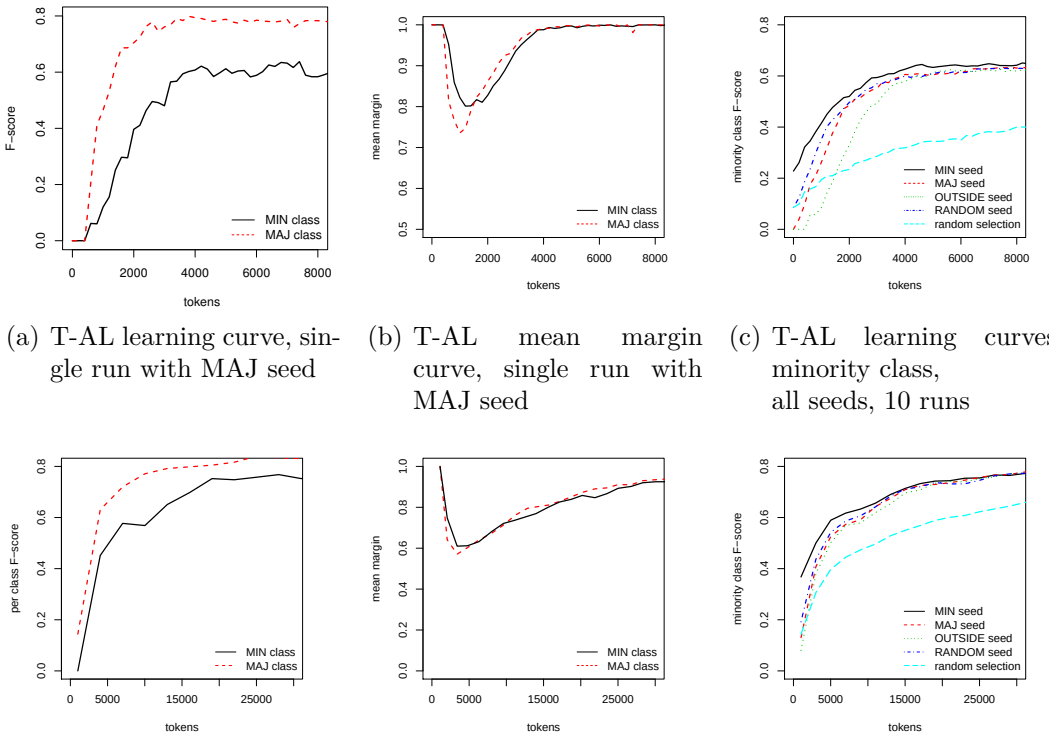
(e) S-AL mean margin curve, single run with MAJ seed



(f) S-AL learning curves, minority class, all seeds, 10 runs

Figure 4.4: Results on PBio corpus for token selection (a,b,c) and sentence selection (d,e,f)

Results on ACE corpus



(a) T-AL learning curve, single run with MAJ seed

(b) T-AL mean margin curve, single run with MAJ seed

(c) T-AL learning curves, minority class, all seeds, 10 runs

(d) S-AL learning curve, single run with MAJ seed

(e) S-AL mean margin curve, single run with MAJ seed

(f) S-AL learning curves, minority class, all seeds, 10 runs

Figure 4.5: Results on ACE corpus for token selection (a,b,c) and sentence selection (d,e,f)

We will only briefly discuss the results on the ACE datasets, shown in Figure 4.5. The missed class effect is less pronounced on ACE than it is on PBIO. We believe that the similarity of instances of the majority ("PERsons") and minority ("ORGanizations") classes is much higher on the ACE corpus than, for example, in PBIO. Still, the different seed sets visibly influence the T-AL learner. Especially the OUTSIDE seed has a marked negative effect. S-AL is again able to exploit the co-selection of co-ocurring entities and can avoid the

missed class effect: S-AL progress is much better for the minority class right from the beginning.

We conclude that sentence selection is an advantageous selection strategy for sequence classification problems in NLP. Through the co-selection of other entities that co-occur in a sentence, the problem of missing examples from minority classes can be avoided. In the following section, we will also show that selection of meaningful compound units is also beneficial for other NLP tasks—namely, coreference resolution.

Sentence selection expends some annotation effort (at least when measured in tokens) to this kind of implicit exploration of the sample space. When measured in tokens, convergence is thus slower than in pure token selection. However, sentence selection is also advantageous as the sentence is a natural unit that provides context for easy annotation of entities.

4.3 Example Selection for Coreference Resolution Using Committees and Neighborhood Co-Selection

While active learning has been successfully employed for a number of complex natural language processing tasks (see section 4.4.1), it has seldom been applied to coreference resolution. Previous work on this task has indicated that simple uncertainty sampling is not sufficient for successful AL for coreference resolution (Gasperin, 2009). In the following section, we propose an active learning strategy for coreference annotation that outperforms a random selection baseline by combining query by committee with a strategy that exploits co-selection in the neighborhood of coreferent mentions.

4.3.1 Negative Results with Standard Uncertainty Sampling

Attempts in previous work

Gasperin (2009) describes an experiment for coreference annotation using active learning with uncertainty sampling. She uses the popular mention-pair model (see section 3.2.1) for coreference resolution on a biomedical corpus.

A mention-pair model uses a pair classifier to classify pairs of mentions as coreferent or non-coreferent, and a clustering stage to group coreferent pairs that belong to the same entity into a coreference chain. Gasperin uses a variant of Naive Bayes for the pair classifier. This classifier is then used to select training examples in an active learning setup based on uncertainty sampling. Gasperin evaluates three versions of the entropy uncertainty measure: *local entropy* rates single candidate pairs, while two versions of *global entropy* compute an uncertainty measure that aggregates uncertainty over a number of candidates for a mention.

Gasperin's evaluation results are negative: example selection with the global entropy measures does not give improvements compared to random sampling

of the examples. Selection with local entropy even performs considerably worse than random selection.

Own attempts confirm negative results

We conducted experiments with standard uncertainty sampling to reproduce the findings of Gasperin (2009) with our own system and dataset. For these experiments, we use the SUCRE coreference system (Kobdani et al., 2011). SUCRE results were competitive in the SemEval-2010 shared task (Recasens et al., 2010).

Like Gasperin's system, SUCRE is based on the mention-pair model. It offers a choice of pair classifiers, of which we chose a decision tree classifier, which performed best in previous supervised evaluations (Kobdani and Schütze, 2010). The following clustering step is a variant of best-first clustering (Ng and Cardie, 2002).

We use the English part of the SemEval-2010 coreference resolution dataset, a subset of OntoNotes 2.0 (Hovy et al., 2006). Using the SUCRE system and SemEval-2010 corpus, we replicated Gasperin's result: uncertainty sampling is not better than random sampling for coreference annotation.

We believe that there are two factors for the unsatisfactory performance of uncertainty sampling for coreference annotation.

Unreliable example utility assessments In uncertainty sampling, example utility is derived from a confidence estimate. If these confidence estimates are too unstable, uncertainty sampling can fail. Instead of uncertainty sampling, we therefore adopted a query-by-committee approach (see section 4.3.2).

Class imbalance and missed clusters Coreference training data exhibits a very severe class imbalance: on our dataset of approx. 1.7 million links (candidate pairs), only about 25,000 are labeled as coreferent, which is only 1.5% of the links. Imbalanced datasets cause problems for classifier training and in active learning can trigger the missed cluster problem, where parts of the sample

space are not explored and clusters of the rare class are neglected. Selective sampling therefore needs to be combined with a strategy to balance the ratio of positive to negative examples. We therefore combine the query-by-committee approach with a class-balancing strategy that also exploits the co-selection effect. We describe this novel strategy in section 4.3.3.

4.3.2 Reliable Example Utility Assessment Using Query by Committee

We found that standard uncertainty sampling was unsuccessful when applied to coreference resolution. We think that one of the reasons is that the confidence measures could be too unstable for reliable example selection. This seems to be the case for decision trees, the preferred pair classifier in the SUCRE system, as shown by Dwyer and Holte (2007). For the Naive Bayes classifier, which Gasperin (2009) uses, uncertainty ratings could also be unreliable as it is known that despite good classification performance, Naive Bayes gives bad confidence assessments (Domingos and Pazzani, 1997).

Fortunately, there exists a family of example selection strategies that do not use potentially unreliable confidence estimates: the query-by-committee approach to active learning. Query by committee works by training a committee of several slightly different classifiers. Examples on which the classification decisions of the committee disagree most are then selected (see section 2.2.3). Therefore, this approach works on just the classification decisions and does not need confidence estimates. In fact, Dwyer and Holte (2007) show that for decision trees, QbC outperforms uncertainty sampling by a significant margin.

We therefore adopt the query-by-committee approach using SUCRE's decision tree pair classifier. Following Dwyer and Holte (2007), we use query by bagging (Abe and Mamitsuka, 1998) to construct the classifier committee. The committee consists of 10 instances of the pair classifier, each trained on a randomly chosen subset of the coreference links that have been manually labeled so far. For measuring the strength of disagreement, we use the vote entropy measure (see section 2.2.3), which is the most common utility mea-

sure for query-by-committee setups. (The "vote margin" measure, despite its superficial similarity to the "margin" uncertainty measure, has not been widely adopted.)

Using this query-by-committee approach, we found that selection is indeed more stable, but this is still not sufficient for successful AL applied to coreference annotation.

4.3.3 Neighborhood Co-Selection as an AL Strategy for Coreference Resolution.

When looking at coreference resolution training corpora, we find that classifiers in a mention-pair model are faced with severe class imbalances. There are many more disreferent than coreferent links. In fact, the coreferent links make up only 1.5% of all links in the pool.

This means we have a very similar situation to that of the rare named entity class scenario we described in section 4.2.1. Like in the NER scenario, the selection can fail to discover important examples, leading to slow learning progress—the missed cluster effect. We therefore adopt the same idea as in the NER scenario: selecting multiple instances to add some exploratory characteristic to the selection process.

Thus, we now need a sampling granularity that consists of more than a single link but is also both natural for human annotators and useful for the mention-pair system.

The neighborhood of a markable

We find that these criteria are fulfilled by a grouping of links surrounding a markable as proposed by Soon et al. (2001). We call one of these groups a *neighborhood*.

Each neighborhood is constructed as follows: given one markable x, the neighborhood of x is only the set consisting of the link between x and its

closest coreferent markable y to the left and all disreferent links in between

$$\{(z, x) \mid y \leq z < x\}$$

where $<$ means that the markable z occurs before x. This set is empty if x does not have a coreferent markable to the left.

We call the set of all such neighborhoods the *neighborhood pool* or *N-pool*, which we will then use for sampling. This means that we will not sample individual links, but entire neighborhoods of links. By definition, these neighborhoods contain both positive and negative links, thereby enabling the co-selection effect.

At the same time, this construction of neighborhoods leads to a more balanced ratio of coreferent to disreferent links (Soon et al., 2001), which is why we chose this definition of neighborhoods as our sampling granularity for co-selection.

Bootstrapping the neighborhoods

The definition of the neighborhoods assumes labeled data: since a neighborhood is defined using a positive coreferent link, coreference labels must be available. In AL, however, no labeled data (or just the very small part that has so far been annotated) is available.

Instead of using labeled data, we therefore bootstrap the neighborhoods using the committee of classifiers that we use for AL example selection. We query the committee of classifiers from the last AL iteration and treat a link as coreferent if and only if the majority of the classifiers classifies it as coreferent. We then construct the N-pool using the bootstrapped labels. The N-pool is then used as the pool for active learning example selection. In each iteration, the N-pool is recreated from the set of all links in order to improve the bootstrapped neighborhoods as the classifiers improve with training.

If this procedure yields no coreferent links in an iteration, we sample links left of randomly selected markables instead of N-pooling.

Sampling from the neighborhood pool

As described in section 4.3.2, we use a query-by-committee approach to AL, using query by bagging to construct the classifier committee: the committee consists of 10 instances of the link classifier of the CR system (a decision tree), each trained on a randomly chosen subset of the links that have been manually labeled so far.

To test the efficiacy of the neighborhood co-selection, we compare two selection granularities.

Neighborhood selection In *neighborhood selection*, entire neighborhoods are selected and labeled in each iteration. We define the utility of a neighborhood as the average of the vote entropies (see section 2.2.3) of its links.

Link selection In *link selection*, individual links with the highest utility are selected—in most cases these will be from different neighborhoods. Utility is again defined as vote entropy, this time of the single link.

Our hypothesis is that—compared to the selection of individual links—neighborhood selection yields a more balanced sample that covers both positive and negative links for a markable and thus makes use of the co-selection effect. At the same time, neighborhood selection retains the benefits of AL sampling: difficult (or highly informative) links are selected.

N-pool bootstrapping as well as neighborhood and link selection were implemented on top of the AL framework of Tomanek et al. (2007). The base coreference resolution system is, like in the preliminary experiments, SUCRE (Kobdani et al., 2011).

4.3.4 Experiments

Setup

We used the English part of the SemEval-2010 CR task dataset, a subset of OntoNotes 2.0 (Hovy et al., 2006). This dataset comes with predefined training and test splits. Training and test set sizes are about 96,000 and 24,000 words. We use the training split as the basis for generating the pool, and the test split for evaluation. Since we focus on the coreference resolution subtask, we use the true mention boundaries for the markables.

The base pool for example selection is created by pairing every markable with every preceding markable within a window of 100 markables. As mentioned above, this yields a pool of approx. 1.7 million links, of which only 1.5% are labeled as coreferent. This pool serves as the input for the N-pooling strategies.

We run two baseline experiments for comparison:

1. Random selection on the entire pool, without any class balancing.
2. Random selection from a gold-label-based N-pool, resembling the Soon et al. (2001) method.

We chose to use gold neighborhood information for the baseline to remove the influence of badly predicted neighborhoods and focus on the performance of random sampling. Hence, this is a very strong random baseline. The performance with bootstrapped neighborhoods would likely be lower.

These baselines are compared with the two active learning setups described above:

3. Link sampling on a bootstrapped N-pool.
4. Neighborhood sampling on a bootstrapped N-pool.

We perform 10 runs of each experiment, starting from 10 different seed sets. These seed sets contained 200 links, drawn randomly from the entire pool, for random sampling, and 20 neighborhoods for neighborhood selection,

with a comparable number of links. We verified that each seed set contained instances of both classes. Evaluation scores were averaged over the 10 runs for the following results.

Results

			MUC	B^3	CEAF	mean
20,000 links						
(1)	random	entire pool	49.68	86.07	82.34	72.70
(2)		(gold) N-pooling	61.60	85.00	82.85	76.48
(3)	AL	link selection	55.65	86.91†	83.67†	75.41
(4)		neighborhood sel.	63.07†	86.94†	84.42†	78.14†
50,000 links						
(1)	random	entire pool	48.81	86.00	82.24	72.34
(2)		(gold) N-pooling	62.60	85.99	83.44	77.33
(3)	AL	link selection	55.84	86.94†	83.70	75.49
(4)		neighborhood sel.	63.81†	87.11†	84.33†	78.42†

Table 4.3: Performance of different methods. All measures are F_1 measures. † indicates statistical significance at $p < .05$ compared to baseline 2 using the sign test ($N = 10$, $k \geq 9$ successes).

Similar to AL evaluation for NER (see section 4.1.3), we compare performance of the AL systems and the baselines according to the number of links used for training.

We determine the performance of coreference according to the number of links used for training. The results of the experiments are shown in Table 4.3 and learning curves are shown in Figure 4.6. We show results for four coreference measures: MUC (Figure 4.6(a)), B^3 (Figure 4.6(b)), entity-based CEAF (henceforth: CEAF, Figure 4.6(c)), and the arithmetic mean of MUC, B^3 and CEAF (as suggested by the CoNLL-2011 shared evaluation, Figure 4.6(d)).

For all four measures, active learning has converged and the learning curves exhibit a plateau at about 20,000 links. At this point, neighborhood selection

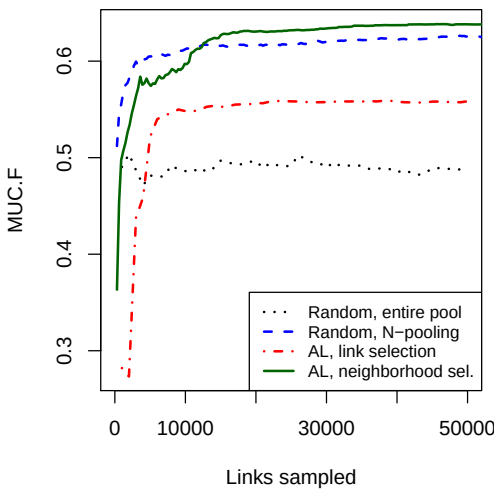
AL (line 4 in Table 4.3) outperforms random sampling from the N-pool (line 2) for all coreference measures, with gains from 1.47 points for MUC to 1.94 points for B^3.

The random selection baselines show much slower convergence. At 20,000 links, the N-pooling random baseline (line 2) has not yet reached maximum performance. Even given more training data, at 50,000 links, neighborhood selection AL still outperforms the baselines.

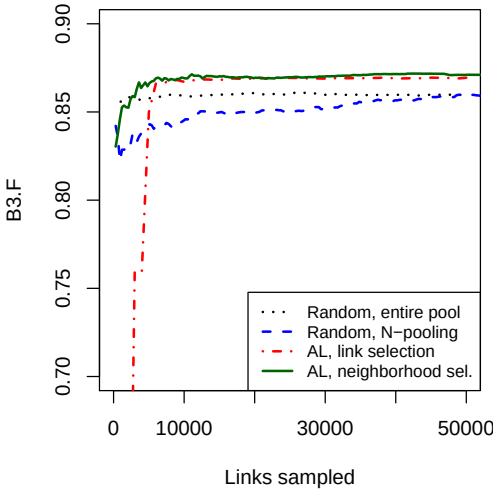
For link selection AL (line 3) the results are mixed, depending on the evaluation measures. Link selection AL outperforms the baselines for B^3 and CEAF, but performs markedly worse than the N-pooling random baseline (line 2) for MUC (due to low recall for MUC) and mean F_1. Link selection yields a coreference system that proposes a lot of singleton entities that are not coreferent with any other entity. The MUC scoring scheme does not give credit to singletons at all, thus the lower recall.

Neighborhood selection AL initially has low MUC scores but starts to outperform the baseline at 15,000 links (Figure 4.6(a)). For B^3 and CEAF, neighborhood selection AL outperforms the baselines much earlier, at a few thousand links (Figures 4.6(b) and 4.6(c)). Thus the performance of neighborhood selection AL is more robust across all evaluation metrics.

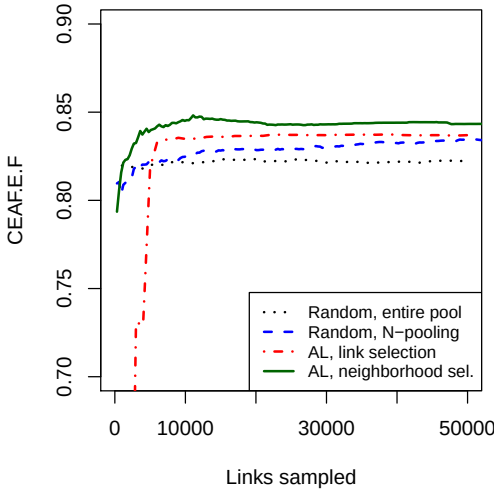
Neighborhood selection AL also performs at least as well as (for B^3) or better than link selection AL (for MUC and CEAF). Learning curves of neighborhood selection AL are consistently above the link selection curves. We therefore consider neighborhood selection AL to be the preferred AL setup for coreference annotation.



