CURIOUS MACHINES:
ACTIVE LEARNING WITH STRUCTURED INSTANCES

by

Burr Settles

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2008

For my parents, who nurtured my curiosity,
and for Natalie, who now piques it.

ABSTRACT

Supervised machine learning is a branch of artificial intelligence concerned with automatically inducing predictive models from labeled data. Such learning approaches are useful for many interesting real-world applications, but particularly shine for tasks involving the automatic organization, extraction, and retrieval of information from large collections of data (e.g., text, images, and other digital media).

In traditional supervised learning, one uses "labeled" training data to induce a model. However, labeled instances for real-world applications are often difficult, expensive, or time consuming to obtain. Consider a complex task such as extracting key person and organization names from text documents. While gathering large amounts of unlabeled documents for these tasks is often relatively easy (e.g., from the World Wide Web), labeling these texts usually requires experienced human annotators with specific domain knowledge and training. There are implicit costs associated with obtaining these labels from domain experts, such as limited time and financial resources. This is especially true for applications that involve learning from instances with complex structures, which can require labels at varying levels of granularity.

Active learning addresses this inherent bottleneck by allowing the learner to selectively choose which parts of the available data are labeled for training. The goal is to maximize the accuracy of the learner through such "queries," while minimizing the work required of human annotators. In this thesis, I explore several important questions regarding active learning for these and similar tasks involving structured instances. What query strategies are available for these learning algorithms, and how do they compare? How might a learner pose queries at different levels of granularity, as with multiple-instance learning? Are there relationships between certain properties of a query and its difficulty for the annotator? If so, can these relationships be learned and exploited during active learning? The answers to the questions illustrate the utility and promise of active learning algorithms in complex real-world learning systems.

ACKNOWLEDGMENTS

This thesis would not exist without the help and guidance of many people. I am happy to take this opportunity to thank those who have influenced me during my graduate career.

Professionally, I am most indebted to my advisor, Mark Craven. Throughout my time working with him, he has provided an ideal balance of independence and direction regarding the work I present in these pages. He gave me the freedom to pursue my own research interests, but was always there to reign me in (when I got carried away) or help me get back on my feet (when I floundered). He has taught me a great deal about artificial intelligence, machine learning, and how to look at real problems—and real results—from an insatiably curious and scientific point of view. Most importantly, he has been a good friend and ally for the journey[1].

I also thank the members of my PhD committee for the investments they have made in my education and research. Jude Shavlik, David Page, and Xiaojin "Jerry" Zhu have introduced me to a variety of ways of thinking about and using machine learning in the real world. They were also always (well, usually) game for a good discussion whenever I popped in unannounced. Lew Friedland played an important role as well, through our useful conversations and his interest in collaborating on research involving the CKB data set in Chapter 6.

Further down the food chain, various students, post-docs, and visitors in the machine learning research group have made valuable contributions to my personal and professional life. I am specifically grateful to Soumya Ray and David Andrzejewski, who assisted me in annotating the SIVAL and Spec data sets, respectively, which are described in Chapter 6. Keith Noto made a great flat-mate during our time annexed at the University of Cambridge, sharing fabulous teas but bland British food. Mark Goadrich, Louis Oliphant, Michael Waddell, and Housam Nassif have been great office-mates over the years, always ready to scribble away on the whiteboards with me (whether it was relevant to research or not). Mark's fetish for German board games proved vital to my sanity at times as well. Thanks also to Bess Berg, Joe Bockhorst, Jesse Davis, Andrew Goldberg, Gautam Kunapuli, Rich Maclin, Michael Molla, Sriraam Natarajan, Irene Ong, Yue Pan, Beverly Seavey, Adam Smith, Ameet Soni, Lisa Torrey, Jurgen Van Gael and Trevor Walker for sharing in ideas and lunch breaks along the way.

Beyond the research group, I must thank Matt Anderson for his exquisite taste in Bourbon, marginal taste in pop culture, and encyclopedic knowledge of Macintosh computers. I am also pleased to have spent a portion of my time in Madison as part of its most celebrated all-graduate-student guitar, cello, and trumpet folk-rock trio. Brian and Sarah Wynia Smith were the other

[1]Mark may also be partially responsible for any physical fitness I possess. Before we met, I had never been skiing or run a marathon. Due to a knee injury from the former I have yet to do the latter, but he let me graduate anyway.

two thirds of that equation, and have been great friends and colleagues as well. A multitude of other people have come and gone through various departments (and apartments) over the years, too numerous to list here, but I appreciate the impact they have made to my life in the tundra.

Outside of Madison, Chinatsu Aone and Dmitry Zelenko created a stimulating research environment for me at SRA International during the summer of 2001. Likewise, Tom Rindflesch and Lan Aronson provided me with challenging research opportunities at the National Library of Medicine during the fall of 2003. These experiences not only put me in proximity to all of the amazing museums in our nation's capital, but also influenced my decision to pursue the PhD and to look more closely at ways of applying artificial intelligence to real-world problems. Rolling back the clock a little further, Dave Berque, my undergraduate advisor at DePauw, is who talked me into applying to graduate school in the first place (but I'm not one to hold a grudge). Thanks to Tom Warhover, Reuben Stern, and Brian Hamman at the *Columbia Missourian* and the Reynolds Journalism Institute at the University of Missouri, who helped coordinate annotation efforts for the CKB corpus described in Chapter 6. At the University of Cambridge Computer Laboratory, I thank Ted Briscoe, Niki Karamanis, Ian Lewin, Advaith Siddharthan, and Andreas Vlachos for making my time there enjoyable as well as productive. I have also been fortunate to meet and converse with many other helpful people at conferences over the years. In particular, thanks to Andrew McCallum, Aron Culotta, and Gregory Druck at UMass-Amherst for useful discussions regarding conditional random fields and active learning.

I rarely went hungry during my tenure as a graduate student. That is due in no small part to the fact that my research was financially supported, at various times, by the National Science Foundation, grant IIS-0093016, and the National Institutes of Health, grants R01-LM07050 and T15-LM07359.

Modulo one or two incidents as a teenager, my family has always been very supportive of my life choices. My stint in graduate school has been no different. In a way, the research path I have taken is a clear union of my mother's love for language and my father's love for biology and mathematics (culminating in my schizophrenic love for all of it). Even though my studies have taken me far away from them, my parents have been an unwavering source of support and encouragement, constantly assuring me that "of course" I can do this, even though most of the time they have no clue what the heck I do.

Finally, I am most personally indebted to my wife, lover, and best friend Natalie. Graduate school has been a difficult journey for both of us (and this thesis certainly look longer than expected), but it is because of her that I even did it at all. Every step of the way, she has made me laugh, made me food, and then made me go do the dishes. Now that the ink is dry on this volume in our journey together, I cannot wait to see what is in store for us next...

TABLE OF CONTENTS

Page

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

NOMENCLATURE

$\mathcal{L}$	Labeled data set
$\mathcal{U}$	Unabeled data set
x, y	Input data point and corresponding label
$\mathbf{x}, \mathbf{y}$	Input sequence and corresponding label sequence
$\mathcal{X}, y$	Input bag (multiple-instance representation) and bag label
θ	Parameters in a learned model
ℓ	Objective function for training a model (e.g., log-likelihood)
$\phi(\cdot)$	Query selection strategy (i.e., measure of informativeness)
$H(\cdot)$	Entropy
$D(\cdot\|\|\cdot)$	Kullback-Liebler (KL) divergence

Chapter 1

Introduction

We live in an age of information. Businesses, government organizations, scientific researchers, and individuals today are confronted with it in vast quantities: from news feeds and web sites to financial transactions to traffic updates (all largely in electronic form). Furthermore, things like email and instant messaging, address books, digital photos, and personal music and video libraries add to our growing individual data footprints. Managing all this data is becoming a real challenge. Disciplines like biology and medicine are rapidly transitioning into information sciences. Large-scale DNA sequencing (Blattner et al., 1997) and other high-throughput technologies such as SNP genotyping chips (Wang et al., 1998) and expression microarrays (Schena et al., 1995) allow researchers to easily obtain thousands of measurements of biological interest at once, but these measurements often lack any obvious interpretation. Such advances are prodding biomedical discovery toward information management tasks. Increasingly, these large collections of data are becoming critical to our everyday life and work. As digital computers become more tightly woven into the fabric of society, the proliferation of data will grow in both quantity and kind, as will the need to effectively organize, extract, retrieve, and even interpret the information buried therein.

Fortunately, the technological advances that have helped give rise to all this information have also created a fruitful environment for research in *machine learning*. Machine learning is the study of computer algorithms that automatically improve through experience. They improve by becoming better at explaining observations, making decisions, or predicting outcomes. For example, machines can interpret human speech by learning from vocal recordings that have been annotated for words and sentences (Tur et al., 2005). They can learn to drive a car after observing human driving behavior for a period of time (Urmson et al., 2008). They can even diagnose diseases by analyzing profiles of healthy vs. unhealthy patients (Mangasarian et al., 1995; Golub et al., 1999). Generally, the learning methods used for information management tasks fall into two groups:

- *Unsupervised learning*. The learning system is given a collection of "unlabeled" data. The goal is to organize aspects of the collection in some way. For example, clustering data points, called *instances*, into natural groups based on a set of observable features.

- *Supervised learning*. The learning system is given a collection of "labeled" instances, each denoted by the pair $\langle x, y \rangle$. The goal is to predict the label y for any new instance x, based on a set of features that describe it. When y is a real number, the task is called *regression*, when it is a set of discrete values, the task is called *classification*.

This thesis is concerned with applications that can be approached as supervised learning problems and, more specifically, as tasks that assign discrete labels to instances with complex input structures (more details in the next chapter). Algorithms have been developed for a host of interesting information management challenges. For example, models learned via supervised learning have been used to categorize nearly every type of digital media, from classifying newspaper articles (Lewis and Ringuette, 1994) and web pages (Craven et al., 1998), to identifying the contents of digital images (Chapelle et al., 1999), determining musical genre for audio recordings (Li et al., 2003), and even detecting objects and activities in video surveillance feeds (Hoiem et al., 2006). Supervised learning has also had much success in *information extraction*, such as recognizing person and organization names in newswire texts (Sang and DeMeulder, 2003), or identifying gene and protein mentions in biomedical journal articles (Settles, 2005), with the goal of mining interesting semantic information out of plain text. For many real-world applications, robust and scalable supervised machine learning algorithms have been developed and are in common use.

1.1 Active Learning

So what, then, is *active learning*? Active learning is the study of machine learning algorithms that ask questions. The key hypothesis underlying this approach is that, if the learning algorithm is allowed to choose the data from which it learns—to be "curious," if you will—it will perform better with less training. Why is this a desirable property for learning algorithms to have? Consider that, for any supervised learning system to perform well, it must often be trained on hundreds (even thousands) of labeled instances. Sometimes these labels come at little or no cost, such as the the "spam" flag you mark on unwanted email messages, or the five-star rating you might give to films on a social networking website. Learning systems use these flags and ratings to better filter your junk email and suggest movies you might enjoy. In these cases you provide such labels for free, but for many other more sophisticated supervised learning tasks, labeled instances are very difficult, time-consuming, or expensive to obtain. Here are a few examples:

- *Speech recognition*. Accurate labeling of speech utterances is extremely time consuming and requires trained linguists. Zhu (2005a) reports that annotation at the word level can take ten times longer than the actual audio (e.g., one minute of speech takes ten minutes to label), and annotating phonemes can take 400 times as long (e.g., nearly seven hours). The problem is compounded for rare languages or dialects, since expert annotators are equally rare.