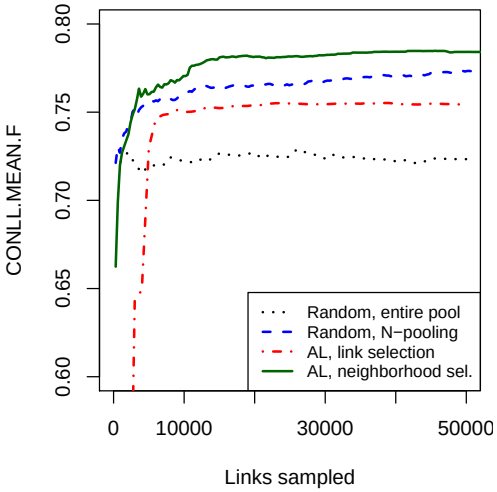
(a) Learning curve for MUC-F



(b) Learning curve for B^3-F



(c) Learning curve for CEAF-F



(d) Learning curve for the mean of the CR measures

Figure 4.6: Evaluation of random sampling and active learning strategies for coreference resolution

4.4 Related Work

4.4.1 Active Learning for Named Entity Recognition

Active learning has been applied to NER using predominantly the query-by-committee and the uncertainty sampling frameworks. Uncertainty sampling has been employed by, for example, Cheng et al. (2008) using structured SVMs as the base classifier, or by Wanvarie et al. (2010) using CRFs. Query by committee has been used by Becker et al. (2005) using maximum entropy Markov models (MEMM) as well as by Tomanek et al. (2007) and Olsson (2009) using CRFs. Committees are usually constructed using query by bagging, except for Becker et al. (2005), who use multi-view committees, i.e. committees whose member classifiers have different feature sets. Settles and Craven (2008) compared some of the more elaborate approaches like expected model change (see section 2.2.4) to classical and density-weighted uncertainty sampling as well as query by committee but did not find improvements using these more complex approaches. All of these approaches do not perform exploration of the sample space and thus may be affected by issues like the missed cluster effect.

The sampling granularity is usually chosen depending on the model that is being trained, i.e. sequences for sequence models (e.g. Settles and Craven, 2008) or tokens (e.g. Shen et al., 2004) for token-based models. Sassano and Kurohashi (2010), however, recently proposed to select sub-sentence constituents for AL for dependency parsing. Robustness is usually not a consideration when choosing sampling granularity. An exception is Olsson (2009), who proposes to select entire documents for annotation. He argues that this helps to create a versatile corpus as opposed to just a training set. Using such a large sampling granularity for AL may bring large co-selection effects, but it is not focused enough to outperform random sampling.

Active learning has also been applied to other sequence classification tasks. Examples include part-of-speech tagging (Engelson and Dagan, 1996; Ringger et al., 2007), NP chunking (Ngai and Yarowski, 2000) or labeling of prosodic events (Fernandez and Ramabhadran, 2011). Beyond sequence classification,

active learning has also been applied to other complex natural language processing tasks, such as parsing (Tang et al., 2002; Baldridge and Osborne, 2008) and statistical machine translation (Haffari et al., 2009; Bloodgood and Callison-Burch, 2010).

4.4.2 Coreference Resolution

The only direct attempt at active learning for coreference annotation that is known to us is due to Gasperin (2009). Gasperin uses uncertainty sampling to select training examples for a mention-pair coreference resolver. We discuss her work in section 4.3.1. Gasperin reported negative results in her evaluation. Example selection using uncertainty sampling did not outperform random selection. In the above section, we proposed a strategy that solves the weaknesses of standard uncertainty sampling and outperforms random sampling.

Very recently, Miller et al. (2012) applied active learning to the pair classification stage of a mention-pair coreference resolver. They propose a hybrid selection of single instances and entire documents that appears to be appealing from a human-computer-interaction point of view. However, they only evaluate on pair classification performance, not on standard coreference measures. We have found in preliminary experiments that because there is also the clustering step, good AL performance on pair classification alone is not sufficient for good performance on full coreference resolution.

Three other publications are somewhat related to active learning for coreference annotation and to the committee-based approach that we chose. Ng and Cardie (2003) propose a co-training strategy to coreference resolution. Co-training is a semi-supervised strategy that unlike active learning does not use interactive annotation of training data.

Ren and Zhu (2008) use active learning in the context of coreference resolution, but they employ active learning as a strategy for feature induction of the coreference classifiers, not for the annotation of coreference resolution training data.

Vemulapalli et al. (2009) use ensembles of classifiers for high-accuracy predictions for coreference resolution. However, they employ the ensemble for the final predictions, while we use it as a component for example selection in active learning.

4.5 Summary

We have investigated active learning strategies for the selection of training examples for two NLP tasks: named entity recognition and coreference resolution. We show that uncertainty sampling is quite successful for NER. Performance levels of baseline selection by random sampling is reached after annotating only 7% of the data. Performance even surpasses the baseline by about 2.5 percentage points F_1 score at 10% of the data. The *Margin* confidence measure performed slightly better than the other confidence measures tested.

However, active learning has the risk of the *missed cluster* or *missed class* effect: slow learning on (parts of) a target concept if the sampling strategy fails to explore important regions of the sample space, often triggered by bad selection of the initial dataset. We show that for NER, this behavior can be avoided by choosing an appropriate size of the examples. When choosing sentences as the unit of annotation, the co-selection of co-occurring entities within a sentence provides some exploration of the sample space at low overheads. We can therefore avoid the missed class effect.

We also make use of co-selection of instances for different classes in our novel AL strategy for coreference resolution. For this task, standard uncertainty sampling does not improve over random selection. We propose a novel AL strategy, *neighborhood sampling*. The strategy balances the sampling pool by bootstrapping neighborhoods, groups of mentions that are enclosed by a coreferent mention pair. These neighborhoods then get selected using a query-by-committee AL approach. We demonstrate that AL with the neighborhood sampling strategy outperforms random sampling of examples for coreference resolution.

5 Controlling the Active Learning Process

Despite the demonstrated reduction in annotation effort, active learning is still an interactive process that can require a considerable amount of time and expense, especially when complex tasks or large datasets are involved. Therefore, we seek ways to monitor the progress of annotation and training to avoid wasted effort. One particularly interesting aspect is *early stopping*: stopping the selection and annotation loop before the whole pool is labeled. In fact, when the pool is completely labeled, performance is the same as with other sampling strategies, so using only a fraction of the pool is essential to the success of any active learning method.

It is often the case that the amount of unlabeled data that is available far exceeds the available budget for annotation, and we can only annotate a fraction of the unlabeled data, regardless of the selection strategy. Nevertheless, it is still advantageous to monitor the learning progress for two reasons:

Performance estimation Before deploying a statistical classifier in practice, one has to evaluate if the classifier meets the required classification performance. Usually, this is done by evaluation on held-out labeled data, but labeled data is scarce in the scenarios where active learning is of interest.

Learning convergence As the number of annotated examples increases, performance does not grow steadily, but slows and at some point may converge to a performance level that will not be exceeded. At this point, annotation should

stop to avoid wasting effort on annotation that is not likely to yield further improvement.

We investigate methods to monitor the active learning process with regard to both aspects. Our focus is on stopping the annotation process to ensure that the annotation budget is not wasted. Performance estimation enables us to stop annotation when a minimum performance requirement is met; checking for convergence enables us to stop when the maximum performance that is possible using this classifier setup and dataset is reached.

5.1 Stopping by Minimum Absolute Performance

As mentioned above, in practical applications classifiers can only be reliably deployed when they attain a predefined *minimum absolute performance* level. We would like to determine if this level has been reached and then stop the annotation process. This is commonly done by evaluating the classifier's performance on a separate held-out set of labeled test data. However, in order for an evaluation to provide a reliable performance estimate, the test set needs to be of sufficient size. In situations where active learning is used, labeled data is not easily available but needs to be annotated first. Creating a labeled test set therefore requires substantial additional annotation effort, which is what we want to avoid by using AL in the first place.

Therefore, we will try to estimate the classifier's performance on unlabeled data.

5.1.1 Performance Estimation

Lewis (1995) proposes to estimate a classifier's performance based on its current estimates of the class probabilities. We give a brief recap of Lewis's method. The F_1 measure (henceforth: F) is defined as the harmonic mean of precision (P) and recall (R). We can write F as a function of true positives

(TP), false positives (TP), and false negatives (FN).

$$F = \frac{2 \cdot P \cdot R}{P + R} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

Table 5.1 shows the calculation of true positives (TP), false positives (TP), false negatives (FN), and true negatives (TN) over n examples. s_i is a variable that indicates the classifier's decision for the example i, while Z_i indicates the true label of the example.

	Reference is 1	Reference is 0
Response is 1	$\mathrm{TP} = \sum_{i=1}^{n} s_i Z_i$	$\mathrm{FP} = \sum_{i=1}^{n} s_i(1 - Z_i)$
Response is 0	$\mathrm{FN} = \sum_{i=1}^{n}(1 - s_i)Z_i$	$\mathrm{TN} = \sum_{1=n}^{n}(1 - s_i)(1 - Z_i)$

Table 5.1: Contingency table of classification decisions and true labels, modeled as random variables

If the true labels are not known, they can be modeled as Bernoulli random variables Z_i. Lewis then proposes to estimate performance by taking the expected values. This can simply be done by replacing the random variables with the classifier's estimate of the probability of the positive class p_i (see Table 5.2).

	Reference is 1	Reference is 0
Response is 1	$\widehat{\mathrm{TP}} = \sum_{i=1}^{n} s_i p_i$	$\widehat{\mathrm{FP}} = \sum_{i=1}^{n} s_i(1 - p_i)$
Response is 0	$\widehat{\mathrm{FN}} = \sum_{i=1}^{n}(1 - s_i)p_i$	$\widehat{\mathrm{TN}} = \sum_{1=n}^{n}(1 - s_i)(1 - p_i)$

Table 5.2: Contingency table with expectations

The multiclass classification case

We now extend this performance estimation for $\widehat{\mathrm{TP}}$, $\widehat{\mathrm{FP}}$, and $\widehat{\mathrm{FN}}$ to one-vs.-all multiclass classification for an NER task.

Scoring for an NER task has one peculiarity, the O (outside or "not an NE") class, which indicates that the token is not part of a named entity. Since the majority of tokens are not part of a named entity (i.e. they have

an O tag), evaluation scores would be meaninglessly high if systems would get credit for correctly predicted O tokens. Instead, systems get only "true positive" credit for named entity (non-O) tokens. O tokens are only credited as "true negatives". One can think of a named entity recognition as retrieving (and classifying) entities from a sea of otherwise irrelevant O tokens.

Response	Reference		
	O	NE1	NE2
O	TN	FN	FN
NE1	FP	TP	FN, FP
NE2	FP	FN, FP	TP

Table 5.3: Confusion matrix for a two-class NE problem

Consider the confusion matrix in Table 5.3 for a simplified NE classification problem with two NE classes and the class O. Like standard NER evaluation schemes, for example Tjong Kim Sang and De Meulder (2003), we consider only those decisions to be TPs where
(i) the reference class matches the selected class
(ii) this class is not O.

We therefore need to treat the NE classes differently from the O class in the estimation. We do this by replacing the decision variable with a combination of flags indicating the NE class ($e_{i,j}$) and the chosen class ($d_{i,j}$), and loop over all possible classes.

$$\widehat{\mathrm{TP}} = \sum_{i}^{n} \sum_{j}^{C} \hat{p}(c_j|x_i) e_{i,j} d_{i,j} \tag{5.1}$$

$$\widehat{\mathrm{FP}} = \sum_{i}^{n} \sum_{j}^{C} (1 - \hat{p}(c_j|x_i)) e_{i,j} d_{i,j} \tag{5.2}$$

$$\widehat{\mathrm{FN}} = \sum_{i}^{n} \sum_{j}^{C} \hat{p}(c_j|x_i) e_{i,j} (1 - d_{i,j}) \tag{5.3}$$

For each example, we loop over the C different classes. The flag $e_{i,j}$ indicates "is an NE class":

$$e_{i,j} = \begin{cases} 1 & \text{if the class } c_j \text{ is an NE class} \\ 0 & \text{if the class } c_j \text{ is the O class (not NE)} \end{cases}$$

The flag $d_{i,j}$ indicates "is winning class", i.e. the class with the highest probability assigned by the classifier:

$$d_{i,j} = \begin{cases} 1 & \text{if } j = \underset{j}{\operatorname{argmax}}\, \hat{p}(c_j|x_i) \\ 0 & \text{else} \end{cases}$$

When estimating TP, we assume that the probability of a match equals the probability of the selected class (which is $\hat{p}(c_j|x_i) \cdot d_{i,j}$). The entity flag $e_{i,j}$ ensures that this only gets counted if the selected class is in fact an NE class, so O predictions do not get credited toward TP. The probability of making an FP error is just the remaining probability mass that is distributed over entity classes.

For FN, there are two cases to consider. The first case is that the classifier predicts O when the correct class is in fact an NE class. In this case, $(1 - d_{i,j})$ will be 1 for all non-chosen entity classes, and the probability of making an FN error will therefore sum up to the remaining probability mass.

The second case is that the classifier predicts an NE class (say, NE1), but the correct class is some other NE class (e.g. NE2). In that case we also score an FN (in addition to an FP). We can again calculate the estimated probability of FN by summing up the class probabilities of the remaining non-chosen classes. The entity flag $e_{i,j}$ serves to skip the probability mass that got assigned to the O class, since predicting NE1 with a reference of O would not count toward an FN.

Note that since the estimation method works on completely unlabeled data, the true labels on the evaluation data do not influence the estimated scores.

5.1.2 Evaluation

To evaluate the performance estimation method, we ran an AL experiment on the ACE NER dataset. We used an active learning setup with token selection and the BBR logistic regression classifier as described in section 4.1.2. The iteration was stopped after selecting 10,000 examples with the *Margin* measure. The following performance scores and estimates are derived from a classifier that was trained on the snapshot of the labeled set after we stopped the iteration. Since the estimation method is intended for unlabeled data, we do not use the labels on the test set for estimation purposes. For comparison to "true" performance, we did a standard held-out evaluation using the labels on the test set.

	Ref.	Lewis	Δ Lewis	LOO	Δ LOO
F	79	92	+13	85	+6
P	81	92	+11	86	+5
R	77	92	+15	84	+7

Table 5.4: Performance estimation. LOO and Lewis overestimate true F by 6% and 13%, respectively.

Table 5.4 compares the true held-out performance on the test set, reported as "Ref.", with two performance estimates. The column "Lewis" shows the score of our extension of Lewis's performance estimate to multiclass classification as proposed above in section 5.1.1. The Δ columns report the difference of the named method to the "Ref." column. We also tested leave-one-out (LOO) estimation on the selected training set.

Leave-one-out estimation is a performance estimation method that relies on labeled data. Hence, we do not use it on the labels of the test set (we assume that we will not have a labeled test set in practice), but instead apply it to the labeled training set that has been selected and annotated so far. However, in active learning setups, this set is likely to be biased by the selective sampling procedure. Also, leave-one-out estimation is computationally expensive, especially on larger datasets.

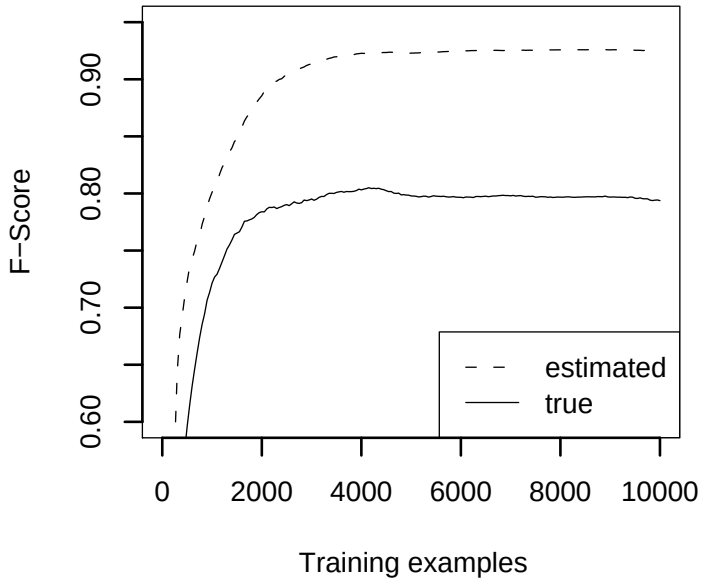


Figure 5.1: Estimated and true performance

We find that both estimation methods, Lewis and LOO estimation, overestimate precision and recall by a large margin.

In addition to the table, Figure 5.1 shows true and Lewis-estimated performance as a function of training examples used. We note that the peak in true performance at about 4,000 training examples (see also section 4.1.3) does not occur when evaluating performance using the Lewis method. Instead, the estimate of F just grows asymptotically. This means that we cannot use a peak of estimated F as a criterion for stopping.

When setting an absolute threshold of $F = 80\%$ for stopping, active learning stops at about 1,000 iterations, yielding a true performance of only $F = 73\%$ (selection by *Margin*, 20 trials). This indicates that we cannot directly use Lewis estimates for stopping.

5.1.3 Error Analysis

The reason for the overestimation is that the logistic regression classifier is too confident in its own decisions. For positive decisions, the class probability very often is close to 1; for negative decisions, it is close to 0. As a result, the estimator gives very little score for FN (Equation 5.3) or FP (Equation 5.2) in most instances, which leads to the high overestimation of performance.

To verify this, we grouped the empirical probability of a selected class being the correct class in bins according to the estimated probability of the logistic classifier. Table 5.5 shows this empirical probability given a class and its estimate.

		Negative decisions			Positive decisions			
		0-.2	.2-.4	.4-.6	.2-.4	.4-.6	.6-.8	.8-1
O	emp	0.0643	0.269	0.25	0.0	0.25	0.233	0.991
	est	0.00825	0.295	0.438	0.394	0.537	0.714	0.999
	cnt	607	26	12	1	16	30	5609
	err	34	-0.67	-2.25	0.394(tn)	4.6 (tn)	14.4 (tn)	45.4 (tn)
GPE	emp	0.00384	0.391	0.5	0.0	0.333	0.571	**0.875**
	est	0.000812	0.296	0.435	0.357	0.535	0.687	**0.989**
	cnt	5985	23	6	1	9	21	**256**
	err	18.1 (fn)	2.19 (fn)	0.388 (fn)	0.357 (fp)	1.82 (fp)	2.42 (fp)	**29.2 (fp)**
ORG	emp	0.00853	0.393	0.667		0.5	0.615	0.828
	est	0.000847	0.283	0.441		0.545	0.71	0.968
	cnt	6093	28	12		14	26	128
	err	46.8 (fn)	3.06 (fn)	2.7 (fn)		0.631 (fp)	2.46 (fp)	17.9 (fp)
PER	emp	0.0041	0.455	0.5		0.273	0.5	0.93
	est	0.000748	0.283	0.48		0.563	0.718	0.98
	cnt	6102	22	6		11	18	142
	err	20.4 (fn)	3.78 (fn)	0.121 (fn)		3.2 (fp)	3.93 (fp)	7.19 (fp)

Table 5.5: Empirical probabilities and contribution to estimation errors. (We omit small classes and empty columns.) The bold cell serves as an example that is detailed in the text.

The table is split into two halves, such that the empirical probabilities for positive decisions (the class got chosen as the best class) and negative decisions are shown separately. The top value in each cell ("emp") shows the empirical probability, i.e. the fraction of the examples that in fact have the predicted label. This is compared to the average of the estimated probability of these examples, which is the value below ("est"). The product of the difference of these two probabilities and the number of instances that were counted into

this bin ("cnt") gives an estimate of how much the probability estimates in the bin contribute to the error (absolute value) of the performance estimation.

Example (cell in bold):
256 tokens were labeled as GPE[1] by the classifier with estimated probabilities between 0.8 and 1.0. The average probability estimate was 0.989. In reality, only 224 of these tokens (87.5%) were GPEs. The contribution of this cell to the true FP count is $(1 - 0.875) \cdot 256 = 32$, while the estimation of FP is only $(1 - 0.989) \cdot 256 \approx 2.8$. The contribution of this cell to the estimation error of the FP count is thus $(0.989 - 0.875) \cdot 256 \approx 29.2$.

The table shows that class probabilities are in fact estimated too optimistically. For many of the entries in the positives table, the estimated probabilities are greater than the empirical probabilities. In the negatives table, the estimated probabilities are smaller. In both cases, the estimates are closer to the respective extreme values 1 or 0, which means they are overconfident. Note that for positive decisions, the estimation error of the values in a single bin contributes to the overall estimation error in two ways: overestimating TPs and underestimating FPs. For example, the estimation error for the example cell in bold is 29.2, contributing −29.2 for FP (underestimation) and +29.2 for TP (overestimation). Also note that due to the high number of non-NE tokens in the text, there are many negative decisions for each entity-class classifier; thus, small differences in the probabilities make large contributions to error.

To distinguish between errors introduced by bias of the selective sampling method as opposed to bias introduced by the classifier or the whole dataset we ran a separate experiment in which we trained a classifier on the entire labeled pool. The Lewis estimator overestimated F by +12% in this case. This indicates that the estimation error does not primarily come from the biased selection of training examples resulting from the selective sampling method, but from bias inherent in either the whole pool of training data or the base classifier.

[1]Recall that GPE means a location with a government, i.e. a country or a city.

5.1.4 Corrected Estimates

Since over-optimistic estimates for precision and recall stem from the classifier's over-optimistic probability estimates, we try to correct the estimates. We do this by replacing the predicted class probabilities by the appropriate value in an empirical probability table like the one shown in Table 5.5.

This adjustment table counts the fraction of correct predictions depending on the estimated probability and the predicted class. For each class, there is a set of five bins of ranges of estimated probabilities, in which examples will be counted. The bins are evenly sized and cover the ranges $0 \ldots 0.2$, $0.2 \ldots 0.4$ etc. to $0.8 \ldots 1.0$. For each example in a dataset, we get the prediction and associated probability estimate from the classifier and check the prediction against the example's label. We then record if the decision was correct or incorrect in the appropriate bin. (Example: an example that gets correctly classified with "GPE" and a probability estimate of 0.83 gets recorded as correct in the GPE, $0.8 \ldots 1.0$ bin.)

We assume that in practice we would not have labels for the test set. Therefore, we use leave-one-out estimation to bootstrap the adjustment table from the selected training data. This means a classifier gets trained on every example from the training set except one. This classifier is used to make a prediction on the remaining example. The remaining example also has a label, which we can then compare to the prediction. The result is then recorded in the appropriate bin. This procedure is repeated for all examples in the training set.

After filling the table, we assume that the ratio of correct decisions to all decisions in a bin is the empirical class probability. We can now make an adjusted performance estimation: we use the same estimation formulas as before (formulas 5.1 to 5.3), but whenever we need a class probability $\hat{p}(c_j|x_i)$, we replace the classifier's estimate by the empirical probability value from the appropriate bin.

Results for corrected estimates

The adjusted estimation shows a marked increase in the estimates for FP and FN, leading to a quite accurate estimate for precision (+5 absolute error); however, the now pessimistic estimate for recall (−16) leads to underestimation of F-Score overall (−8) (see Table 5.6).

	True	Lewis	adj. Lewis	Δ adj. Lewis
F	78	91	70	-8
P	81	93	86	+5
R	76	89	60	-16
TP	520	596	555	+35
FP	125	48	90	-35
FN	163	70	379	+216

Table 5.6: Lewis estimation with adjusted probabilities

As we see, the adjustment overshoots for recall, indicating that the new estimated probabilities are still off. There could be several reasons for this. The first reason might be that the bin width is quite coarse, as there are only five bins for the entire probability interval, each bin covering a range of 0.2. However, using finer bin widths can lead to data sparsity problems. A possible solution to this problem is a dynamic bin width algorithm as used by Li and Sethi (2006). Another reason might be the estimation errors within individual bins that compound to a quite large overall error especially in the negative case. Finally, the distribution of the training set can differ from the reference set such that the estimates become unreliable. In fact, since the empirical probabilities for the adjustment table are estimated with leave-one-out on a training set created by selective sampling, the training set will necessarily be biased.

While the precision estimate improved, the difficulty in recall estimation makes the overall F estimate not accurate enough for a reliable decision if the classifier performance is acceptable for practical use. (Recently, some work-arounds for this issue have been proposed, but these need a small amount of

labeled test data, see section 5.3.) In the next section, we will therefore focus on detecting when the maximum possible performance is reached.

5.2 Stopping by Maximum Possible Performance

5.2.1 Confidence-Based Stopping

We have found that performance estimation is not yet reliable enough to stop when a desired performance level is reached. However, we found that there is a maximum performance that can be reached on any given sampling pool (see section 4.1.3). Therefore the annotation process should stop at this point regardless of whether a target performance level has been reached or not. Even though the target performance might not have been reached, further annotation will not improve performance in that case. In this case, one should consider gathering fresh data to refill the pool or improve the model.

We therefore seek a stopping criterion that finds the *maximum possible performance* when the classifier is iteratively trained on a given sampling pool. Again, in practice we do not have a labeled test set to evaluate against, so we have to try to find the stopping point from either the remaining pool or a separate set of *unlabeled* data we call the *reference set*.

Vlachos (2008) proposes calculating *confidence* of the classifier by using the *average uncertainty on the unlabeled reference set*. For multiclass problems, he uses SVM classifiers with the SVM margin size as the uncertainty measure. Using this measure, Vlachos reports finding a peak pattern in this confidence measure, albeit distorted by fluctuations. He reports that his confidence scores rise, reach a peak, and then fall again, and that this coincides with reaching maximal performance in his experiments. Vlachos then suggests using this *peak confidence* as the stopping criterion.

We tested this approach with the multiclass logistic regression setup. However, in our experiments, we could not find the peak pattern described by Vlachos when calculating the confidence using the three measures introduced above: *1-Entropy*, *Margin*, and *MinMax*.

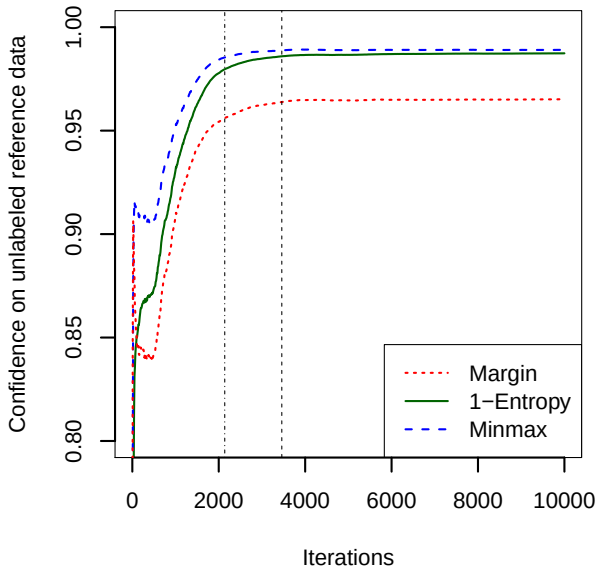


Figure 5.2: Confidence on unlabeled reference set (Selection: *1-Entropy*). The vertical lines indicate when baseline and peak performance are reached. There is no peak pattern in the curves, so reaching peak confidence cannot be used as a stopping criterion.

In Figure 5.2, we show the three confidence measures, averaged over 20 trials as described in section 4.1.2. There are some fluctuations in the first 100 iterations, but these are due to instability of the AL process in the early start-up phase. Also note that the fluctuations are more marked for the confidence measures not used as a selection criterion because the sampling process does not directly optimize for those. After 500 iterations the confidence curves stabilize and at about 4,000 iterations approach asymptotes close to 1: 0.965 for *Margin*, 0.989 for *MinMax*, and 0.987 for *1-Entropy*.

However, there is no rise-then-fall peak pattern that could be detected and used for stopping. Thus, Vlachos's proposed criterion of peak confidence based on average confidence on the reference set does not seem to be applicable for controlling AL with multiclass logistic regression.

5.2.2 Gradient-Based Stopping

Since we cannot find the peak pattern for stopping, we propose to stop when a base measurement characterizing the progress of active learning has converged. We identify the point of convergence by computing gradients on two different series of values: the performance estimate (*performance convergence*) and the uncertainty measure used for selection (*uncertainty convergence*).

Performance convergence

Looking back at Figure 5.1, we see that the rise of the performance estimate slows to an almost horizontal slope at about the time when the true performance reaches its peak. We therefore propose the following new stopping criterion: estimate the gradient of the curve of the Lewis performance estimate and stop when it approaches 0. Since we do not need an accurate estimation of absolute performance here and only a useful gradient is sufficient, we can use the unadjusted Lewis estimate for this method. We call this stopping criterion performance convergence.

Uncertainty convergence

In a similar way, we can use the gradient of the confidence measures of the instance last selected for labeling and training. The instance selected is always the least confident example—in other words, the example with maximum uncertainty—and thus the most informative for training. When this confidence measure comes close to the extreme value of 1 (since confidence is the opposite of uncertainty, 1 means *no* uncertainty for this instance), we decide that there are no informative examples left in the pool and we stop the AL process. The gradient of the uncertainty measure approaches 0 at this point (see Figure 5.3), so we can again use a gradient criterion for implementing this idea. We call this stopping criterion uncertainty convergence.

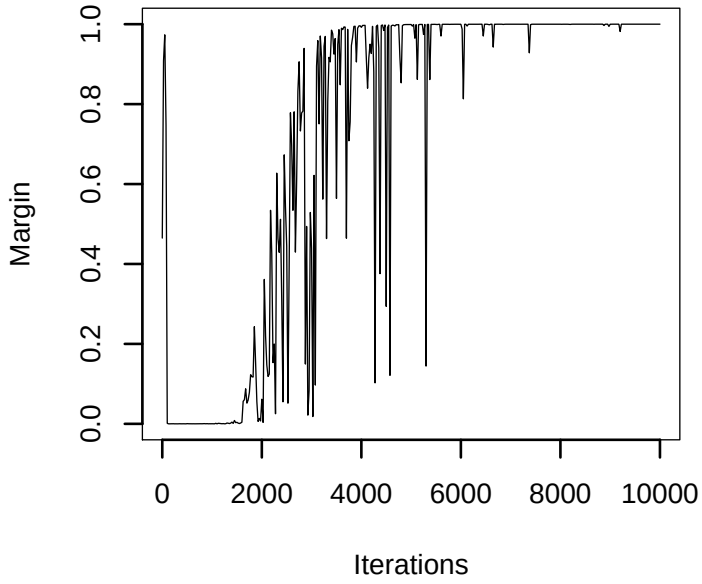


Figure 5.3: *Margin* confidence of selected instance (single run). The graph demonstrates that without smoothing, this criterion is too noisy.

Computation of the gradient

In Figure 5.3, which shows a graph of the *Margin* confidence of the selected instance, we can also see that the confidence value is quite noisy. The confidence value drops sharply when some examples are encountered but quickly returns to the previous level after a few iterations.

The performance estimation measure is slightly noisy as well, so we need a robust way of filtering the noise and computing the gradient. We achieve this with a moving median approach. At each step, we compute the median of $w_2 = \{a_{n-k}, \ldots, a_n\}$ (the last k values) and of $w_1 = \{a_{n-k-1}, \ldots, a_{n-1}\}$ (the previous last k values). Each value a_i is the performance estimate at iteration i for the performance gradient computation or the uncertainty of the instance selected in iteration i for the uncertainty gradient computation. The number of the current iteration is n.

We then estimate the gradient using the medians of the two windows:

$$g = (\text{median}(w_2) - \text{median}(w_1))/1 \tag{5.4}$$

(Since the windows differ by just a single element, the denominator is 1). For less noisy measures, such as the Lewis performance estimate, we can also use the arithmetic mean instead of the median. We then just replace "median" with "mean" in Equation 5.4.

We found that a window of size $k = 100$ yields good results in mitigating the noise while still reacting fast enough to the changes in the gradient. We combine this criterion with a maximum criterion and only stop if the last value a_n is a new maximum.

We stop the AL process when
(i) the current confidence or performance estimate is a new maximum, and
(ii) the newly calculated gradient g is positive, and
(iii) g falls below a predefined level ϵ.

Evaluation

Table 5.7 shows the results of gradient stopping applied to each of the three uncertainty measures and the Lewis performance estimate. For comparison, we also include results with a threshold-based criterion, where AL stops when the uncertainty measure of the selected instance reaches a threshold of $1 - \epsilon$. This is similar to Zhu and Hovy (2007) but extended by us to all three confidence measures.

In all cases, we performed 20 trials of active learning, each starting from different seed sets. The selection measure was *1-Entropy* in all cases. Table 5.7 shows results for each criterion. The "Stop" value indicates the iteration at which the stopping criterion triggered and stopped AL, the percentage values indicate the fraction of the pool that is annotated, which translates to the annotation effort saved.

"ΔBl" indicates the difference between baseline performance and performance at the stopping point, i.e. how much we miss (or gain) by stopping

Stop crit.	ϵ	Peak	Stop		Δ Bl	sd	Δ Pk	sd
1-Entropy threshold	0.01	80.8	3645	12.0%	1.44	0.7	−0.68	0.4
MinMax threshold	0.01	80.8	3133	10.3%	0.11	1.0	−2.0	0.8
Margin threshold	0.01	80.8	3158	10.4%	1.1	0.8	−1.0	0.8
1-Entropy gradient	0.00005	80.8	4572	15.0%	0.97	0.4	−1.1	0.5
MinMax gradient	0.00005	80.8	4397	14.5%	1.02	0.4	−1.1	0.5
Margin gradient	0.00005	80.8	5292	17.5%	0.81	0.3	−1.32	0.4
Lewis grd. (Median)	0.00005	80.8	2791	9.2%	0.8	1.4	−1.3	1.4
Lewis grd. (Mean)	0.00005	80.8	3999	13.1%	1.1	0.8	−0.95	0.6

Table 5.7: Performance at stopping points
(baseline perf. 78.7, Selection: *1-Entropy*).
ϵ: Stop threshold, "Peak": observed peak performance on dataset,
"Stop": amount of data after which stop criterion triggered,
"Δ Bl", "Δ Pk": performance difference to baseline or peak performance,
"sd": standard deviation of respective performance difference.

early instead of labeling the whole pool. "ΔPk" indicates the difference to peak performance. The "sd" columns show the respective standard deviations.

We find that all stopping criteria stop before 20% of the pool is used, providing a large reduction in annotation effort. While the point of peak performance cannot be precisely found by the criteria, all criteria reliably stop at a performance level that surpasses the fully supervised baseline. The threshold criteria seem to be a bit better in finding a stopping point closer to optimal performance. Not unsurprisingly, the stopping function that matches the selection function performs best. The gradient methods, however, seem to be providing a better-than-baseline performance slightly more consistently (less variation) and might require less tuning of the threshold parameter when other factors (e.g., the batch size) change. If lower noise allows it, as for the Lewis estimate, moving averages should be used in place of moving medians.

The gradient of the performance estimate and the confidence value of the selected instance can successfully be used as a stopping criterion relative to the optimal performance that is attainable on a given pool. The criteria reliably

determine stopping points that result in performance that is better than the supervised baseline and close to the optimal performance. We believe that these criteria can be applied to any AL setting based on uncertainty sampling, not just NER.

If it turns out that the maximum possible performance does not meet a user's expectations, the user needs to acquire fresh data and refill the pool. This might lead to an approach to reduce the computational cost of AL: subdivide a large sampling pool into smaller sub-pools; run AL sequentially on the sub-pools; switch to the next sub-pool when the stopping criterion is reached.

We also found that uncertainty curves of the selected examples are quite noisy. It would be interesting to see which properties of the training examples cause these drops in the certainty curve.

5.3 Related Work

Around the time the work on stopping criteria that the above chapter is based on was published, there was a flurry of related research on stopping criteria on active learning. The stopping criteria in related work can roughly be divided into two categories: *confidence-based* criteria, which try to determine the stopping point by using the classifier's confidence or uncertainty ratings, and *prediction-based* criteria, which use the classifier's prediction as a signal for stopping.

Confidence-based criteria

The most prolific researchers on stopping criteria are J. Zhu and his colleagues. They proposed two criteria that are directly derived from classifier confidence: *max-confidence* (Zhu and Hovy, 2007), where iteration stops when the uncertainty of the most informative unlabeled example falls below a user-defined threshold, and *overall-uncertainty* (Zhu et al., 2008b), which stops when the uncertainty of the remaining unlabeled examples in the pool falls below a threshold. They also propose a third criterion, *minimum expected error* (Zhu

et al., 2008a), which stops when estimated classifier performance falls below a user-defined threshold. Performance is estimated based on the classifiers posterior probability estimates, similar to Lewis's method. How to tune the thresholds is not clear and might differ from task to task. All three criteria are compared in Zhu et al. (2010).

Ghayoomi (2010) proposes to use the variance of the confidence scores as the stopping criterion, which stop if the variance falls below a threshold.

Vlachos (2008) observed a rise-peak-drop pattern of confidence on a separate unlabeled reference set and proposed to use detection of this pattern as a stopping criterion. While this is appealing as it would not need parameter tuning, we could not observe this kind of pattern in our experiments, as it might be specific to the task or the employed classifier.

Specific to the support vector machine classifier is the *margin exhaustion* stopping criterion proposed by Schohn and Cohn (2000). The criterion stops when, in SVM active learning, none of the selected examples are closer to the hyperplane than the already existing support vectors. In this case, adding these examples to the training set will not further change the model. Campbell et al. (2000) propose to combine this criterion with sampling some extra examples for a held-out error estimate. In a similar vein, Dimitrakakis and Savu-Krohn (2008) combine an estimation of the expected error with some extra labeled examples.

Donmez et al. (2007) use a gradient of estimated performance (similar to our "Lewis gradient", but with 0/1 loss [accuracy]) to control switching between two strategies of active learning. Small and Roth (2010) adopt this technique to control active learning for a pipeline of structured prediction tasks. These approaches are not stopping criteria in the literal sense but control the active learning process in complex ensemble active learning setups.

For the problem of stopping by minimum absolute performance, Sawade et al. (2010a) proposed to sidestep the problem of classifier performance self-estimation by using an active-learning-style procedure to *selectively sample an evaluation dataset*. This dataset is drawn using a sampling distribution designed to minimize the performance estimation error. This procedure needs

some extra annotation effort to create the labeled evaluation dataset, but this is smaller than conventional randomly sampled held-out datasets. Sawade et al. (2010b) extend this procedure to the (non-sequence) F-Score performance measure.

Prediction-based criteria

The second class of stopping criteria uses the predictions made by the classifier(s) to control the iterations. Predictions are compared with gold labels or other predictions to make a stopping decision.

Tomanek et al. (2007), Tomanek and Hahn (2008), and Olsson and Tomanek (2009) propose various stopping criteria for query-by-committee AL that are based on the principle that if predictions of the committee members agree, the iteration should stop. They apply the criterion to different possible datasets, such as the remaining pool or a separate reference set.

Zhu et al. (2008b) propose the *classification-change* criterion, which is also applicable to uncertainty sampling or other types of AL that do not use a committee: AL should stop if predicted labels on remaining unlabeled examples do not change from one iteration to the other. They also investigate combinations with their confidence-based criteria.

Bloodgood and Vijay-Shanker (2009) refine the principle of comparing predictions from subsequent iterations to the *stabilizing predictions*. They compare the agreement of predictions made by successive models on a subset of the remaining pool. By using Cohen's Kappa (Cohen, 1960) for measuring the agreement, they avoid issues with imbalanced classes. The user needs to set a Kappa threshold for agreement intensity, plus a longevity threshold to avoid stopping too early caused by fluctuating agreement.

Zhu and Hovy (2007) also propose a criterion that also incorporates user feedback: the *Min-error* compares predicted labels of selected examples to true labels that are returned from the annotator and stops if the accuracy on these examples exceeds a threshold. This approach is only applicable to batch-mode AL and it is unclear if or how batch size affects the results. Also, the

accuracy estimate on these examples may not correspond well to true classifier performance, so there is again the question of how to set a suitable threshold.

As with the confidence-based approaches, prediction-based criteria are also used to provide finer control for ensemble methods in AL: Baram et al. (2004) use the entropy over classifier predictions on the unlabeled pool and a multi-armed bandit algorithm to choose between several AL strategies during the iteration.

5.4 Summary

We presented two different approaches to monitor the progress of active learning annotation. The first attempts to estimate the performance of the classifier trained on the annotated data, in order to stop annotation when a satisfactory performance level has been reached. The second monitors when learning has converged and classifier performance has reached a peak. In this case, further annotation would be unnecessary.