- *Information extraction*. Good information extraction systems must be trained using labeled documents, and require detailed annotations. Users must highlight entities or relations of interest in text, such as person and organization names, or whether a person works for a particular organization. Locating such entities and relations can take half an hour for even simple newswire stories (as discussed in Chapter 6). Annotations for certain knowledge domains may require additional expertise, for example, annotating gene and disease mentions for biomedical information extraction often requires PhD-level biologists.

- *Classification and filtering*. Learning to classify documents (e.g., articles or web pages) or any other kind of media (e.g., audio and video files) requires that users label each document or media file with particular labels, like "relevant" or "not relevant." Having to annotate thousands of these instances can be tedious and even redundant. As with the previous examples, some applications may require hiring expensive annotators with rare, specialized training and expertise.

For many of these sorts of learning tasks, training instances take the form of complex input structures, e.g., sequences of words which must be individually labeled. These structured instances contribute in no small part to the high annotation costs associated with training models for these tasks. However, for these and similar applications, vast amounts of unlabeled data are readily available at low cost or even for free. In such cases, we can often take advantage of what the system has already learned by allowing it to ask questions about these vast and inexpensive unlabeled resources. In particular, if we allow an active learner to examine the unlabeled data and then *query* for the labels of instances it considers to be informative, we can make the most of the unlabeled data that is available. In effect, an active learner learns only what it needs to in order to improve, thus reducing the overall cost of training an accurate system.

1.2 Thesis Statement

This thesis aims to explore various key aspects of active learning for tasks that involve *structured* instances. The chapters that follow (i) describe machine learning approaches to various structured learning tasks, (ii) present the active learning scenarios and algorithms I have developed for these learning methods, and (iii) discuss how these approaches can mitigate the amount of work required to acquire labeled data in practice. Specifically, I focus on the following hypotheses:

i. Strategies that take into account how "representative" or "relevant" query instances are can produce more accurate systems with fewer labeled instances than strategies that do not.

ii. When querying instances with complex structures (e.g., labels on individual words in a sentence), strategies that consider the structured instance as a whole can perform better than strategies that aggregate individual label information.

iii. For some structured instances, labels can be acquired at multiple levels of granularity (e.g., documents and paragraphs). By selectively querying at these various granularities, particularly when one is easier to label than another, we can even further reduce annotation effort.

iv. Not all instances have equal annotation cost. To truly minimize the cost of acquiring labeled data, an active learning system should not only consider how informative each query is to the learner, but also take into account how expensive it will be for an annotator to label.

1.3 Outline

The remainder of this thesis is organized as follows:

- Chapter 2 presents an overview of supervised learning and active learning in general, as well as the particular problem settings that I consider in this work.
- Chapter 3 details several active learning strategies in the context of sequence labeling tasks, such as information extraction. I survey previously used query selection algorithms, and propose various novel algorithms to address their shortcomings.
- Chapter 4 further explores different characteristics and extensions of the information density algorithm, which I introduce in Chapter 3 of this thesis.
- Chapter 5 proposes multiple-instance active learning, a novel framework for acquiring and learning from labels at mixed levels of granularity. This setting can further reduce the annotation effort required for tasks such as text and image classification and retrieval.
- Chapter 6 investigates several key aspects of active learning with respect to minimizing real-world annotation costs, specifically elapsed annotation time.
- Chapter 7 provides a survey of the active learning literature.
- Chapter 8 describes some of my additional research in biomedical natural language processing, not directly related to active learning.
- Chapter 9 summarizes the key contributions of this work, discusses open problems and future directions in active learning, and offers some concluding remarks.

Chapter 2

Background

This chapter provides a brief introduction to supervised machine learning and active learning, as they pertain to the research in the chapters that follow.

2.1 Supervised Learning for Classification

In *supervised learning*, the machine is provided with labeled instances, each denoted by the pair $\langle x, y \rangle$. Its objective, then, is to determine how to properly assign a label y to each new instance x that it encounters in the future. For *classification* tasks, these labels come from a discrete set of known class descriptions. For example, suppose we want a system to scan through a collection of sports articles about baseball and hockey, and to automatically distinguish the two topics. This is an example of the simplest type of classification problem, binary classification, for which there are only two labels (e.g., baseball and hockey).

In general, the problem of supervised classification can be characterized as follows. The learner is given a *training set* of labeled instances, denoted $\mathcal{L} = \{\langle x, y \rangle^{(1)}, \ldots, \langle x, y \rangle^{(L)}\}$, for which the instances are described by some *input representation*. In some cases, each x is represented by a vector $\vec{x} \in \mathbb{R}^J$ composed of real-number measurements for each *feature* in the representation; this instance description is called a *feature vector*. Under this representation, instances can be thought of as points in a J-dimensional coordinate space (where J is the number of features), called a *feature space*. Finally, we provide the learner with a *training procedure* which it uses to induce a *model* (also called a *hypothesis*). A model here means a formal representation with an interpretation, under which instance inputs are mapped to label outputs. Generally, the model defines some classification function g on the instances: $y = g(x)$. Depending on what form a model can take, there may be an infinite number of them capable of explaining the observed data in $\mathcal{L}$. So the purpose of the training procedure is to find the "best" model from among the possibilities. The set of all possible models for a given problem defines the *model space* (or *hypothesis space*). The training procedure, then, searches through this space to find the optimal model according to some *objective function*.