We first investigated a method that attempts to estimate performance on unlabeled data by computing expectations of TP, FP, and FN, and thus precision, recall, and F-Score. Comparing this with evaluation on labeled data, we find that performance estimates are too high. When trying to correct the estimation using bootstrapped empirical class probabilities, the correction overshoots and yields too pessimistic performance estimates. We therefore conclude that further work would be needed for direct performance estimation without labels.

Nevertheless, the gradient of these performance estimates can be used to predict when learning has converged to a point where the maximal performance that is attainable with a given AL setup is reached. A similar criterion can also be derived from the gradient of the confidence measures of the last selected instances: when the value of these gradients approaches 0, performance has converged. Experiments show that these two stopping criteria, performance convergence and uncertainty convergence can reliably stop annotation close to the point of maximal possible performance.

6 Using Crowdsourcing for Active Learning

6.1 Introduction

6.1.1 Crowdsourcing

The active learning experiments in the previous chapters have been performed by simulating the annotation process with preexisting gold-labeled data, but it is also important to perform experiments with actual human annotators.

Recently, *crowdsourcing* has been proposed as a way of performing human annotation (Snow et al., 2008). The term *crowdsourcing* is a combination of the words *crowd* and *outsourcing*, and means distributing human labor tasks that can be performed on a computer (such as annotation tasks) to large groups of potential workers via the World Wide Web. In traditional forms of outsourcing, the relationship between clients and agents working on the task is established beforehand. Usually, there exists a negotiated contract between client and outsourcing provider, or—a solution not pertaining strictly to outsourcing, but a popular setup for annotation tasks at least in academia— the agents are temporary employees.

Crowdsourcing, however, aims to distribute tasks just in time when they arise via an open call to a group of workers whose identity and qualifications are usually not known to the client beforehand (Alonso and Lease, 2011).

Ipeirotis (2012) defines crowdsourcing (or "crowd labor") as follows:

> Cloud labor is a model for enabling convenient, on-demand network access to a (shared) pool of human workers with different skills (e.g., transcribers, translators, developers, virtual assistants, graphic designers, etc) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Crowdsourcing can be implemented in different ways (cf. Quinn and Bederson, 2009). *Games with a purpose* (von Ahn, 2006) in which people perform annotation tasks as a side effect of a cleverly constructed game they play purely for fun are one example. Some even consider collaborative content creation on sites such as Wikipedia, where people are motivated by altruism or seeking personal reputation, to be a form of crowdsourcing (Alonso and Lease, 2011). However, the form of crowdsourcing that is most relevant for annotation for NLP is *micro-task* crowdsourcing. With micro-tasking, tasks are split into very small pieces of work which are offered to the workforce for a small price. Here, the motivation of the workers is monetary income. For example, for a linguistic annotation task, the task could be split into annotating each sentence individually.

Microtask market platforms, such as Amazon Mechanical Turk (MTurk) or SamaSource, accept task offers which they then advertise to their pools of workers. These platforms also handle billing and payment of workers as well as some basic handling of complaints and selection of qualified workers. Amazon Mechanical Turk is the most popular microtask crowdsourcing platform, which will also be the platform we will be using for the work presented in this chapter.

6.1.2 Noisy Annotations

One caveat of using crowdsourcing for linguistic annotation is that the workers usually have no linguistic training. This means that annotation tasks need to be phrased in such a way that linguistic laypersons are able to understand and work on them.

It also means that crowdsourcing workers are more likely to make mistakes while performing their tasks, leading to *noisy*, i.e. sometimes incorrect, annotations. While some level of noise is present in any annotation performed by humans, this issue is much more prominent with crowdsourced annotations. One reason is, as already mentioned, the fact that workers in crowdsourcing lack specific training for the task. Another reason is that crowdsourcing workers often perform their tasks between their usual responsibilities and thus are devoting less attention to the task than annotators in a controlled lab scenario. Furthermore, there is a non-negligible number of workers who try to "cheat" or "spam", i.e. try to get by with submitting answers to tasks without putting in the effort that is needed for a proper answer. Whatever the reason for annotation errors, it is important that the quality of the annotations obtained by crowdsourcing is controlled and annotation errors are filtered out.

Active learning and noisy annotation

Noisy annotations present a particular challenge to active learning setups: Example selection depends on accurate assessment of example informativeness, which in turn depends on the set of examples that have been labeled so far. If these labels are noisy, the informativeness assessments may be unreliable, leading to the selection of uninformative examples—which might be "misleading" for the procedure that selects further examples.

However, most active learning research has ignored this issue and focused on *simulated* active learning, where the annotation process is simulated by an oracle that just inserts labels from a previously annotated high-quality gold-labeled dataset. This approach is useful as it allows testing new AL approaches without needing to expend a lot of time and money on annotation experiments for potentially sub-par approaches. This is the reason why the experiments in the previous chapters were also performed using simulations on gold data. However, Baldridge and Palmer (2009) and Rehbein et al. (2010) have questioned the applicability of simulated active learning results to noisy labels in real-world annotation scenarios.

Thus, the high noise level of crowdsourced annotations is both burden and boon: on the one hand it requires us to devise methods for explicit quality control of annotations; on the other hand it allows us to investigate the robustness of active learning algorithms in a noisy environment.

6.2 Annotation System

6.2.1 System Architecture

We designed a system that performs annotation with active learning using the popular crowdsourcing platform Amazon Mechanical Turk. One fundamental design criterion for our annotation system was the ability to select examples *in real time* to support the interactive select-annotate-retrain loop that is essential to active learning.

The base unit of work on the Mechanical Turk platform is called *Human Intelligence Task*. Each HIT is offered to workers on the platform and may, for example, contain one or more examples for annotation. The usual workflow on MTurk works in batch mode. It assumes that people offering tasks (*requesters*) upload a number of HITs in a batch, where the examples contained in the HIT are fixed at the time of upload. After workers have processed the HIT, the requester can download the results from the MTurk website. Furthermore, this is essentially also the workflow that value-added services like CrowdFlower[1] support. CrowdFlower provides services like management of tasks and quality control of workers and annotations on top of MTurk and other crowdsourcing platforms. This workflow, however, is too rigid for interactive annotation in an active learning framework.

We therefore designed our own system for annotation experiments. It consists of a two-tiered application architecture with a front-end component and a back-end component (see Figure 6.1 for an illustration of interactions between worker, front-end component, and back-end component).

[1] Available at http://crowdflower.com/

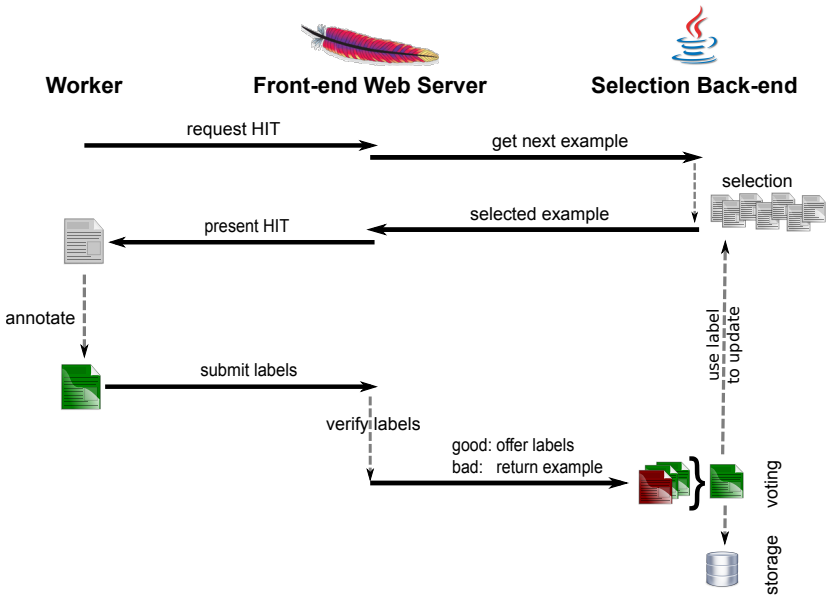


Figure 6.1: Interaction of annotation worker and system components

Front-end

The front-end tier is a web application that serves two purposes: First, the administrator can manage annotation experiments using a web interface and publish annotation tasks associated with an experiment on MTurk. The front-end also provides tools for efficient review of the received answers. Mechanical Turk provides a programmatic interface to the platform via a Web Service API that is used by the front-end to publish HITs on the site and to approve or reject answers from workers after review.

Second, the front-end web application presents annotation tasks to MTurk workers. As we wanted to implement interactive annotation experiments, we used the "external question" feature of MTurk. An external question is a HIT that does not contain fixed content but instead contains a callback URL to our front-end web application. When a worker views the HIT, the callback URL is queried by the worker's browser. The front-end receives this query and can now fill in a suitable example.

In our system, it does this by in turn querying the back-end component for an example to be annotated and rendering it in HTML. This way, the system can at any time choose and display the example that currently is most informative for annotation.

Back-end

The back-end component is responsible for the selection of an example to be annotated in response to a worker's request for an annotation task. It implements a diverse choice of selection strategies (random sampling and active learning). It also features the relabeling strategies described in section 6.3.1 and concurrent retraining as described in section 6.2.2. The back-end component runs as a stand-alone server and is queried by the front-end via REST-like HTTP remote procedure calls.

When the administrator starts an annotation task using this system, the system creates a predefined number of HITs. These HITs are posted on the MTurk platform, but they do not contain the actual example yet. Only when a worker views the contents of a HIT to work on it does an actual example get queried from the selection back-end. As soon as an answer from a worker is submitted to the MTurk platform, the front-end again gets notified and forwards the answer to the selection back-end. The back-end can then update the example selection according to the new information.

Depending on the relabeling strategies used, some examples may be repeated for quality improvement, so the back-end may choose to respond to several HIT queries with the same example. Thus, the number of distinct examples that are labeled may be lower than the number of HITs posted.

6.2.2 Concurrent Example Selection

In the select-annotate-retrain cycle of active learning, both the "select" and the "retrain" step can take a significant amount of CPU processing time. The time for the "retrain" step depends mainly on the type of classifier that is used and the size of the labeled set. The time for the "select" step depends on the

time required to estimate the informativeness of each example and on the size of the unlabeled pool. If an AL system requires significant time for the retrain and select steps, annotators may need to wait before they can work on the next example. This is undesirable as it breaks the concentration of the annotators and damages efficiency (Haertel et al., 2010).

Traditional AL implementations therefore often select examples in batches of the most informative examples. However, batch selection might not give the optimum selection, as examples in a batch are likely to be redundant (Brinker, 2003). Even then, wait times can still occur between one batch and the next. Other approaches to reduce wait times include using a faster, yet similar, learning algorithm for selection (Tomanek et al., 2007) to speed up the retraining step, or speeding up the rescoring step by approximative techniques, for example by rescoring only a subset of the pool (Segal et al., 2006) or by using locality-sensitive hashing (Jain et al., 2010). However, most of these techniques are limited to specific classifiers and may still cause the annotator to wait for the next example.

When performing annotation with Mechanical Turk, it is even more important to avoid wait times when workers request a HIT (an example) to work on. Workers will not accept waiting long times for tasks that only pay a few cents per example and can easily switch to other tasks that are available on the same platform. Furthermore, traffic on crowdsourcing platforms can be bursty: many different annotators can be interested in working on the task at the same time, leading to a spike of many requests in a very short time that can quickly exhaust a batch of selected examples, again forcing workers to wait while the next batch is being filled by the system.

We therefore decouple classifier retraining and uncertainty rescoring from handling annotation user requests and perform both parts concurrently. Retraining and rescoring are performed within a second compute thread. The unlabeled pool is stored in a priority queue that is ordered according to the informativeness of the examples. When an annotator requests an example, the annotation user interface takes the most informative example from this pool queue and presents it to the annotator. After annotation, the now labeled ex-

ample is then inserted into a second queue that feeds the retraining+rescoring process.

The retraining+rescoring runs in a continuous loop. At the start of a retraining+rescoring cycle, the process takes all examples that have been accumulated in the second queue and adds them to the labeled set. It then retrains the classifier(s) on the labeled set and proceeds to rescore the informativeness rating on the remaining unlabeled examples. After retraining and rescoring, the pool queue then is sorted again according to the new informativeness. In this way, annotation and example selection can run in parallel. This is similar to Haertel et al. (2010).

6.2.3 User Interfaces for the Annotation Tasks

We conducted experiments with AL on Mechanical Turk for two different NLP tasks: Named entity recognition and sentiment classification of documents. In this section, we describe the user interfaces that we designed for annotation on Mechanical Turk. The annotation interface runs in the worker's browser, which restricts the implementation to a combination of HTML and client-side JavaScript code. This precludes most off-the-shelf annotation tools, like, for example, MMAX2 (Müller and Strube, 2006), but thanks to modern advanced JavaScript still gives enough freedom for the development of annotation interfaces. When designing user interfaces for Mechanical Turk, it is, however, important to keep in mind that the interface needs to be understandable without training.

Named entity recognition (NER)

For the NER task, we present one sentence per HIT. The sentence is presented segmented into tokens, with a select box underneath each token containing the entity classes, as well as "–", corresponding to the "O" label for tokens that are not part of a named entity. (Internally, the system uses "O", in line with the CoNLL tag set.) Figure 6.2 shows a screenshot of the annotation interface.

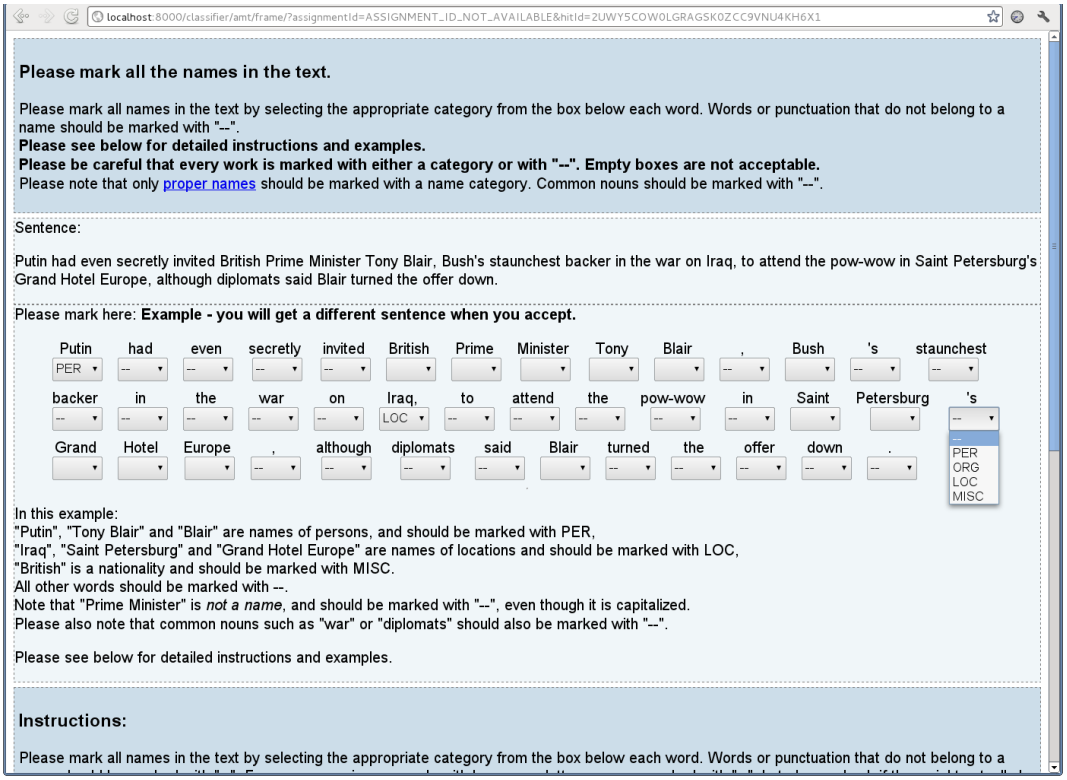


Figure 6.2: Annotation interface for the NER task

The selection of entity classes is similar to the BIO encoding scheme described in section 3.1.1, but to make the task simpler for the annotators, we leave away the distinction between tokens beginning ("B-") or continuing ("I-") a sequence of entity tokens. In most cases (except back-to-back entities of the same type), the B- and I- prefixes can be recreated. Below the sentence input area there are definitions of the classes, based on the CoNLL-2003 annotation guidelines (Tjong Kim Sang and De Meulder, 2003) but rephrased to be more intuitively understandable for laymen. Examples were given for every class.

In order to avoid annotators submitting examples without actual annotation, they are forced to make a selection for uppercase tokens. This constraint is checked using client-side JavaScript before the annotator is allowed to submit her input, and it is validated a second time on the server side in the front-end component to catch manipulation. In a similar way, we implemented some simple heuristics to catch obvious attempts at cheating, such as submitting examples with all tokens set to "MISC". However, we considered it too tedious to force annotators to click on a selection box for every token of the sentence, even if it was an obvious non-entity token. Therefore, lowercase tokens were prelabeled with "–" (no named entity, corresponds to the O label of the (B)IO encoding), except when they are between uppercase tokens. For example, in the phrase "Louis van Gaal", the token "van" requires the choice of a class, but not in the phrase "Rafael van der Vaart". Annotators are encouraged to change the "–" label if a lowercase token is in fact part of an entity phrase.

Sentiment classification

For sentiment annotation, we asked the annotators to rate documents if they expressed a positive or a negative sentiment toward the topic discussed. We found in preliminary experiments that using simple radio button selection for the choice of the document label ("positive" or "negative") leads to a very high amount of spam submissions, taking the overall classification accuracy down to around 55%.

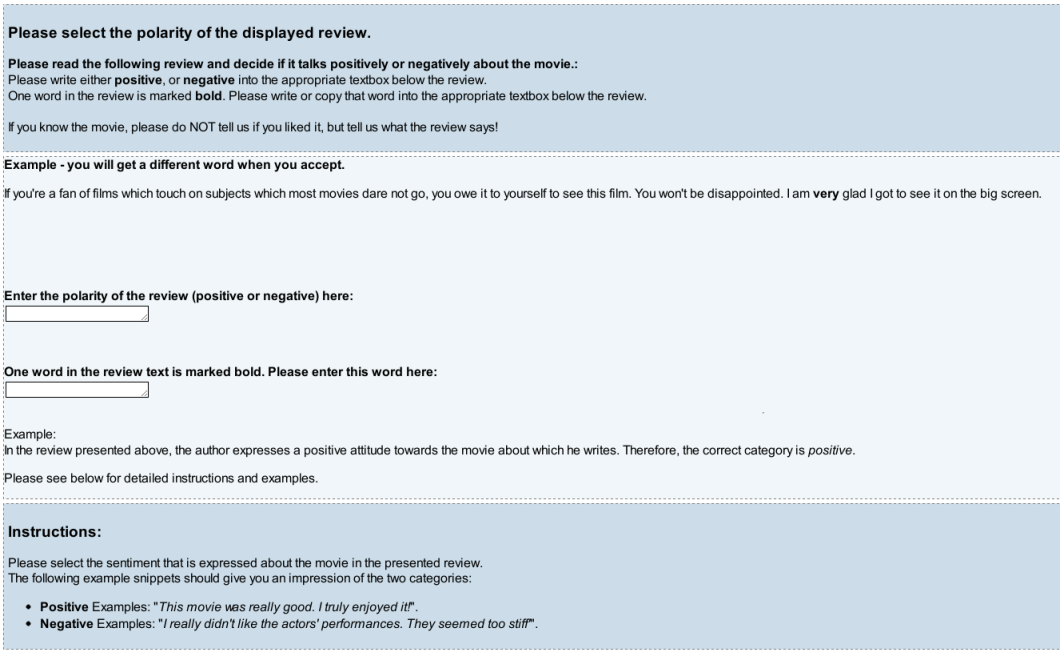


Figure 6.3: Annotation interface for the sentiment task

We then designed an annotation template that forced annotators to type the label as well as a randomly chosen word from the text, as a check to see that the annotator did in fact read the document before entering the rating (see Figure 6.3). The validation of this field was again done in the front-end server to be safe against manipulation. Individual label accuracy was around 75% in this scheme.

6.2.4 Active Learning Setup

We set up the active learning core in the back-end module as follows: we used the concurrent example selection described in section 6.2.2, which is essentially a pool-based AL approach, as the rescoring, and the selection of the most informative example was performed on the entire pool of unlabeled examples.

Informativeness is calculated using uncertainty sampling, with the margin measure as the base uncertainty measure. For NER, the margin measures of individual tokens were averaged to get an uncertainty assessment of the sentence (as in the S-AL setup in section 4.2.3). For sentiment, whole documents were classified with a single label, thus uncertainties could be used directly.

The selected example was removed from the unlabeled pool for annotation. Once its labels had been obtained, it was added to the set of labeled examples via the labeled queue. The classifier was then retrained on the labeled examples, and the informativeness of the remaining examples in the pool was re-evaluated.

6.2.5 Annotation Log Recording and Replay

We want to avoid rerunning experiments on Mechanical Turk over and over again, but we also believe that using synthetic data for simulations is problematic because it is difficult to generate synthetic data with a realistic model of annotator errors. Thus, we logged a play-by-play recording of the annotator interactions and labels. This recording is designed to capture every interaction of the annotator with the front-end system, every interaction of front-end and back-end systems, and every important decision within the back-end system. Using these recordings it should be possible to create a replay of the interactions which behave exactly like the original. For example, one should be able to create a set of virtual annotators that request HITs and provide answers in the same order as the original workers on MTurk. We therefore create recordings on both the front-end and the back-end server.

The front-end records:

- Interactions with the MTurk workers: display of a HIT, submission of a completed annotation for a HIT or notification from the MTurk platform that a HIT was returned by the worker or not completed in time.
- Decisions of spam mitigation heuristics: automatic rejection of annotations that trigger anti-spam heuristics, blocking of workers that repeatedly trigger anti-spam heuristics.
- Interaction with the back-end: requests for the next example, forwarding of a completed annotation, notification that a HIT was not completed and needs to be processed again.

The back-end records:

- Interaction with the front-end: requests for next example, completed annotations, notification that an example was not completed.
- The voting and example selection layer further also records:
  - Tentatively adding a labeled example to the set of labeled examples while the final label is being determined by multiple annotation and voting (see section 6.3.1 for details).
  - Updating a tentatively labeled example with the final example from the voting strategy.
  - Retraction of a tentatively labeled example from the labeled pool, in the event that a voting strategy decided to discard this example.
  - Request to put an example that could not be annotated back into the pool queue.

The playback of the play-by-play log provides a faithful reproduction of worker or system actions, but they cannot invent annotations that workers did not submit. As a consequence, if the playback is run against a multiple annotation strategy that waits for more repeated annotations for one example than

are recorded or it requests examples for which no annotations are recorded, these examples can't be completed.

In some cases, replaying a play-by-play log against a differently parameterized system therefore results in a labeled set that is smaller than the budget would allow. Consequently, in some cases we requested extra annotation on MTurk to fill in the missing annotations.

6.3 Strategies for Dealing with Label Noise

6.3.1 Adaptive Voting

MTurk labels often have a high error rate, or high label noise. A common strategy for improving label quality is to acquire *multiple labels* from different workers for each example and then consolidate the annotations into a single label of higher quality. To cut costs, however, we still want to keep the number of queried examples as low as possible. To trade off the number of annotated examples against the quality of annotations, we propose *adaptive voting*.

It uses majority voting and is adaptive in the number of repeated annotations. For NER, a sentence is first annotated by two workers. Then majority voting is performed for each token individually. If there is a majority for every token that is greater than an *agreement threshold* α, the sentence is accepted with each token labeled with the majority label. Otherwise additional annotations are requested.

To avoid spending excessive annotation effort on a few difficult examples, we also introduced a *discard threshold* d. If the number of repeated annotations exceeds d, the example is discarded. In what follows, we call this approach d-voting.

In this scheme, it can take a while for annotators to agree on a final annotation for a sentence. We make *tentative* labels of a sentence available for classifier retraining immediately and replace them with the final labels once voting is completed. Similarly, when the discard threshold is hit and there is no final label, the sentence gets retracted from the labeled set. This way, we avoid

selecting redundant examples because of stale uncertainty ratings by providing labels as soon as possible. At the same time, the update with the final label serves to avoid "misleading" the classifier, and thus the AL selection, with erroneous labels.

We use the same scheme for sentiment; note that there is just one decision per HIT in this case, not several as in NER.

6.3.2 Fragment Recovery

For NER, we also use *fragment recovery*: we salvage tokens with agreeing labels from discarded sentences. We cut the token sequence of a discarded sentence into several fragments that have agreeing tokens and discard only those parts that disagree. We then include these recovered fragments in the training data just like complete sentences.

Since we are using a CRF sequence model, we are making a minor mistake here, as these fragments have sequence beginnings that may be slightly different from the beginning of regular sentences. Our experiments nevertheless showed a small but significant improvement when using fragments.

We experimented with different minimal lengths of fragments and with the requirement that fragments do not start with an entity, to minimize the issue with sentence beginnings. Since the performance differences of different parameter settings were only very small, we will only report on one parameter combination in section 6.4, using fragments with a minimum length of 2 tokens and allowing entity tokens at the beginning of a fragment.

6.4 Experiments with Crowdsourcing on Amazon Mechanical Turk

6.4.1 Experiment Setup

NER experiment setup

In our NER experiments, we had workers reannotate the English corpus of the CoNLL-2003 NER shared task. We chose this corpus to be able to compare crowdsourced annotations with gold standard annotations. This way we could measure the annotation quality against gold labels and also compare models trained on the crowdsourced data to models trained on gold data.

A HIT is one sentence and is offered for a base payment of \$0.01. This also determines the unit of annotation cost that we use for reporting performance relative to annotation cost: one HIT is one cost unit. Including Amazon's commission, we could get 6,931 sentence annotations with a budget of \$100.

We filtered out answers that contained unannotated tokens or were obvious spam (e.g. all tokens labeled as "MISC"). For example selection and testing NER performance, we used a system similar to the one that we used when performing the sentence selection experiments in section 4.2. It is based on conditional random fields with standard named entity features including the token itself, orthographic features like the occurrence of capitalization or special characters and context information about the tokens to the left/right of the current token. NER performance is evaluated as F_1 on the CoNLL-2003 test set A.

Sentiment experiment setup

The sentiment detection task was modeled after a well-known document analysis setup for sentiment classification, introduced by Pang et al. (2002). We use their corpus of 1,000 positive and 1,000 negative movie reviews[2] and the

[2]Available at http://www.cs.cornell.edu/people/pabo/movie-review-data/

Stanford maximum entropy classifier (Manning and Klein, 2003) to predict the sentiment label of each document d from a unigram representation of d. We randomly split this corpus into a test set of 500 reviews and an active learning pool of 1,500 reviews. Each HIT consists of one document, valued at \$0.01. Again, one cost unit corresponds to one HIT.

Example selection setup

We compare random sampling (RS) and active learning (AL) in combination with the proposed voting and fragment strategies with different parameters. Since we wanted to avoid rerunning experiments on Mechanical Turk over and over again (and spend money on the reruns), we used the play-by-play log facility (described in section 6.2.5) to replay annotator interactions with different parameter settings.

We chose adaptive voting with at most $d = 5$ repetitions as our main re-annotation strategy for both random sampling and active learning for NER annotation. We use simple majority voting ($\alpha = .5$) for NER.

For sentiment, we set $d = 4$ and minimum agreement $\alpha = .75$ because the number of labels is smaller (2 vs. 5) and so chance agreement is more likely for sentiment. We chose $d = 4$ repetitions for sentiment because with two classes and $\alpha = .75$, getting a fifth annotation can never change the result of the voting compared to just 4 repetitions.

To get results for 3-voting NER, we take the recording and discard 5-voting votes not needed in 3-voting. This results in roughly the same number of annotated sentences, but at a lower cost. This simulation of 3-voting is not exactly what would have happened on MTurk (e.g. the final vote on a sentence might be different, which then influences AL example selection), but we will assume that differences are rare and simulated and actual results are similar. The same considerations apply to single votes and to the sentiment experiments.

We always compare two strategies for the same annotation budget. For example, the *number of training sentences* in Table 6.1 differ in the two relevant columns, but all strategies compared use exactly *the same annotation budget*

(5,820, 6,931, 1,130, and 1,756, respectively). Because different strategies implement different trade-offs of sampling a larger number of distinct examples vs. expending more effort on getting repeat annotations for the same examples, the number of training examples for a given budget differs depending on the strategy used.

The main annotation strategy was 5-voting (or 4-voting for sentiment), which uses a large portion of the budget for reannotation. Due to this, for the single annotation strategy each interaction record contained only about 40% usable annotations, the rest were repeats. A comparison with the single annotation strategy over those approx. 2,000 sentences or 450 documents would not have been meaningful; therefore we chose to run an extra experiment with the single annotation strategy to match this up with the budgets of the voting strategies. The results are presented in two separate columns of Table 6.1 (budget 6,931) and Table 6.2 (budget 1,756).

6.4.2 Results

For sentiment detection, *label quality* or *worker accuracy*—the percentage of correctly annotated documents—is 74.8 (Table 6.2, line 1). In contrast, for NER, label quality—measured as the percentage of non-O tokens annotated correctly—is only 51.6 (Table 6.1, line 1). (Label quality over all kinds of tokens is around 90%, but O tokens are both easy to annotate and pre-annotated by the user interface, so quality on non-O tokens is more relevant.) This demonstrates the challenge of using MTurk for NLP annotation tasks.

When we use single annotations of each sentence, NER performance is 59.6 F_1 for random sampling (Table 6.1, line 1). When training with gold labels on the same sentences, the performance is 80.0 F_1 (not shown). This means we lose more than 20 points due to poor worker accuracy. Adaptive voting and fragment recovery manage to recover a small part of the lost performance (lines 2–4); each of the three F_1 scores is significantly better than the one above it as indicated by †. (Statistical significance was determined using

NER			Budget: 5820				6931	
			#train	F_1	cost/sent	label-qual.	#train	F_1
RS	1	single	5820	59.6	1.00	51.6	–	–
	2	3-voting	1624	61.4†	3.58	70.1	–	–
	3	5-voting	1488	63.0†	3.91	71.6	1774	63.5
	4	5-voting+f	1996	63.6†	2.91	71.8	2385	64.9†
AL	5	single	5820	67.0	1.00	66.5	–	–
	6	3-voting	1808	70.0†	3.21	78.8	–	–
	7	5-voting	1679	70.4†	3.46	79.6	1966	70.6
	8	5-voting+f	2165	70.5	2.68	79.3	2691	71.2

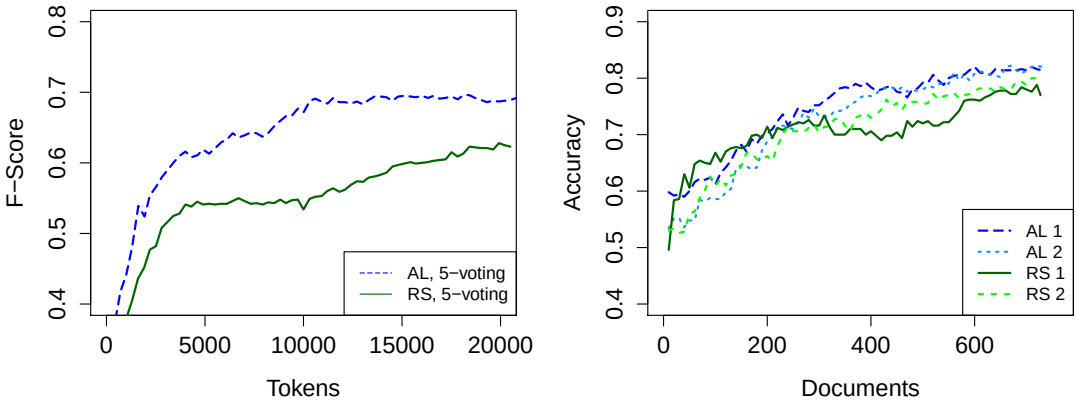
Table 6.1: NER results on Mechanical Turk
Columns: #train = number of sentences in training set, F_1 = performance on test set, cost/sent = average annotation cost per sentence, label-qual. = quality of annotated labels.
Rows: +f = using fragments

Sentiment			Budget: 1130				1756	
			#train	Acc	cost/doc	label-qual.	#train	Acc
RS	1	single	1130	70.4	1.00	74.8	–	–
	3	4-voting	450	71.2	2.51	89.6	735	79.2
AL	5	single	1130	74.8	1.00	76.0	–	–
	7	4-voting	455	77.4	2.48	89.0	715	81.8

Table 6.2: Sentiment classification results on Mechanical Turk. Sentiment budget 1130 for run 1, sentiment budget 1756 averaged over 2 runs.
Columns: #train = number of sentences in training set, Acc = performance on test set, cost/sent = average annotation cost per sentence, label-qual. = quality of annotated labels.

the Approximate Randomization Test (Noreen, 1989; Chinchor et al., 1993) as implemented by Padó, 2006).

Using AL turns out to be quite successful for NER performance. For single annotations, NER performance is 67.0 F_1 (line 5), an improvement of 7.4 points compared to random sampling (line 1). Adaptive voting and fragment recovery again increase worker accuracy (lines 6–8), although the total improvement of 3.5 points (lines 8 vs. 5) is smaller than 4 points for random sampling (lines 4 vs. 1).



(a) AL vs. RS on NER

(b) AL vs. RS on sentiment

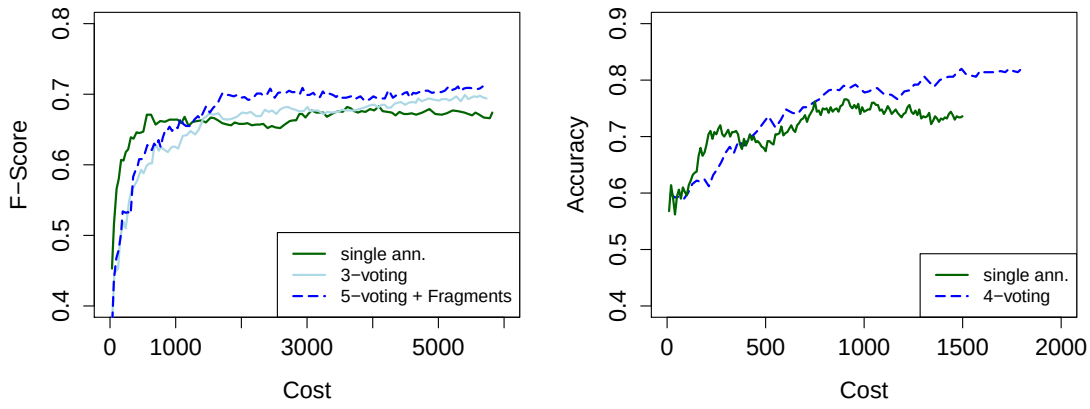
Figure 6.4: Active learning (AL) vs. random sampling (RS) for NER (left) and sentiment (right)

The learning curves of AL vs. RS in Figure 6.4(a) confirm this result for AL. These learning curves show performance relative to *tokens*—not sentences—to show that the reason for AL's better performance is not that it selects slightly longer sentences than random sampling, rather that the AL-selected examples provide additional value for classifier training. In addition, the relative advantage of AL over RS decreases over time, which is typical of pool-based AL experiments.

We carried out two runs of the same experiment for sentiment (Figure 6.4(b)) to validate our first positive result since the difference between the two conditions is not as large as in NER (Figure 6.4(a)). After about 300 documents,

active learning consistently outperforms random sampling. The first AL run performs better due to higher label quality in the beginning. The overall advantage of AL over RS is lower than for NER as the set of labels is smaller in sentiment, making the classification task easier. Second, there is a large amount of simple lexical clues for detecting sentiment (cf. Wilson et al., 2005). It is likely that some of them can be learned well through random sampling at first; however, active learning can gain accuracy over time as it selects examples with more difficult clues.

6.4.3 Effectiveness of Noise Mitigation Strategies



(a) NER

(b) Sentiment

Figure 6.5: Active learning: adaptive voting vs. single annotation for NER (left) and sentiment (right).

In Figure 6.5, we compare single annotation with adaptive voting. The graphs show F_1 as a function of cost. Adaptive voting trades quantity of sampled sentences for quality of labels and thus incurs higher net costs per sentence. This results in a smaller dataset for a given budget, but this dataset is still more useful for classifier training. For NER (Figure 6.5(a)), the single annotation strategy has a faster start; so for small budgets, covering a somewhat

larger portion of the sample space is beneficial. For larger budgets, however, quality of the voted labels trumps quantity.

For sentiment (Figure 6.5(b)), results are similar: voting has no benefit initially, but as finding maximally informative examples to annotate becomes harder in later stages of learning, adaptive voting gains an advantage over single annotations.

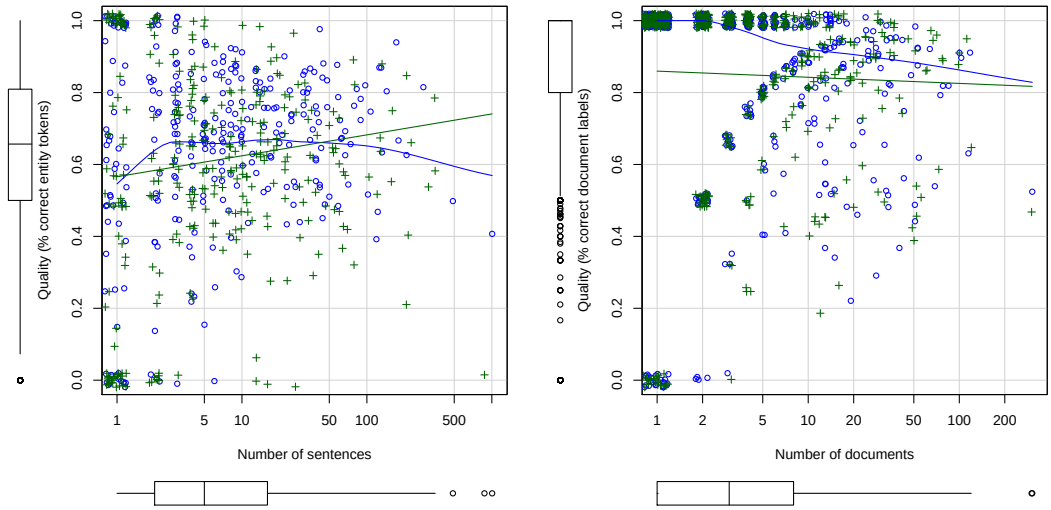
The main result of the experiment is that active learning is better by about 7 points F_1 than random sampling for NER and by 2.6 points accuracy for sentiment (averaged over two runs at budget 1,756). Adaptive voting further improves AL performance for both NER and sentiment.

6.5 Oracle Experiments with Quality Ratings

So far we have assumed that all workers provide annotations of the same quality. However, this is not the case. Figure 6.6 shows plots of worker accuracy as a function of worker productivity (number of annotated examples). Like in the previous section, worker accuracy is measured as the percentage of correct entity tokens for NER or correct documents for sentiment, compared to the gold standard.

Some workers submit only one or two HITs just to try out the task. For NER, the majority of workers submit between 5 and 10 sentences, with label qualities between 0.5 and 0.8. The chance level for correctness is around 0.25 (four different named entity categories for uppercase tokens). For sentiment, most workers submit 1 to 5 documents, with label qualities between 0.5 and 1. Chance level lies at around 0.5 (for two equally distributed labels).