2.1.1 Logistic Regression

For many supervised learning algorithms, the model can be characterized by a parameter vector $\theta \in \mathbb{R}^K$, which is used to parameterize the classification function g on instances. In this thesis, I primarily focus on probabilistic models of this form, such that $g(x) = \text{argmax}_y P(y|x;\theta)$. In other words, the predicted class label y is the one with the highest probability, conditioned on the instance x under model parameters θ. A simple example of this model type for binary classification is *logistic regression*, which defines the conditional probability that an instance is "positive" to be:

$$P(y = 1|x;\theta) = \frac{1}{1 + \exp(-\sum_{k=1}^{K} \theta_k f_k(x) + \theta_0)}.$$

Here, $y = 1$ denotes the positive class ($y = 0$ would denote the negative), and θ is an array of parameter weights corresponding to features in the input representation, each written $f_k(x)$. The special parameter θ_0 is called the *bias term*, which is a classification threshold of sorts. If the sum of the weighted features is greater than this bias term, then $P(y = 1|x;\theta) > 0.5$ and logistic regression will predict the positive class (otherwise it predicts negative).

For a properly trained logistic regression model, a learned weight θ_k should be positive for a feature f_k that is indicative of the positive class, negative for a feature indicative of the negative class, and near zero for a relatively uninformative feature. These values are determined by the training procedure, which we can think of as an optimization algorithm that searches through the model space for a good set of parameters. The "goodness" of a model's parameters is measured by the objective function, which evaluates how well a model fits the observations in the training set. A common objective function for logistic regression is the *log-likelihood* of the data:

$$\begin{aligned}\ell(\mathcal{L};\theta) &= \sum_{l=1}^{L} \log P(y^{(l)}|x^{(l)};\theta) \\ &= \sum_{l=1}^{L} \left(y^{(l)} \log o^{(l)} + (1 - y^{(l)}) \log(1 - o^{(l)})\right),\end{aligned}$$

where L is the size of the labeled training set $\mathcal{L}$, and ℓ is the conditional log-likelihood of the training set under the model θ. The output variable $o^{(l)}$ in the expanded form is shorthand for $P(y = 1|x^{(l)};\theta)$, or the conditional probability that $x^{(l)}$ is positive under the model. Since these equations represent smooth functions of the model parameters θ, we can estimate their values using gradient-based optimization methods (Nocedal and Wright, 1999). The partial derivative $\frac{\partial \ell}{\partial \theta_k}$, used in computing the training gradient $\nabla \ell(\mathcal{L};\theta)$, is given in the Appendix. Also note that the log-likelihood of a so-called "exponential" model like logistic regression is a concave function. As a result, there is a single set of parameters with the maximal value, and our training procedure is guaranteed to find the optimal solution with respect to the log-likelihood function[1].

[1] Another reason to use log-likelihood is that it transforms the likelihood of the data into a sum over the logs of labeled instance probabilities. With plain likelihood estimation, the objective function must compute the product of many probabilities, resulting in extremely small numbers that can be difficult to represent on computer hardware.

If we wish to make more complex label predictions, logistic regression can be easily generalized to more complex exponential models. For example, a common task is classifying objects into three or more label classes. One such approach is a multinomial logistic regression model, also known in the literature as a *maximum entropy* or *MaxEnt* model (Berger et al., 1996). This is a generalization of binary logistic regression that is able to make predictions for an arbitrary number of labels. MaxEnt models define the conditional probability of some label y given instance x to be:

$$P(y|x;\theta) = \frac{1}{Z(x)} \exp\left(\sum_{k=1}^{K} \theta_k f_k(y,x)\right),$$

where $Z(x) = \sum_i P(y_i|x;\theta)$ is a normalization factor over all possible class labels. Note that each feature $f_k(y,x)$ is now tied not only to the instance x, but also to the label y. This means that there can be many more parameters than input features (i.e., $K \gg J$), but it also allows us to use potentially different feature sets for each label. MaxEnt models are usually trained via gradient optimization as well, using a log-likelihood objective function:

$$\ell(\mathcal{L};\theta) = \sum_{l=1}^{L} \log P(y^{(l)}|x^{(l)};\theta) - \sum_{k=1}^{K} \frac{\theta_k^2}{2\sigma^2}.$$

Here, the second term is a Gaussian "regularization penalty" on $\|\theta\|$ to prevent over-fitting due to feature sparsity in $\mathcal{L}$. This regularizer essentially encourages the training procedure to prefer a set of parameters θ with many smaller weights over a θ with only a few very large weights. This regularization term is sometimes added to the objective function of binary logistic regression models as well. The partial derivative calculation used to compute this training gradient is given by Berger et al. (1996).

In this thesis, I conduct experiments using exponential methods that generalize logistic regression in similar ways. This family of algorithms can be used for many important supervised learning problems, such as labeling sequences (Chapters 3, 4, and 6), classification with more than two labels (Chapters 4 and 6), and multiple-instance learning (Chapters 5 and 6), as well as many other applications I do not consider here. Years of machine learning research have also demonstrated that these sorts of models are state-of-the-art for many important learning problems, in particular those that require large feature spaces. Moreover, for most of these model types, gradient-based training procedures are guaranteed to find the optimal set of parameters.

Note that there are many other types of supervised learning approaches. Some, like *naïve Bayes*, model the joint probability $P(x,y)$ of the labeled training instances, and use Bayes' rule to predict label probabilities[2]. Other approaches are non-probabilistic in nature, such as *decision trees* and *inductive logic programming*, and rely instead on classification rules in a subset of propositional or first-order logic (respectively). Still others, such as *nearest-neighbor* classification, maintain no model parameters at all and assign labels to instances based on the most similar labeled instances

[2]Learning algorithms that model the joint probability $P(x,y)$ and classify using Bayes' rule are called, by convention, *generative* approaches. Algorithms that model the conditional probability $P(y|x)$ directly (e.g., logistic regression) are called *discriminative* approaches. The approaches in this thesis are mainly discriminative.

in the training set. Each approach possesses its own inductive bias and unique attributes, which make it more or less well-suited to certain problem settings. For a good overview of supervised learning methods in general, I recommend Duda et al. (2001) or Mitchell (1997). While the active learning approaches I present in this work are described and evaluated in terms of exponential models, most of them are in no way limited to a particular learning algorithm. Many of them have analogs for these other learning approaches as well. I stress that *whenever* supervised learning is possible, active learning should also be possible.