While quality for highly productive workers is mediocre in our experiments, other researchers have found extremely bad quality among their most prolific workers (Callison-Burch, 2009). Some of these workers might be spammers who try to submit answers with automatic scripts. We encountered some spammers our heuristics (see section 6.2.3) did not detect (shown in the bottom-right area



(a) NER (b) Sentiment

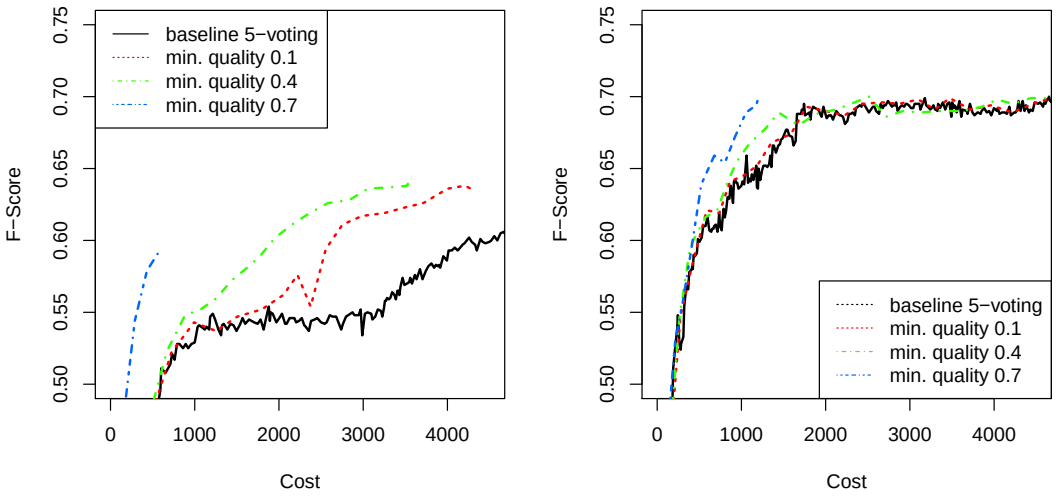
Figure 6.6: Worker accuracy vs. number of HITs. Each point corresponds to one worker (∘ = active, + = random sampling).
Left: NER. Right: Sentiment.

of Figure 6.6(a)), but the voting mechanism was able to mitigate their negative influence.

Given the large variation in Figure 6.6, using worker quality in crowdsourcing for improved training set creation seems promising. An obvious way seems to be the use of quality ratings as weights in the voting procedure. Surprisingly, we found that quality-weighted voting yielded no noticeable performance increase in our experiments. Instead, we test two different strategies: excluding low-quality workers and bypassing voting for high-quality workers. We now test these strategies for NER in an oracle setup.

6.5.1 Blocking Low-Quality Workers

The first approach is to refuse annotations from workers that have been determined to provide low quality answers. We simulated this strategy on NER



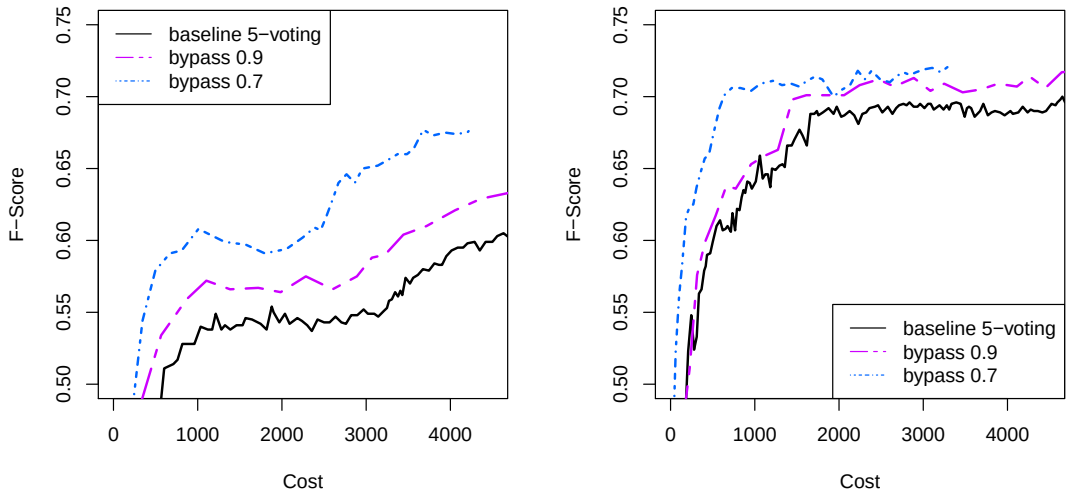
(a) Filtering low quality workers (RS)

(b) Filtering low quality workers (AL)

Figure 6.7: Performance when blocking low-quality workers at different thresholds

data using oracle quality ratings. Oracle quality ratings are determined by comparing worker-submitted labels to the gold standard. Every worker gets an individual quality rating determined over all of his/her submitted examples. How to estimate worker quality ratings when no gold labels are available in an online fashion is an open question.

We chose to use the NER task for this experiment because of its lower overall label quality. The results are presented in Figure 6.7 for random sampling (a) and active learning (b). For RS, quality filtering with low cutoffs helps by removing bad annotations that likely come from spammers. While the voting strategy prevented a performance decrease with bad annotations, it needed to expend many extra annotations for correction. With filtering, these extra annotations become unnecessary and the system can learn faster. When low-quality workers are less active, as in the AL dataset, we find no meaningful performance increase for low cutoffs up to 0.4. For very high cutoffs (0.7), the



(a) Bypassing voting (RS)

(b) Bypassing voting (AL)

Figure 6.8: Performance when bypassing voting for high-quality workers

beginning of the performance curve shows that further cost reductions can be achieved. However, we did not have enough recorded human annotations available to perform a simulation for the full budget.

6.5.2 Trusting High-Quality Workers

The complementary approach is to take annotations from highly rated workers at face value and immediately accept them as the correct label, *bypassing* the voting procedure. Bypassing saves the cost of repeated annotation of the same sentence. Figure 6.8 shows learning curves for two bypass thresholds on worker quality (measured as the proportion of correct non-O tokens) for RS (Figure 6.8(a)) and AL (b). Bypassing performs surprisingly well. We find a steeper rise of the learning curve, meaning less cost for the same performance. We find not only substantial cost reductions but also higher overall performance. We believe this is because high-quality annotations can sometimes be

voted down by other annotations. If we can identify high-quality workers and directly use their annotations, this can be avoided.

These experiments are oracle experiments using gold data that are normally not available. In future work, we would like to repeat the experiments using methods for worker quality estimation (Ipeirotis et al., 2010; Donmez et al., 2009). For AL, the choice as to which labels are used (as a result of voting, bypassing, or other) also has an influence on the selection. However, we had to keep the sequence of the selected sentences fixed in the simulations reported above. While our method of sample selection for AL proved to be quite robust even in the presence of noise, higher quality labels do have an influence on the sample selection (see section 6.7), so the improvement could be even better than indicated here.

6.6 Influence of Selection Type on Annotator Performance

6.6.1 Annotation Time

Most AL work assumes constant cost per annotation unit. This assumption has been questioned because AL might select hard examples that take longer to annotate (Hachey et al., 2005; Settles et al., 2008a). In annotation with MTurk, cost is not a function of annotation time because workers are paid a fixed amount per HIT. Thus, a unit cost per HIT is in fact the appropriate cost measure here.

Nevertheless, annotation time plays a part in whether workers are willing to work on a given task for the offered reward. This is particularly problematic for NER since workers have to examine each token individually. We therefore investigate for NER whether the time MTurk workers spend on annotating sentences differs for random sampling vs. active learning.

We first compute median and mean annotation times and number of tokens per sentence (Table 6.3). We see that most sentences are annotated in a very

	sec/sentence		tokens/sentence	
Strategy	median	mean	all	required
Random sampling	17.2	33.1	15.0	3.4
Active learning	17.8	33.0	17.7	4.0

Table 6.3: Annotation time per uppercase token

short time, but the mean is much larger than the median because there are outliers of up to eight minutes. (We had chosen 10 minutes as the maximum time for annotation. After that, a sentence gets returned to the queue to become available to a different worker.) AL tends to select slightly longer sentences as well as sentences with slightly more uppercase tokens that require annotation. In a more detailed analysis, we attempt to distinguish between
(i) the effect of more uppercase ("annotation required") tokens vs.
(ii) the effect of example difficulty.

We fit a linear regression model to annotation time vs. the number of uppercase tokens. For the regression fit, we removed all annotation times > 60 seconds. Such long times indicate distraction of the worker and are not a reliable measure of difficulty.

Figure 6.9 shows the distribution of annotation times for both cases combined and the fitted models for each. The model estimated an annotation time of 2.3 secs for each required token for RS vs. 2.7 secs for AL. We conclude that the difference in difficulty between sentences selected by RS vs. AL is small, but noticeable.

6.6.2 Quality Differences

Another effect to expect from difficulty could be lower annotation accuracy. We therefore examined the accuracies for each worker who contributed to both the active learning and the random sampling experiment. We found that in

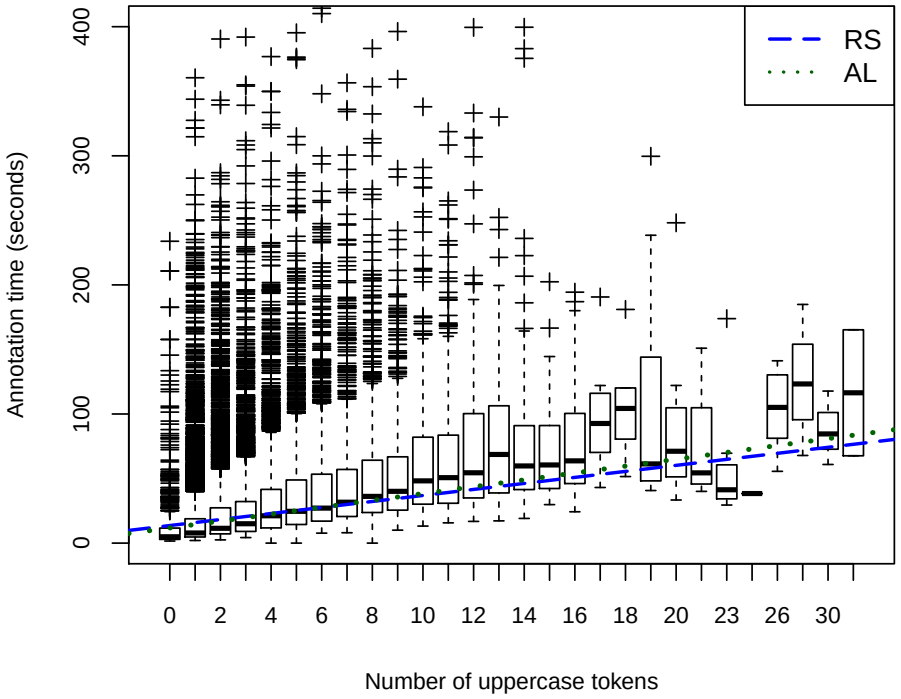


Figure 6.9: Annotation time vs. number of uppercase tokens

the NER task, the 20 workers in this group had a slightly higher (0.07) average quality for randomly selected (RS) examples. This difference is low and does not suggest a significant drop in accuracy for examples selected by active learning.

6.7 Influence of Noise on the Selection Process

While NER performance for AL is much higher than for random sampling, it is still quite a bit lower than what is possible on gold labels. In the case of AL, there are two possible reasons why this happens:

(i) The noisy labels negatively affect the classifier's ability to learn a good model that is used for classifying the test set.

(ii) The noisy labels result in bad intermediate models that then select suboptimal examples to be annotated next. The AL selection process is "misled" by the noisy examples.

We conduct an experiment to determine the contribution of factors (i) and (ii) to the performance loss. First, we preserve the sequence of examples chosen by our AL experiments on MTurk, with 5-voting for NER and 4-voting for sentiment but replace the noisy worker-provided labels with gold labels. The performance of classifiers trained on this sequence is the dashed line "MTurk selection, gold labels" in Figure 6.10 for NER (a) and sentiment (b).

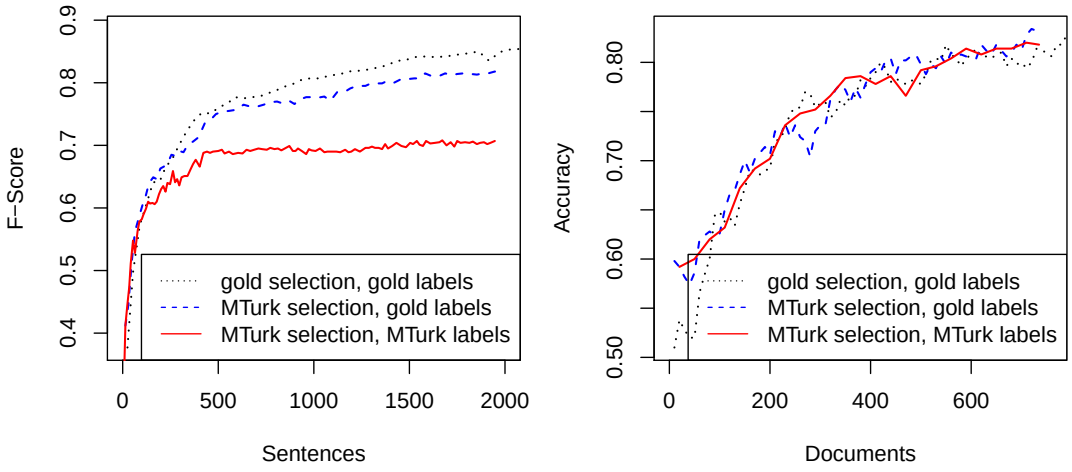
Second, we compare it with a traditional simulated AL experiment with gold labels. Here, the selection is also controlled by gold labels, so the selection has a noiseless classifier available for scoring and can perform optimal uncertainty selection. These are the dotted lines "gold selection, gold labels" in Figure 6.10.

Batch size of simulation

We used a batch-mode AL setup for this comparison experiment. For a fair comparison, we adjust the batch size to be equal to the average *staleness* of a selected example in concurrent MTurk active learning. The staleness of an example is defined as the number of annotations the system has received, but not yet incorporated in the computation of an example's uncertainty score (Haertel et al., 2010). Using the play-by-play log recordings we could determine the staleness of each selected example.

For our concurrent NER system, the average staleness of an example was about 12 (min: 1, max: 40); for sentiment it was about 2. The figure for NER is higher than the number cited by Haertel et al. (2010) because there are more annotators accessing our system at the same time via Mechanical Turk compared to the lab annotation setting of Haertel et al. It was not as high for sentiment since documents take longer to read and retraining the sentiment

classifier is faster. The average staleness of an example in a batch-mode system is half the batch size. Thus, we set the batch size of our comparison system to 25 for NER and to 4 for sentiment.



(a) NER (b) Sentiment

Figure 6.10: Performance on gold labels. Left: NER. Right: sentiment (run 1).

Results

Returning to the two factors introduced above—(i) final effect of noise on test set performance vs. (ii) intermediate effect of noise on example selection—we see in Figure 6.10 that (i) has a large effect on NER whereas (ii) has a noticeable, but small effect.

Note that in this experiment, our comparison unit for NER is the sentence. We cannot compare on cost here since we do not know what the per-sentence cost of a gold expert annotation is.

For example, at 1,966 sentences, F_1 scores are 70.6 (MTurk-MTurk), 81.4 (MTurk-gold) and 84.9 (gold-gold). This means that a performance difference of 10 points F_1 has to be attributed to noisy labels resulting in a worse final classifier (effect i), and another 3.5 points are lost due to sub-optimal example selection (effect ii).

For the *-gold variants, the performance has not yet converged. (For gold-gold, performance can go up to close to 90 F_1 at 6,000 sentences.) However, we did not have the budget to annotate the corresponding amount of sentences on Mechanical Turk for comparison in the MTurk-MTurk case, so we stopped at 1,966 sentences.

For sentiment, the results are different. There is no clear difference between the three runs. We attribute this to the fact that the quality of the labels is higher in sentiment than in NER. Our initial experiments on sentiment were all negative (showing no improvement of AL compared to RS) because label quality was too low. Only after we introduced the template described in section 6.2 and used 4-voting with $\alpha = .75$ did we get positive results for AL. This leads to an overall label quality of about 90% (over all runs) which is so high that the difference to using gold labels is small, if present at all.

6.8 Related Work

6.8.1 Crowdsourcing for NLP

Pioneered by Snow et al. (2008), crowdsourcing has become a widely used service in the NLP community, especially using Mechanical Turk.

Since then, crowdsourcing has been applied to obtain annotations for human judgments for several tasks, for example machine translation evaluation (Callison-Burch, 2009; Denkowski and Lavie, 2010), summarization (Marge et al., 2010), sentiment detection (Mellebeek et al., 2010; Yano et al., 2010), and textual entailment (Wang and Callison-Burch, 2010; Negri and Mehdad, 2010). Burrows et al. (2012) use Mechanical Turk for paraphrase acquisition, a creative task in which workers are asked to paraphrase sentences. The authors point out that using Mechanical Turk allows to scale up to large datasets and that the many different workers increase diversity of the created data. This diversity is advantageous for creative tasks like paraphrasing but does not help with annotation tasks in which we want workers to adhere to common annotation guidelines.

A number of studies have looked at crowdsourcing for NER. Voyer et al. (2010) use a combination of expert and crowdsourced annotations. Yetisgen-Yildiz et al. (2010) use MTurk for annotating medical named entities. Researchers propose different annotation user interfaces for NER: Finin et al. (2010) annotate Twitter messages—short sequences of words—exploiting the short length by designing a vertically oriented annotation user interface. Lawson et al. (2010) choose an annotation interface where annotators have to drag the mouse to select entities, while Carpenter and Poesio (2010) argue that dragging is less convenient for workers than marking tokens.

These papers do not address AL in crowdsourcing. Another important difference is that previous studies on NER have used datasets for which no "linguistic" gold annotation is available. In contrast, we reannotated the CoNLL-2003 English NER dataset. This allowed us to conduct a detailed comparison of MTurk AL to conventional expert annotation.

Handling noisy labels The challenge of noisy annotations had been identified early on (Snow et al., 2008), and several approaches to mitigate noise in crowdsourcing scenarios have been proposed. Noisy annotations have also been considered from an active learning perspective (not necessarily using crowdsourcing). Although some of the latter approaches employ similar techniques, I will discuss them separately in the next section (6.8.2).

The leading crowdsourcing platform, Amazon Mechanical Turk, comes with the option of pre-screening workers using qualification questions and only allowing workers who pass this test to work on the actual task. However, this creates a barrier to entry that is unpopular among crowdsourcing workers. It also offers no incentive to continue giving good answers once workers have passed the test. Qualification questions also need to be manually designed. An alternative is injecting examples with known gold labels every now and then to check if workers try to provide good answers. This, however, requires a fairly large number of gold labeled examples. Oleson et al. (2011) therefore suggest creating synthetically labeled examples for the purpose of checking worker's answers.

Snow et al. (2008) collect multiple annotations for an example and then consolidate these into a single annotation by voting. They also assume that individual annotators have different biases which they try to estimate and then incorporate as weights into the voting procedure. Carpenter (2008), Raykar et al. (2010) and Ipeirotis et al. (2010) propose estimating worker biases and labels using expectation maximization. Kamar et al. (2012) and Sheng et al. (2008) propose approaches that control the number of repeated annotations for these kinds of models. However, as Oleson et al. (2011) point out, these models usually require the collection of a batch of annotations for each worker that is usually not possible in an active learning setup.

Chamberlain et al. (2008) propose splitting the annotation task into an annotation mode, where examples are annotated, and a validation mode, where workers double-check other workers' annotations. Similarly, for tasks requiring some creativity, Little et al. (2009) and Bernstein et al. (2010) propose a creative mode and a ballot mode, where workers vote to decide on a good answer. This requires designing both an annotation/creation and a validation/voting interface. It also requires to trade off the annotation budget between annotation and validation tasks. Dai et al. (2011) propose a decision-theoretic model to control this trade-off.

The fourth approach to obtaining consistent annotations is to offer incentives to workers who submit annotations that agree on the labels. Chamberlain et al. (2008) offer bonus points to players in a game-like annotation framework for agreeing coreference annotations, and Yetisgen-Yildiz et al. (2010) pay bonus payments to workers on MTurk for agreeing named entity spans. Shaw et al. (2011) compare different ways of designing incentives.

6.8.2 Active Learning with Noisy Labels

Hachey et al. (2005) were among the first to investigate the interaction of active sample selection on the annotation task and examined actively sampled instances on agreement of labels and annotation time. They demonstrate applicability of AL when annotators are trained experts. This is an important result. However, when trained experts are performing annotation, noise levels in the annotated data are lower than with crowdsourcing, so it is still necessary to investigate whether AL is robust under stronger noise.

Recently, the problem of AL performance with noisy labels has become a topic of interest in the AL community. Rehbein et al. (2010) investigate AL with human expert annotators for word sense disambiguation but do not find convincing evidence that AL reduces annotation cost in a realistic (non-simulated) annotation scenario.

Brew et al. (2010a) carried out experiments on sentiment active learning, using a small set of volunteer labelers instead of anonymous paid workers. This can be seen as "simulated crowdsourcing" with non-expert labelers. In a related paper (Brew et al., 2010b), they briefly sketch relabeling strategies that are similar to the adaptive voting strategies we propose in section 6.3.1. We believe that crowdsourcing on a real microtask market like Mechanical Turk poses additional challenges for the annotation task, such as additional noise introduced by negligent or "cheating" workers and the difficulty to foresee who will work on which examples. The only paper known to us that investigates AL for an NLP task in a real microtask market is Ambati et al. (2010), who investigate it using Mechanical Turk for creating training translation pairs for a machine translation system. They use a simple consensus schema for quality control. For an image recognition task, an AL approach for crowdsourcing using a similar relabeling strategy was also independently proposed by Zhao et al. (2011).

For a more theoretical perspective, Donmez and Carbonell (2008) propose a method to choose annotators from a set of noisy annotators. This assumes multiple different annotations; however, it also assumes that it is possible to

pick a specific annotator for a label query. This is usually not possible in crowdsourcing, as workers join and leave the site as they wish. Furthermore, they only evaluate their approach in simulations with a synthetic noise model, which may contain unrealistic assumptions when modeling annotators.

Du and Ling (2010) do not assume the ability to query individual annotators. Instead, they assume a uniform noise level for the labeling oracle and propose to combine uncertainty sampling AL with a relabeling strategy, querying more than one annotation per example. Their proposed relabeling strategy favors examples where the difference between class probabilities estimated by leave-one-out estimation on the other labeled examples and the empirical class probabilities determined from the (potentially multiple) labels of the example is maximal. Du and Ling (2010) show promising results on various non-NLP classification tasks from the UCI collection (Frank and Asuncion, 2010), but like Donmez and Carbonell (2008) they only use a synthetic noise model. Furthermore, the assumption of a uniform noise level is unrealistic when many different annotators are involved; our experiments show that noise levels of individual annotators vary significantly (see section 6.5). Brew et al. (2010b) also find that simple noise models are not adequate.

Paquet et al. (2010) propose to combine active learning with a Bayesian model of combining noisy labels. They use a special-purpose classifier that allows for the label combination of a fixed number of labels per example and that only does binary classification. They do not present experiments that show the performance of their model.

6.9 Summary

We have investigated the use of AL in a real-life annotation experiment with human annotators instead of traditional simulations with gold labels for named entity recognition and sentiment classification. The annotation was performed using the Amazon Mechanical Turk crowdsourcing platform in an AL framework that features concurrent example selection without wait times. We also

evaluated two strategies, adaptive voting and fragment recovery, to improve label quality at low additional cost. We find that even for the relatively high noise levels of annotations gathered with MTurk, AL is successful, improving performance by +6.9 points F_1 compared to random sampling for NER and by +2.6 points accuracy for sentiment. Furthermore, this performance level is reached at a smaller MTurk cost compared to random sampling. Thus AL not only reduces annotation costs but also offers an improvement in absolute performance for these tasks. This is clear evidence that active learning and crowdsourcing are complementary methods for lowering annotation cost and can be used together in training set creation for natural language processing tasks.

We have also conducted oracle experiments that show that further performance gains and cost savings can be achieved by using information about worker quality. We plan to confirm these results by using estimates of quality in the future.

Active learning is most useful for annotation tasks where the per-example cost is especially high, for example when experts have to do the annotation, so the budget only allows for small datasets. However, we believe that it can still be useful to combine AL with low-cost annotations. For many NLP tasks, good annotated datasets have sizes where significant annotation cost is required even with comparatively low cost per example. In these cases, active learning can therefore be used to make the most of an annotation budget even with low per-example annotation cost.

7 Conclusions

7.1 Contributions

Active learning is a strategy to reduce the amount of training examples that need to be annotated for training a supervised learning system by selecting examples that are maximally useful for the classifier. In Chapter 4, we applied active learning to the tasks of named entity recognition and coreference resolution.

Effective learning for NER In section 4.1, we showed that AL using uncertainty sampling is very effective for the NER task. Experiments on the ACE-2005 NER dataset showed that a subset of only 7% of the data selected by uncertainty sampling AL was sufficient to train a classifier that achieves the same performance as a baseline classifier trained on the entire dataset. Peak performance was reached on a subset of 12% of the data. At peak performance, the AL-trained classifier outperformed the classifier trained on the entire dataset by up to 2.5% F. As performance decreased again after passing the peak, we conclude it is important to monitor the learning progress and stop close to the peak. We proposed strategies for stopping in Chapter 5.

Avoiding the missed cluster effect Since active learning focuses on sampling examples from uncertain regions of the sample space, it may miss important regions of that sample space. This missed cluster effect can lead to slow learning of some classes. In section 4.2, we investigated the influence of example size on the missed cluster effect. We show that for NER, selecting whole sentences instead of individual tokens provides a natural way to allow some exploration

of the sample space by enabling co-selection of entity tokens that would not be in the focus of a pure AL setup. This makes the learning progress more robust against unfortunate initialization and avoids the missed cluster effect.

First successful AL for coreference resolution For coreference annotation, we found that standard uncertainty sampling AL does not outperform a random selection baseline. We identified two factors that contribute to this challenge: (i) for the decision tree classifiers used in our coreference system, uncertainty ratings are not reliable, and (ii) coreference annotation exhibits a severe imbalance of coreferent and disreferent pairs. Coreferent pairs are much rarer than disreferent pairs. This is again problematic because of the missed cluster effect.

We propose an active learning approach that addresses these factors by (i) using an explicit class-balancing strategy that focuses on links close to a mention (neighborhood pooling), (ii) sampling these links as a contiguous group (neighborhood sampling), again enabling co-selection, and (iii) selecting these groups using query-by-committee AL. Using this approach we could demonstrate the first successful application of active learning for coreference annotation.

Performance monitoring and early stopping We have seen that in using AL optimal performance can be reached using small subsets of the data. It is therefore important to monitor annotation and stop when optimal performance is reached. We investigated an approach to estimate classifier performance for entity classification using no additional labeled data. While probability estimates from an AL-trained classifier are too unreliable to directly estimate performance, we can use the gradient of the estimated performance for stopping the annotation process close to optimal performance. We also successfully used use the gradient of the uncertainty measure for the same goal. We demonstrated for the NER task that these gradient-based stopping criteria can reliably stop the AL loop at a point where performance exceeds the entire dataset baseline by up to 1.5% F, while using only 17% of the data.

Active learning using crowdsourcing Active learning offers annotation cost reductions by reducing the number of annotated examples. We combine this with crowdsourcing, the outsourcing of annotation tasks to a large group of workers via the Internet. Crowdsourcing offers low cost per annotation, so a combination with AL will offer further cost savings. At the same time, we validate the effectiveness of active learning with actual human annotation instead of simulated annotation. We designed a software for managing active learning annotation experiments on the Mechanical Turk crowdsourcing platform. This software offers concurrent retraining so annotators don't have to wait for selected examples.

Crowdsourced annotations contain more mistakes than gold-standard annotations. We therefore implemented an adaptive voting strategy to reconcile a good annotation from multiple noisy annotations. We demonstrate using two tasks, NER and sentiment classification, that active learning can be successfully applied with noisy crowdsourced labels when using our voting strategy. Further, we show that the active learning selection does select useful examples even when encountering noisy labels. Oracle experiments indicate the potential of further improved performance when taking into account the different quality of annotations from individual workers.

7.2 Outlook

We have seen that active learning is effective in reducing annotation effort. Despite impressive savings, however, active learning is only slowly being employed in annotation projects. Better availability of annotation frameworks that support active learning can help here. Web-based annotation platforms can relieve the complexity in developing and deploying a annotation software. We hope that the web-based active learning software presented in this thesis can help pave the way for an increased use of active learning in annotation projects.

Active learning probably delivers most value when it is used for data annotation for specialized domains, for example when targeting an information extraction system to a new extraction task or new text domain. In this case, several system components will need to be retargeted. So far, we have treated annotation for these tasks individually. We think it is interesting to investigate how these components can be trained together and how active learning strategies can select training data that is maximally useful for all components.

This is challenging because not only do we need to balance between examples that might be interesting but also because some components need the output of other components as input. If this output changes during the AL process, informative examples for subsequent components might lose their value. As an example, consider an end-to-end coreference system where the coreference resolver depends on good quality mention candidates. Therefore we believe that there are complex interactions of sample choice in systems with interdependent components and that sampling strategies designed for that purpose may be needed.

Bibliography

Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *ICML*, 1998.

Omar Alonso and Matthew Lease. Crowdsourcing for research & engineering. Tutorial at CrowdConf 2011, 2011. Slides online at http://www.slideshare.net/mattlease/crowdconf2011-tutorial-crowdsourcing-for-research-and-engineering; visited on April 4th, 2012.

Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Active learning and crowd-sourcing for machine translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1988.

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

Josh Attenberg and Foster Provost. Inactive learning? Difficulties employing active learning in practice. *SIGKDD Explorations Newsletter*, 12:36–41, 2011. URL http://doi.acm.org/10.1145/1964897.1964906.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *In The First International Conference on Language Resources and Evaluation Workshop on Linguistic Coreference*, pages 563–566, 1998.

Jason Baldridge and Miles Osborne. Active learning and logarithmic opinion pools for HPSG parse selection. *Natural Language Engineering*, 14(02): 191–222, 2008. doi: 10.1017/S1351324906004396. URL http://dx.doi.org/10.1017/S1351324906004396.

Jason Baldridge and Alexis Palmer. How well does active learning *actually* work? Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 296–305, 2009. URL http://www.aclweb.org/anthology/D/D09/D09-1031.

Yoram Baram, Ran El-Yaniv, and Kobi Luz. Online choice of active learning algorithms. In *ICML '03: Proceedings of the 20th International Conference on Machine Learning*, pages 19–26, 2003.

Yoram Baram, Ran El-Yaniv, and Kobi Luz. Online choice of active learning algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1005332.1005342.

Eric B. Baum and Kenneth Lang. Query learning can work poorly when a human oracle is used. In *Proceedings of the IEEE International Joint Conference in Neural Networks*, pages 335–340. IEEE Press, 1992.

Markus Becker, Ben Hachey, Beatrice Alex, and Claire Grover. Optimising Selective Sampling for Bootstrapping Named Entity Recognition. In *Proceedings of the ICML-2005 Workshop on Learning with Multiple Views*, 2005.

Adam L. Berger, Stephen Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5. doi: 10.1145/1866029.1866078. URL http://doi.acm.org/10.1145/1866029.1866078.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: a high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 194–201, Washington, DC, USA, 1997. Association for Computational Linguistics. doi: 10.3115/974557.974586. URL http://www.aclweb.org/anthology/A97-1029.

Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Michael Bloodgood and Chris Callison-Burch. Bucking the trend: Large-scale cost-focused active learning for statistical machine translation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 854–864, Uppsala, Sweden, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1088.

Michael Bloodgood and K. Vijay-Shanker. A method for stopping active learning based on stabilizing predictions and the need for user-adjustable stopping. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 39–47, Boulder, Colorado, 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1107.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. NYU: Description of the MENE Named Entity System as used in MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, volume 6. Citeseer, 1998.

Leo Breiman. *Classification and regression trees*. Chapman & Hall/CRC, 1984.

Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996. URL http://dx.doi.org/10.1007/BF00058655.

Anthony Brew, Derek Greene, and Pádraig Cunningham. Using crowdsourcing and active learning to track sentiment in online media. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 145–150, 2010a.

Anthony Brew, Derek Greene, and Pádraig Cunningham. The interaction between supervised learning and crowdsourcing. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, 2010b.

Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003)*, pages 59–66, 2003.

Steven Burrows, Martin Potthast, and Benno Stein. Paraphrase Acquisition via Crowdsourcing and Machine Learning. *Transactions on Intelligent Systems and Technology (ACM TIST) (to appear)*, 2012.

Chris Callison-Burch. Fast, cheap, and creative: evaluating translation quality using Amazon's Mechanical Turk. In *Proceedings of the 2009 Conference on*

Empirical Methods in Natural Language Processing, pages 286–295, 2009. URL http://www.aclweb.org/anthology/D/D09/D09-1030.

Colin Campbell, Nello Cristianini, and Alex J. Smola. Query learning with large margin classifiers. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, pages 111–118. Morgan Kaufmann, 2000.

Bob Carpenter. Multilevel bayesian models of categorical data annotation. Technical report, Lingpipe Inc., 2008.

Bob Carpenter and Massimo Poesio. Models of data annotation. Tutorial at the seventh international conference on Language Resources and Evaluation (LREC 2010), 2010.

Nicolas Cebron and Michael R. Berthold. Active learning for object classification: From exploration to exploitation. *Data Mining and Knowledge Discovery*, 18(2):283–299, 2009.

Jon Chamberlain, Massimo Poesio, and Udo Kruschwitz. Phrase detectives: A web-based collaborative annotation game. In *Proceedings of the International Conference on Semantic Systems (I-Semantics 08)*, 2008.

Stanley F. Chen and Ronald Rosenfeld. A gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Carnegie Mellon University, 1999.

Haibin Cheng, Ruofei Zhang, Yefei Peng, Jianchang Mao, and Pang-Ning Tan. Maximum margin active learning for sequence labeling with different length. In Petra Perner, editor, *Advances in Data Mining, Medical Applications, E-Commerce, Marketing, and Theoretical Aspects*, volume 5077 of *Lecture Notes in Computer Science*, pages 345–359. Springer Berlin/Heidelberg, 2008. ISBN 978-3-540-70717-2.

Nancy Chinchor. Overview of MUC-7/MET-2. In *Proceedings of the Message Understanding Conference MUC-7*, 1999.

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

Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602, Sydney, Australia, 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-1670.

Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

David A. Cohn. Neural network exploration using optimal experiment design. *Neural Networks*, 9(6):1071 – 1083, 1996. ISSN 0893-6080. doi: 10.1016/0893-6080(95)00137-9. URL http://www.sciencedirect.com/science/article/pii/0893608095001379.

David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

Nigel Collier and Jin-Dong Kim. Introduction to the bio-entity recognition task at jnlpba. In Nigel Collier, Patrick Ruch, and Adeline Nazarenko, editors, *COLING 2004 International Joint workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA/BioNLP) 2004*, pages 73–78, Geneva, Switzerland, 2004. COLING.

Michael Collins. Ranking algorithms for named entity extraction: Boosting and the voted perceptron. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 489–496, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073165. URL http://www.aclweb.org/anthology/P02-1062.

Aron Culotta and Andrew McCallum. Reducing labeling effort for stuctured prediction tasks. In *The Twentieth National Conference on Artificial Intelligence (AAAI)*, pages 746–751, Pittsburgh, PA, 2005. URL http://www.cs.umass.edu/~culotta/pubs/culotta05reducing.pdf.

Aron Culotta, Michael Wick, and Andrew McCallum. First-order probabilistic models for coreference resolution. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 81–88,

Rochester, New York, 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N07/N07-1011.

Peng Dai, Mausam, and Daniel S. Weld. Artificial intelligence for artificial artificial intelligence. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

Sanjoy Dasgupta and Daniel Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 208–215, 2008.

Hal Daumé III. Notes on CG and LM-BFGS optimization of logistic regression. Paper available at http://pub.hal3.name#daume04cg-bfgs, implementation available at http://hal3.name/megam/, 2004.

Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

Pascal Denis and Jason Baldridge. Global joint models for coreference resolution and named entity classification. In *Procesamiento del Lenguaje Natural*, volume 42, Barcelona, 2009. SEPLN.

Michael Denkowski and Alon Lavie. Exploring normalization techniques for human judgments of machine translation adequacy collected using amazon mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 57–61, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0709.

Christos Dimitrakakis and Christian Savu-Krohn. Cost-minimising strategies for data labelling: Optimal stopping and active learning. In *5th International Symposium on Foundations of Information and Knowledge Systems*, volume 4932 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2008.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The automatic content extraction (ACE) program–tasks, data, and evaluation. In *Proceedings of the fourth international conference on language resources and evaluation (LREC)*, volume 4, pages 837–840. Citeseer, 2004.

Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3):103–130, 1997.

Pinar Donmez and Jaime G. Carbonell. Proactive learning: cost-sensitive active learning with multiple imperfect oracles. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 619–628, 2008.

Pinar Donmez, Jaime G. Carbonell, and Paul Bennett. Dual strategy active learning. In Joost Kok, Jacek Koronacki, Raomon Mantaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 116–127. Springer Berlin/Heidelberg, 2007. ISBN 978-3-540-74957-8.

Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2009.

Jun Du and Charles X. Ling. Active learning with human-like noisy oracle. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM)*, pages 797–802, 2010. doi: 10.1109/ICDM.2010.114.

Kenneth Dwyer and Robert Holte. Decision tree instability and active learning. In *Machine Learning: ECML 2007, 18th European Conference on Machine Learning*, pages 128–139, 2007.

Sean P. Engelson and Ido Dagan. Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, pages 319–326, Santa Cruz, California, USA, 1996. Association for Computational Linguistics. doi: 10.3115/981863.981905. URL http://www.aclweb.org/anthology/P96-1042.

Seyda Ertekin, Jian Huang, Leon Bottou, and Lee Giles. Learning on the border: Active learning in imbalanced data classification. In *CIKM '07: Proceedings of the 16th ACM conference on Conference on information and knowledge management*, pages 127–136, 2007.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

Raul Fernandez and Bhuvana Ramabhadran. Exploiting active-learning strategies for annotating prosodic events with limited labeled data. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2208–2211, 2011. doi: 10.1109/ICASSP.2011.5946919.

Tim Finin, William Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. Annotating named entities in twitter data with crowdsourcing. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 80–88, 2010. URL http://www.aclweb.org/anthology/W10-0713.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 363–370, Ann Arbor, Michigan, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219885. URL http://www.aclweb.org/anthology/P05-1045.

A. Frank and A. Asuncion. UCI machine learning repository, 2010. URL http://archive.ics.uci.edu/ml.

Yoav Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995. ISSN 0890-5401. doi: 10.1006/inco.1995.1136. URL http://www.sciencedirect.com/science/article/pii/S0890540185711364.

Yoav Freund and Robert Schapire. A desicion-theoretic generalization of online learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer Berlin/Heidelberg, 1995. ISBN 978-3-540-59119-1.

Caroline Gasperin. Active learning for anaphora resolution. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 1–8, Boulder, Colorado, 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1901.

Alexander Genkin, David D. Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007. URL http://www.stat.rutgers.edu/~madigan/PAPERS/shortFat-v3a.pdf.

Masood Ghayoomi. Using variance as a stopping criterion for active learning of frame assignment. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 1–9, Los Angeles, California, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0101.

Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Query by committee made real. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *COLING*, 1996.

Yuhong Guo and Russ Greiner. Optimistic active learning using mutual information. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 7, pages 823–829, 2007. URL http://ijcai.org/Past%20Proceedings/IJCAI-2007/PDF/IJCAI07-132.pdf.

Ben Hachey, Beatrice Alex, and Markus Becker. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 144–151, Ann Arbor, Michigan, 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W05/W05-0619.

Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and McClanahan Peter. Assessing the costs of sampling methods in active learning for annotation. In *Proceedings of ACL-08: HLT, Short Papers*, pages 65–68, Columbus, Ohio, 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P/P08/P08-2017.