2.1.2 Example: Text Classification

Now let us consider how we can apply logistic regression to a text classification task. Recall our example from before of classifying sports documents with one of two labels: baseball vs. hockey. This task comes from the 20 Newsgroups corpus (Lang, 1995), which is a collection of internet newsgroup posts from the early 1990s. Two of the newsgroup collections are `rec.sport.baseball` and `rec.sport.hockey`, consisting of about 1000 documents each.

To treat this as a learning task, we first need an input representation for text documents. A typical approach is the "bag of words" representation, which simply considers each word to be its own feature, and the value of each feature is the number of times the corresponding word appears in a given document. For example, if f_k is the feature PUCK, and the word "puck" occurs three times in document x, then $f_k(x) = 3$. Alternatively, if "puck" doesn't appear in the document at all, then $f_k(x) = 0$. Encoding each document in this way produces a feature vector of real values for each word in the input representation[3]. Next, we simply define class labels, e.g., let baseball be positive ($y = 1$) and hockey be negative ($y = 0$). This is all that is required to train a text classifier using logistic regression.

Table 2.1 shows a few sample word features and their trained logistic regression parameter weights for this example. Features with positive weights along the left-hand side indicate baseball documents, while the negative weights on the right indicate hockey documents. Not surprisingly, the words BASEBALL and HOCKEY are the features with the highest magnitude at the extreme ends of the parameter spectrum. Notice also that as weights approach zero (toward the bottom of either column), the features become more ambiguous, hence they are less informative. A logistic regression classifier achieves average accuracy = 0.985 for this task, estimated using ten-fold cross-validation (see Section 2.3 for more details on experimental methodology).

[3]Common words, called *stop-words* (e.g., "the" and "which"), are usually removed since they are often not useful for classification anyway. It is also sometimes common practice to use *stemming*, which strips off affixes to reveal the root word (e.g., "batting" and "punched" are reduced to their stems "bat" and "punch"). In this example, I do use stop-word filtering, but do not use stemming.

Table 2.1: Sample word features from the baseball vs. hockey example. Features are shown with their corresponding parameter weights from a trained logistic regression model.

Feature	Weight	Feature	Weight
BASEBALL	0.1131	HOCKEY	-0.1749
RUNS	0.0783	GOAL	-0.0625
BRAVES	0.0649	PERIOD	-0.0559
PITCHER	0.0516	PLAYOFF	-0.0486
BALL	0.0493	PUCK	-0.0394
HOME	0.0264	RANGERS	-0.0312
STRIKE	0.0213	CANADIAN	-0.0216
UMPIRE	0.0163	PENALTIES	-0.0204
WINS	0.0093	RESULTS	-0.0105
GAMES	0.0085	INJURIES	-0.0022
ROOM	0.0002	COMPUTER	-0.0005

2.2 Supervised Learning with Structured Instances

In the first chapter, I argued that many interesting supervised learning tasks involve learning from structured instances. In this section, I describe two such learning settings, *sequence labeling* and *multiple-instance learning*, and introduce the exponential models I use in each case.

2.2.1 Sequence Labeling and CRFs

Many real-world applications of machine learning, particularly in the areas of natural language processing (Manning and Schütze, 1999) and bioinformatics (Durbin et al., 1998), involve labeling and segmenting *sequences*. For example, we might wish to extract important organization names from a sentence (i.e., a sequence of words) or identify genes in DNA (i.e., a sequence of nucleic acids). Figure 2.1 illustrates how, for example, an information extraction problem can be viewed as a sequence labeling task. Let $\mathbf{x} = \langle x_1, \ldots, x_T \rangle$ be an observation sequence of length T with a corresponding label sequence $\mathbf{y} = \langle y_1, \ldots, y_T \rangle$. Words in a sentence correspond to *tokens* in the input sequence $\mathbf{x}$, which are mapped to labels in $\mathbf{y}$. Figure 2.1(a) presents an example $\langle \mathbf{x}, \mathbf{y} \rangle$ pair. The labels indicate whether a given word belongs to a particular entity class of interest (org and loc in this case, for "organization" and "location," respectively) or not (null).

Unlike the text classification example from before, each instance $\mathbf{x}$ in this setting is not represented by a single feature vector, but rather a structured sequence of feature vectors: one for each token (i.e., word). For example, the word "Madison" might be described by the features WORD=*Madison* and CAPITALIZED. However, it can variously correspond to the labels person ("The fourth U.S. President James Madison..."), loc ("The city of Madison, Wisconsin..."), and org ("Madison defeated St. Cloud in yesterday's hockey match..."). The appropriate label for a token

(a) (b)

Figure 2.1: An information extraction example viewed as a sequence labeling task. (a) A sample input sequence $\mathbf{x}$ and corresponding label sequence $\mathbf{y}$. (b) A sequence model represented as a finite state machine, illustrating the path of $\langle \mathbf{x}, \mathbf{y} \rangle$ through the model.

often depends on its context in the sequence. For sequence-labeling problems like information extraction, labels are typically predicted by a *sequence model* based on a probabilistic finite state machine, such as the one shown in Figure 2.1(b).