Robbie Haertel, Paul Felt, Eric K. Ringger, and Kevin Seppi. Parallel active learning: Eliminating wait time with minimal staleness. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 33–41, 2010. URL http://www.aclweb.org/anthology/W10-0105.

Gholamreza Haffari, Maxim Roy, and Anoop Sarkar. Active learning for statistical phrase-based machine translation. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 415–423, Boulder, Colorado, 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N09/N09-1047.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Alfonso Valencia. Overview of BioCreAtIvE: critical assessment of information extraction for biology. *BMC Bioinformatics*, 6(Suppl 1):S1, 2005. ISSN 1471-2105. doi: 10.1186/1471-2105-6-S1-S1.

Jerry Hobbs. Resolving pronoun references. In Barbara J. Grosz, Karen Sparck-Jones, and Bonnie Lynn Webber, editors, *Readings in natural language processing*, pages 339–352. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1986. ISBN 0-934613-11-7. URL http://dl.acm.org/citation.cfm?id=21922.24343.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, New York City, USA, 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N06/N06-2015.

Rong Hu, Brian MacNamee, and Sarah Jane Delany. Off to a good start: Using clustering to select the initial training set in active learning. In *Proceedings of the 23rd Florida Artificial Intelligence Research Society Conference (FLAIRS)*, 2010.

Panagiotis G Ipeirotis. The (Unofficial) NIST Definition of Crowdsourcing, 2012. Blog entry at http://www.behind-the-enemy-lines.com/2012/03/unofficial-nist-definition-of-cloud.html; visited on April 4th 2012.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation (HCOMP '10)*, 2010.

Prateek Jain, Sudheendra Vijayanarasimhan, and Kristen Grauman. Hashing hyperplane queries to near points with applications to large-scale active learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 928–936, 2010.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2nd edition, 2008.

Ece Kamar, Severin Hacker, and Eric Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *Proceedings of the 11th*

International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2012.

Jaeho Kang, Kwang Ryu, and Hyuk-Chul Kwon. Using cluster-based sampling to select initial training set for active learning in text classification. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining*, volume 3056 of *Lecture Notes in Computer Science*, pages 384–388. Springer Berlin/Heidelberg, 2004. ISBN 978-3-540-22064-0.

Hamidreza Kobdani and Hinrich Schütze. Sucre: A modular system for coreference resolution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 92–95, Uppsala, Sweden, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S10-1018.

Hamidreza Kobdani, Hinrich Schütze, Michael Schiehlen, and Hans Kamp. Bootstrapping coreference resolution using word associations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 783–792, Portland, Oregon, USA, 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P11-1079.

Seth Kulick, Ann Bies, Mark Liberman, Mark Mandel, Ryan T. McDonald, Martha S. Palmer, and Andrew Ian Schein. Integrated annotation for biomedical information extraction. In *Proceedings of the HLT-NAACL 2004 Workshop "Linking Biological Literature, Ontologies and Databases: Tools for Users"*, pages 61–68, 2004.

Christine Körner and Stefan Wrobel. Multi-class ensemble-based active learning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*, pages 687–694. Springer Berlin/Heidelberg, 2006. ISBN 978-3-540-45375-8.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01: Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

Florian Laws and Hinrich Schütze. Stopping criteria for active learning of named entity recognition. In *Proceedings of the 22nd International Confer-*

ence on Computational Linguistics (Coling 2008), pages 465–472, Manchester, UK, 2008. Coling 2008 Organizing Committee. URL http://www.aclweb.org/anthology/C08-1059.

Florian Laws, Christian Scheible, and Hinrich Schütze. Active learning with Amazon Mechanical Turk. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556, Edinburgh, Scotland, UK, 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D11-1143.

Florian Laws, Florian Heimerl, and Hinrich Schütze. Active learning for coreference resolution. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 508–512, Montréal, Canada, 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1055.

Nolan Lawson, Kevin Eustice, Mike Perkowitz, and Meliha Yetisgen-Yildiz. Annotating large email datasets for named entity recognition with mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 71–79, 2010. URL http://www.aclweb.org/anthology/W10-0712.

David D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 246–254, New York, NY, USA, 1995. ACM. ISBN 0-89791-714-6. doi: 10.1145/215206.215366. URL http://doi.acm.org/10.1145/215206.215366.

David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 3–12, New York, NY, USA, 1994. Springer-Verlag New York, Inc. ISBN 0-387-19889-X. URL http://dl.acm.org/citation.cfm?id=188490.188495.

Mingkun Li and Ishwar K. Sethi. Confidence-Based Active Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1251–1261, 2006.

Shoushan Li, Shengfeng Ju, Guodong Zhou, and Xiaojun Li. Active learning for imbalanced sentiment classification. In *Proceedings of the 2012*

Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 139–148, Jeju Island, Korea, 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D12-1013.

Greg Little, Lydia B. Chilton, Max Goldman, and Robert C. Miller. Turkit: tools for iterative tasks on mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, HCOMP '09, pages 29–30, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-672-4. doi: 10.1145/1600150.1600159. URL http://doi.acm.org/10.1145/1600150.1600159.

Xiaoqiang Luo. On coreference resolution performance metrics. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 25–32, Vancouver, British Columbia, Canada, 2005. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/H/H05/H05-1004.

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. A mention-synchronous coreference resolution algorithm based on the bell tree. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 135–142, Barcelona, Spain, 2004. doi: 10.3115/1218955.1218973. URL http://www.aclweb.org/anthology/P04-1018.

Gideon Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 109–112, Rochester, New York, 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N07/N07-2028.

Christopher Manning and Dan Klein. Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials—Volume 5*, 2003.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. URL http://nlp.stanford.edu/IR-book/.

Matthew Marge, Satanjeev Banerjee, and Alexander Rudnicky. Using the amazon mechanical turk to transcribe and annotate meeting speech for extractive summarization. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 99–107, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0716.

Diana Maynard, Valentin Tablan, Cristian Ursu, Hamish Cunningham, and Yorick Wilks. Named entity recognition from diverse text types. In *Recent Advances in Natural Language Processing Conference (RANLP)*, pages 257–274, 2001.

Andrew McCallum. MALLET: A machine learning for language toolkit, 2002. http://mallet.cs.umass.edu.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning (CoNLL) at HLT-NAACL 2003*, pages 188–191, 2003. URL http://www.aclweb.org/anthology/W03-0430.pdf.

Andrew McCallum and Kamal Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pages 350–358, 1998.

Bart Mellebeek, Francesc Benavent, Jens Grivolla, Joan Codina, Marta R. Costa-Jussà, and Rafael Banchs. Opinion mining of spanish customer comments with non-expert annotations on mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 114–121, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0718.

Timothy Miller, Dmitriy Dligach, and Guergana Savova. Active learning for coreference resolution. In *BioNLP: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, pages 73–81, Montréal, Canada, 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-2409.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. ISBN 978-0-07-042807-2.

John Mount. The equivalence of logistic regression and maximum entropy models, 2011. URL http://www.win-vector.com/dfiles/LogisticRegressionMaxEnt.pdf. visited on September 20th 2012.

MUC-6. MUC-6 coreference task definition, 1995. URL http://www.cs.nyu.edu/cs/faculty/grishman/COtask21.book_1.html.

Christoph Müller and Michael Strube. Multi-level annotation of linguistic data with MMAX2. In Sabine Braun, Kurt Kohn, and Joybrato Mukherjee, editors, *Corpus Technology and Language Pedagogy: New Resources, New Tools, New Methods*, pages 197–214. Peter Lang, Frankfurt a.M., Germany, 2006.

Ion Muslea, Steven Minton, and Craig A. Knoblock. Active learning with multiple views. *Journal of Artificial Intelligence Research*, 27(1):203–233, 2006.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007. ISSN 0378-4169.

Matteo Negri and Yashar Mehdad. Creating a bi-lingual entailment corpus through translations with mechanical turk: $100 for a 10-day rush. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 212–216, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0734.

Vincent Ng. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1396–1411, Uppsala, Sweden, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1142.

Vincent Ng and Claire Cardie. Improving machine learning approaches to coreference resolution. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073102. URL http://www.aclweb.org/anthology/P02-1014.

Vincent Ng and Claire Cardie. Bootstrapping coreference classifiers with multiple machine learning algorithms. In Michael Collins and Mark Steedman,

editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 113–120, 2003. URL http://www.aclweb.org/anthology/W03-1015.pdf.

Grace Ngai and David Yarowski. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *ACL*, 2000.

NIST. The ACE 2004 Evaluation Plan, 2004. URL http://www.itl.nist.gov/iad/mig/tests/ace/2004/.

NIST. The ACE 2008 Evaluation Plan. http://www.itl.nist.gov/iad/mig/tests/ace/2008/, 2008. URL http://www.itl.nist.gov/iad/mig/tests/ace/2008/.

Eric W. Noreen. *Computer-intensive methods for testing hypotheses: an introduction*. Wiley, 1989.

David Oleson, Alexander Sorokin, Greg Laughlin, Vaughn Hester, John Le, and Lukas Biewald. Programmatic gold: Targeted and scalable quality assurance in crowdsourcing. *Proceedings of the AAAI 2011 workshop on Human Computation (HCOMP)*, 2011.

Fredrik Olsson. On privacy preservation in text and document-based active learning for named entity recognition. In *Proceedings of the ACM first international workshop on privacy and anonymity for very large databases (PAVLAD '09)*, pages 53–60. ACM, 2009. doi: 10.1145/1651449.1651460. URL http://doi.acm.org/10.1145/1651449.1651460.

Fredrik Olsson and Katrin Tomanek. An intrinsic stopping criterion for committee-based active learning. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 138–146, Boulder, Colorado, 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1118.

Sebastian Padó. *User's guide to sigf: Significance testing by approximate randomisation*, 2006. URL http://www.nlpado.de/~sebastian/software/sigf.shtml.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics, 2002. doi: 10.3115/1118693.1118704. URL http://www.aclweb.org/anthology/W02-1011.

Ulrich Paquet, Jurgen Van Gael, David Stern, Gjergji Kasneci, Ralf Herbrich, and Thore Graepel. Vuvuzelas & active learning for online classification. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, 2010.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. CoNLL-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, 2011.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

Alexander J. Quinn and Benjamin B. Bederson. A taxonomy of distributed human computation. Technical Report HCIL-2009-23, University of Maryland, Human-Computer Interaction Lab, 2009.

Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, 2009. URL http://www.aclweb.org/anthology/W09-1119.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *J. Mach. Learn. Res.*, 99:1297–1322, 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1859890.1859894.

Marta Recasens and Eduard Hovy. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(04):485–510, 2011. doi: 10.1017/S135132491000029X. URL http://dx.doi.org/10.1017/S135132491000029X.

Marta Recasens, Lluís Màrquez, Emili Sapena, M. Antònia Martí, Mariona Taulé, Véronique Hoste, Massimo Poesio, and Yannick Versley. SemEval-2010 Task 1: Coreference Resolution in Multiple Languages. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 1–8, Uppsala, Sweden, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S10-1001.

Ines Rehbein, Josef Ruppenhofer, and Alexis Palmer. Bringing active learning to life. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 949–957, 2010. URL http://www.aclweb.org/anthology/C10-1107.

Feiliang Ren and Jingbo Zhu. An effective hybrid machine learning approach for coreference resolution. In *Proceedings of the Sixth SIGHAN Workshop on Chinese Language Processing*, 2008.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the Linguistic Annotation Workshop at ACL-2007*, pages 101–108, 2007.

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

Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 2003.

Manabu Sassano and Sadao Kurohashi. Using smaller constituents rather than sentences in active learning for japanese dependency parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 356–365, Uppsala, Sweden, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P10-1037.

Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In Johannes Fürnkranz and Thorsten Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 951–958, Haifa, Israel, 2010a. Omnipress. URL http://www.icml2010.org/papers/285.pdf.

Christoph Sawade, Niels Landwehr, and Tobias Scheffer. Active Estimation of F-Measures. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2083–2091. Curran Associates, Inc., 2010b.

Andrew Schein and Lyle Ungar. Active learning for logistic regression: An evaluation. *Machine Learning*, 68(3):235–265, 2007.

Helmut Schmid and Florian Laws. Estimation of conditional probabilities with decision trees and an application to fine-grained POS tagging. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 777–784, Manchester, UK, 2008. Coling 2008 Organizing Committee. URL http://www.aclweb.org/anthology/C08-1098.

Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pages 839–846. Morgan Kaufmann, 2000.

Hinrich Schütze, Emre Velipasaoglu, and Jan Pedersen. Performance thresholding in practical text classification. In *CIKM '06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 662–671, 2006.

Richard Segal, Ted Markowitz, and William Arnold. Fast uncertainty sampling for labeling large e-mail corpora. In *Conference on Email and Anti-Spam*, 2006.

Burr Settles. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012. doi: 10.2200/S00429ED1V01Y201207AIM018. URL http://www.morganclaypool.com/doi/abs/10.2200/S00429ED1V01Y201207AIM018.

Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, Honolulu, Hawaii, 2008. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D08-1112.

Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1069–1078, 2008a.

Burr Settles, Mark Craven, and Soumya Ray. Multiple-instance active learning. In *In Advances in Neural Information Processing Systems (NIPS)*, pages 1289–1296. MIT Press, 2008b.

H. Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on computational learning theory*, COLT '92, pages 287–294, New York, NY, USA, 1992. ACM. ISBN 0-89791-497-X. URL http://doi.acm.org/10.1145/130385.130417.

Aaron D. Shaw, John J. Horton, and Daniel L. Chen. Designing incentives for inexpert human raters. In *Proceedings of the ACM 2011 conference on Computer supported cooperative work*, CSCW '11, pages 275–284, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0556-3. doi: 10.1145/1958824.1958865. URL http://doi.acm.org/10.1145/1958824.1958865.

Dan Shen, Jie Zhang, Guodong Zhou, Jian Su, and Chew-Lim Tan. Effective adaptation of hidden markov model-based named entity recognizer for biomedical domain. In *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, pages 49–56, Sapporo, Japan, 2003. Association for Computational Linguistics. doi: 10.3115/1118958.1118965. URL http://www.aclweb.org/anthology/W03-1307.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou, and Chew-Lim Tan. Multi-criteria-based active learning for named entity recognition. In *ACL '04*, page 589, 2004. URL http://dx.doi.org/10.3115/1218955.1219030.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 614–622, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401965. URL http://doi.acm.org/10.1145/1401890.1401965.

Kevin Small and Dan Roth. Margin-based active learning for structured predictions. *International Journal of Machine Learning and Cybernetics*, 1: 3–25, 2010. URL http://dx.doi.org/10.1007/s13042-010-0003-y.

Larry Smith, Lorraine K. Tanabe, Rie J. Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M. Friedrich, Kuzman Ganchev, et al. Overview of BioCreative II gene mention recognition. *Genome Biology*, 9(Suppl 2):S2, 2008.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. Cheap and fast—but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 254–263, 2008. URL http://www.aclweb.org/anthology/D08-1027.

Wee Meng Soon, Daniel Chung Yong Lim, and Hwee Tou Ng. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4), 2001.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. In Lise Getoor and Ben Taskar, editors, *Introduction to statistical relational learning*, chapter 4. MIT Press, 2006.

Min Tang, Xiaoqiang Luo, and Salim Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL http://www.aclweb.org/anthology/W03-0419.pdf.

Katrin Tomanek. *Resource-Aware Annotation through Active Learning*. PhD thesis, Technical University of Dortmund, 2010.

Katrin Tomanek and Udo Hahn. Approximating learning curves for active-learning-driven annotation. In *Proceedings of the Sixth conference on International Language Resources and Evaluation (LREC)*. European Language Resources Association (ELRA), 2008.

Katrin Tomanek and Udo Hahn. Annotation time stamps – temporal metadata from the linguistic annotation process. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 486–495, Prague, Czech Republic, 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D/D07/D07-1051.

Katrin Tomanek, Florian Laws, Udo Hahn, and Hinrich Schütze. On proper unit selection in active learning: Co-selection effects for named entity recognition. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, pages 9–17, Boulder, Colorado, 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W09-1902.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

Kees van Deemter and Rodger Kibble. On coreferring: Coreference in muc and related annotation schemes. *Computational Linguistics*, 26:629–637, 1995.

Smita Vemulapalli, Xiaoqiang Luo, John F. Pitrelli, and Imed Zitouni. Classifier combination techniques applied to coreference resolution. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Student Research Workshop and Doctoral Consortium*, pages 1–6, Boulder, Colorado, 2009. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N/N09/N09-3001.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on message understanding*, MUC6 '95, pages 45–52, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics. ISBN 1-55860-402-2. doi: 10.3115/1072399.1072405. URL http://dx.doi.org/10.3115/1072399.1072405.

Andreas Vlachos. A stopping criterion for active learning. *Computer Speech and Language*, 22(3):295–312, 2008.

Luis von Ahn. Games with a purpose. *Computer*, 39(6):92 –94, 2006. ISSN 0018-9162. doi: 10.1109/MC.2006.196.

Robert Voyer, Valerie Nygaard, Will Fitzgerald, and Hannah Copperman. A hybrid model for annotating named entity training corpora. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 243–246, 2010. URL http://www.aclweb.org/anthology/W10-1839.

Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. ACE 2005 multilingual training corpus. *Linguistic Data Consortium, Philadelphia*, 2006.

Rui Wang and Chris Callison-Burch. Cheap facts and counter-facts. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 163–167, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0725.

Dittaya Wanvarie, Hiroya Takamura, and Manabu Okumura. Active learning for sequence labelling with probability re-estimation. In *Proceedings of the 11th Pacific Rim international conference on Trends in artificial intelligence*, PRICAI'10, pages 681–686, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15245-7, 978-3-642-15245-0. URL http://dl.acm.org/citation.cfm?id=1884293.1884366.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, 2005.

Tae Yano, Philip Resnik, and Noah A. Smith. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 152–158, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0723.

Meliha Yetisgen-Yildiz, Imre Solti, Fei Xia, and Scott Halgrim. Preliminary experiments with amazon's mechanical turk for annotating medical named entities. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 180–183, Los Angeles, 2010. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W10-0728.

Bianca Zadrozny, John Langford, and Naoki Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, Washington, DC, USA, 2003. IEEE Computer Society. doi: 10.1109/ICDM.2003.1250950. URL http://dl.acm.org/citation.cfm?id=951949.952181.

Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. Incremental relabeling for active learning with noisy crowdsourced annotations. In *Proceedings of the Third IEEE International Conference on Social Computing (SocialCom)*, 2011.

Jianhan Zhu, Victoria Uren, and Enrico Motta. Espotter: Adaptive named entity recognition for web browsing. In Klaus-Dieter Althoff, Andreas Dengel, Ralph Bergmann, Markus Nick, and Thomas Roth-Berghofer, editors, *Professional Knowledge Management*, volume 3782 of *Lecture Notes in Computer Science*, pages 518–529. Springer Berlin/Heidelberg, 2005. ISBN 978-

3-540-30465-4. doi: 10.1007/11590019_59. URL http://dx.doi.org/10.1007/11590019_59.

Jingbo Zhu and Eduard Hovy. Active learning for word sense disambiguation with methods for addressing the class imbalance problem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 783–790, 2007. URL http://www.aclweb.org/anthology/D/D07/D07-1082.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. Learning a stopping criterion for active learning for word sense disambiguation and text classification. In *Proceedings of the Third International Joint Conference on NLP (IJCNLP-2008)*, 2008a. URL http://www.isi.edu/natural-language/people/hovy/papers/08IJCNLP-zhu-hovy-stopping.pdf.

Jingbo Zhu, Huizhen Wang, and Eduard Hovy. Multi-criteria-based strategy to stop active learning for data annotation. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 1129–1136, Manchester, UK, 2008b. Coling 2008 Organizing Committee. URL http://www.aclweb.org/anthology/C08-1142.

Jingbo Zhu, Huizhen Wang, Eduard Hovy, and Matthew Ma. Confidence-based stopping criteria for active learning for data annotation. *ACM Transactions on Speech and Language Processing*, 6:3:1–3:24, 2010. URL http://doi.acm.org/10.1145/1753783.1753784.

Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semisupervised learning using gaussian fields and harmonic functions. In *Proceedings of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 58–65, 2003.