In this thesis, I focus on a family of sequence models known as *conditional random fields*, or CRFs (Lafferty et al., 2001). The rest of this section serves as a brief introduction. CRFs are undirected statistical graphical models which have demonstrated state-of-the-art accuracy on a wide variety of sequence labeling tasks, including information extraction (Settles, 2005; Sang and DeMeulder, 2003), document segmentation (Carvalho and Cohen, 2004), part-of-speech tagging (Lafferty et al., 2001), and shallow parsing (Sha and Pereira, 2003). In this work I use linear-chain CRFs, which correspond to conditionally trained probabilistic finite state machines.

A linear-chain CRF model with parameters θ defines the posterior probability of label sequence $\mathbf{y}$ given input sequence $\mathbf{x}$ to be[4]:

$$P(\mathbf{y}|\mathbf{x};\theta) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \theta_k f_k(y_{t-1}, y_t, \mathbf{x}_t)\right). \tag{2.1}$$

Here $Z(\mathbf{x})$ is a normalization factor over all possible labelings of $\mathbf{x}$, and θ_k is one of K model parameter weights corresponding to some feature $f_k(y_{t-1}, y_t, \mathbf{x}_t)$. Each feature f_k describes the sequence $\mathbf{x}$ at position t with label y_t, observed along a transition from label states y_{t-1} to y_t in the finite state machine. To clarify, consider the example from Figure 2.1 again. Here, f_k might be the feature WORD=*ACME* and have the value $f_k = 1$ along a transition from the null state to the org state (and 0 elsewhere). Other features set to 1 here might be ALLCAPS and NEXTWORD=*Inc*. The weights in θ are set to maximize the conditional log-likelihood ℓ of training sequences in the

[4]In this thesis I assume, without loss of generality, that each label is uniquely represented by one model state, thus each label sequence $\mathbf{y}$ corresponds to exactly one path through the model.

labeled training set $\mathcal{L}$:

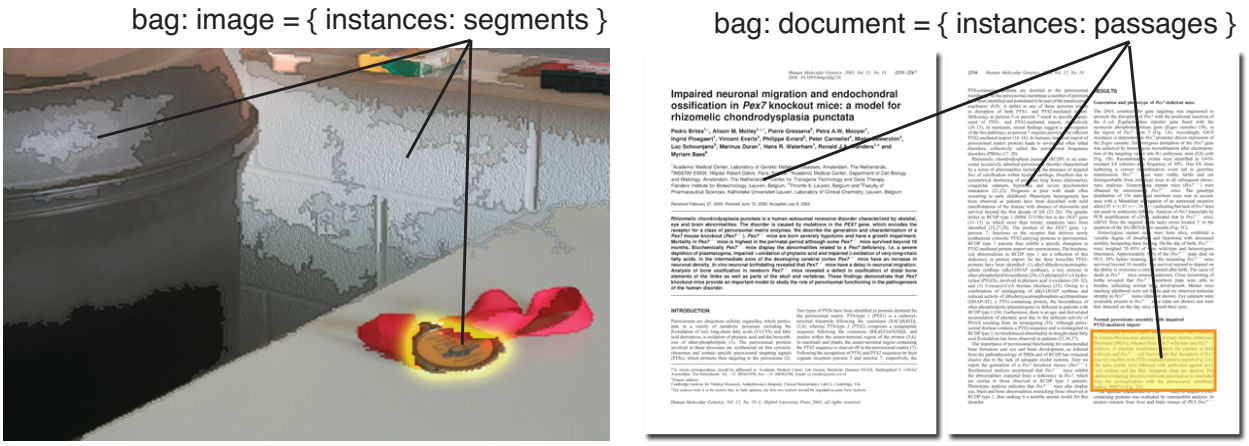
$$\ell(\mathcal{L};\theta) = \sum_{l=1}^{L} \log P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)};\theta) - \sum_{k=1}^{K} \frac{\theta_k^2}{2\sigma^2}, \tag{2.2}$$

where the second term is a Gaussian regularization penalty on $\|\theta\|$ to prevent over-fitting due to sparsity in $\mathcal{L}$. As with other exponential models in this family (e.g., logistic regression), the parameter weights θ can be learned using gradient-based methods, usually a quasi-Newton approach called L-BFGS (Liu and Nocedal, 1989). The partial derivative $\frac{\partial \ell}{\partial \theta_k}$, used in computing the training gradient $\nabla \ell(\mathcal{L};\theta)$, is given by Sutton and McCallum (2006). Once a CRF is trained, labels can be predicted for new sequences using the Viterbi algorithm (Rabiner, 1989). For more details on CRFs and their training or inference procedures, I recommend Sutton and McCallum (2006) for a good introduction.

2.2.2 Multiple-Instance Learning and MILR

A very different setting for learning with structured inputs is the *multiple-instance* (MI) learning framework. In the MI setting, instances are grouped into *bags* (i.e., multi-sets) which may contain any number of instances. More formally, let each bag $\mathcal{X}$ be composed of N constituent instances: $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$. A bag is labeled negative if and only if all of its instances x_n are negative. A bag is labeled positive, however, if at least one of its instances is positive. Note that positive bags may also contain negative instances. The MI setting was formalized by Dietterich et al. (1997) in the context of drug activity prediction, and has since been applied to a wide variety of tasks including content-based image retrieval (Maron and Lozano-Perez, 1998; Andrews et al., 2003; Rahmani and Goldman, 2006), text classification (Andrews et al., 2003; Ray and Craven, 2005), and protein family modeling (Tao et al., 2004).

Figure 2.2 illustrates how the MI representation can be applied to (a) content-based image retrieval (CBIR) and to (b) text classification. For the CBIR task, images are represented as bags and instances correspond to segmented regions of the image. A bag representing a given image is labeled positive if the image contains some object of interest. The multiple-instance paradigm is well suited to this task because only a few regions of an image may represent the object of interest, such as the gold medal in Figure 2.2(a). An advantage of the MI representation here is that it is significantly easier to label an entire image than it is to label each segment, or even a subset of the image segments. For the text classification task, documents can be represented as bags and instances correspond to short passages (e.g., paragraphs) that comprise each document. This formulation is useful in classification tasks for which document labels are freely available or cheaply obtained, but the target concept is represented by only a few passages. For example, consider the task of classifying biomedical journal articles according to whether or not they contain information about a particular protein, characterized by labels in the Gene Ontology (GO Consortium, 2004), such as its sub-cellular location. The article in Figure 2.2(b) is labeled by the Mouse Genome Database (Eppig et al., 2005) as a citation for the protein *catalase* that specifies such a "GO code." However, the text that actually states this relationship is only a short passage on the second page of the article. The MI approach is therefore compelling because document labels can be cheaply



(a) (b)

Figure 2.2: Applications of multiple-instance learning. (a) In content-based image retrieval, images are represented as bags and instances correspond to segmented image regions. (b) In text classification, documents are bags and instances represent passages of text. In each problem domain, an example positive instance is highlighted.

obtained (say from the Mouse Genome Database), but the labeling is not readily available at the most appropriate level of granularity (passages).

Each instance x_n can be represented by a feature vector (e.g., color and texture features in CBIR tasks, or word features in text classification tasks). However, a labeled bag $\mathcal{X}$ is actually a set of these instances (feature vectors), thus there is no direct relationship between features and labels. The inherent challenge in MI learning, therefore, is to induce an accurate model despite the ambiguity about which instances are positive, since they are obscured by the structure of the MI representation. One general approach is the Diverse Density framework (Maron and Lozano-Perez, 1998) for training probabilistic MI learning models. In my work, I train classifiers using *multiple-instance logistic regression* (MILR), a generalization of logistic regression in the Diverse Density framework that has been shown to be a state-of-the-art method for CBIR, text classification, and other MI learning tasks (Ray and Craven, 2005).

For MI classification, we seek $P(y = 1|\mathcal{X})$, the conditional probability that the label y is positive for a given bag $\mathcal{X}$. If a classifier can provide an analogous probability $P(y_n = 1|x_n)$ for each instance x_n, we can use a *combining function* (such as softmax or noisy-or) to combine the posterior probabilities for all instances in a bag and estimate its posterior probability $P(y = 1|\mathcal{X})$. The combining function here is what explicitly encodes the MI assumption. If the model finds an instance likely to be positive, the output of the combining function should find its corresponding bag likely to be positive as well. Figure 2.3 presents a diagram of a generic model in the Diverse Density framework.

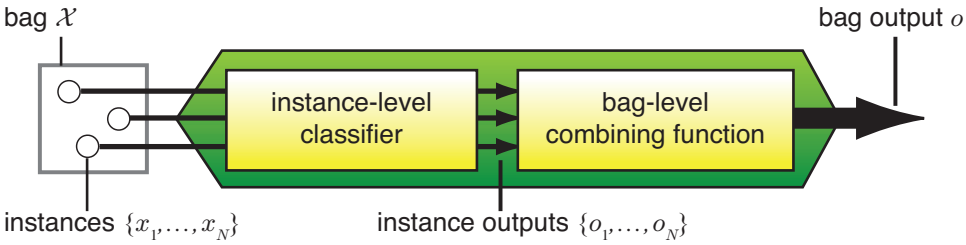


Figure 2.3: Diagram of a generic Diverse Density model for multiple-instance learning.

We can use logistic regression to estimate conditional probabilities for each instance:

$$o_n = P(y_n = 1|x_n;\theta) = \frac{1}{1+\exp(-\sum_{k=1}^{K}\theta_k f_k(x_n)+\theta_0)}. \tag{2.3}$$

Here the output o_n is shorthand for $P(y = 1|x_n;\theta)$, the conditional probability that instance x_n is positive under the model. Following convention, θ_0 is called the *bias term*, and acts as the classification threshold. In order to combine each output probability o_n for instances into an output probability o for the bag, I use the softmax function:

$$\begin{aligned} o = P(y = 1|\mathcal{X};\theta) &= \text{softmax}_\alpha(o_1, \ldots, o_N) \\ &= \frac{\sum_{n=1}^{N} o_n \exp(\alpha\, o_n)}{\sum_{n=1}^{N}\exp(\alpha\, o_n)}. \end{aligned} \tag{2.4}$$

The constant α determines the extent to which softmax behaves like a hard max function. It computes the mean when $\alpha = 0$, approximates the max function as $\alpha \to \infty$, and approximates the min function as $\alpha \to -\infty$.

In the general MI setting, we do not know the labels of instances in positive bags. However, because the equations above represent smooth functions of the model parameters θ, we can still learn parameter values using gradient-based optimization, such as maximizing the log-likelihood of training bags in the labeled set $\mathcal{L} = \{\langle\mathcal{X}, y\rangle^{(1)}, \ldots, \langle\mathcal{X}, y\rangle^{(L)}\}$:

$$\begin{aligned} \ell(\mathcal{L};\theta) &= \sum_{l=1}^{L}\log P(y^{(l)}|\mathcal{X}^{(l)};\theta) - \sum_{k=1}^{K}\frac{\theta_k^2}{2\sigma^2} \\ &= \sum_{l=1}^{L}\left(y^{(l)}\log o^{(l)} + (1-y^{(l)})\log(1-o^{(l)})\right) - \sum_{k=1}^{K}\frac{\theta_k^2}{2\sigma^2}. \end{aligned} \tag{2.5}$$

Here $y^{(l)} \in \{0, 1\}$ is the known label of bag $\mathcal{X}^{(l)}$, $o^{(l)}$ is the bag output probability under the model, and the second sum is a Gaussian regularization penalty on $\|\theta\|$ to prevent over-fitting (note that the bias weight θ_0 is not regularized). As with CRFs, I use L-BFGS (Liu and Nocedal, 1989) to optimize these parameters. Calculations for the partial derivative $\frac{\partial\ell}{\partial\theta_k}$, used in computing the training gradient $\nabla\ell$, are given in the Appendix.

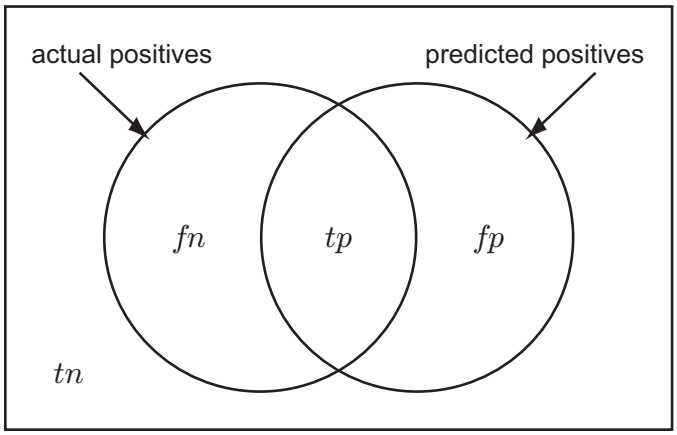


Figure 2.4: A Venn diagram illustrating the relationship between actual and predicted positives. The various overlaps define regions of tp (true positives), fp (false positives), tn (true negatives) and fn (false negatives).

Previous work with MILR has minimized the squared-loss objective function (Settles et al., 2008b; Ray and Craven, 2005). In my work for this thesis, however, I have found that optimizing log-likelihood yields more accurate results. Note that the log-likelihood of MILR (unlike standard supervised logistic regression) is non-convex due to the surface of the combining function. This means that gradient-based optimization is not guaranteed to converge to the globally optimal solution. Nevertheless, my work indicates that regularized log-likelihood usually behaves quite consistently, much more so than squared-loss. A detailed empirical comparison of both MILR formulations and well as the original Diverse Density approach (which couples a Gaussian instance model with a noisy-or combining function) are provided in the Appendix.

2.3 Evaluation Measures and Methodology

Once we have chosen our parameters θ, or *trained* the model, it is a good idea to evaluate how well it performs. To do this, we must use an evaluation measure of some sort, and this section outlines the various measures I use for my experiments in this thesis. For a particular label of interest, we are provided with a set of *actual positives* (e.g., objects that belong to that label) contained within the data set. The model then makes a set of *predicted positives* (e.g., the objects it assigns to that label) for the same data set. The actual and predicted label groupings can be thought of as indicator variables, and their cross product results in four important values: tp (the number of true positives), fp (false positives), tn (true negatives), and fn (false negatives). Figure 2.4 illustrates the relationship between these numbers.

A basic evaluation measure is *accuracy*[5] $= \frac{tp+tn}{tp+fp+tn+fn}$. Basically, this measure represents the fraction of objects that were labeled correctly by the model. In some problems, however, the data may be highly skewed, e.g., there might be nine times as many negative objects as positives. In a cases like this, accuracy is a poor evaluation measure because a model that labels everything negative will still have accuracy $= 0.9$. In these situations, it is common instead to use *precision*, $P = \frac{tp}{tp+fp}$, the fraction of predictions that are correct, and *recall*, $R = \frac{tp}{tp+fn}$, the fraction of actual positives that are correctly predicted. Because of the inherent trade-off between precision P and recall R, a summary statistic called the F_1 measure $= \frac{2 \times P \times R}{P+R}$ is commonly used when both are considered equally important. A final evaluation measure I consider is the area under the Receiver Operating Characteristic (ROC) curve. An ROC curve measures the rate of true positives vs. false positives as a threshold is varied across a measure of confidence in its predictions (e.g., the model's posterior probability of the target label). It is regarded as a more appropriate measure than accuracy for some machine learning applications (Provost et al., 1998). The area under the curve *AUROC*, also called the Wilcoxon signed-rank test, can be interpreted as the probability that the model will rank a randomly chosen positive object higher than a randomly chosen negative.

Since it is trivial for a model to do well on the labeled data $\mathcal{L}$ that was used to train it, we use the practice of randomly partitioning data into a *training set* and an *evaluation set*, which do not overlap. In this way, the model is properly evaluated on new instances it has never seen before. To account for the effects of randomized partitioning, it is common to repeat an experiment for several runs and average the results. One particular way of doing this is *cross-validation*. In five-fold cross-validation, for example, we split the data into five partitions or *folds*. Then we run five experiments for which each fold is held aside for evaluation, and the remaining four folds are used for training; then results are averaged across all folds.

2.4 Active Learning

Supervised learning models, such as the ones described in this chapter, have traditionally been trained on whatever labeled data is made available to them. As I have argued, *active learning* can reduce labeling effort required to train such models by allowing the learner to choose the instances from which it learns. This section provides an overview of the *pool-based* active learning setting, and how it is applied to the kinds of problems investigated in this work. Interested readers may skip ahead to Chapter 7 for an extended review of the active learning literature. It should be noted that active learning is a growing and changing field, so the survey is necessarily incomplete.

There are three general scenarios in which active learning is possible: (i) query instances may be synthesized by the learner de novo, (ii) instances are provided in a stream and the learner chooses to query or discard each one sequentially, or (iii) there exists a large pool $\mathcal{U}$ of unlabeled data which the learner may examine and select queries from. For many real-world tasks, synthesizing queries de novo can lead to instances that are unnatural or difficult for humans to interpret. For

[5] In this thesis, I sometimes use the word "accuracy" to mean the general performance of a model, and not any evaluation measure in particular. However, when reporting empirical results (e.g., figures in a table or a plot), accuracy always refers to the measure defined here.

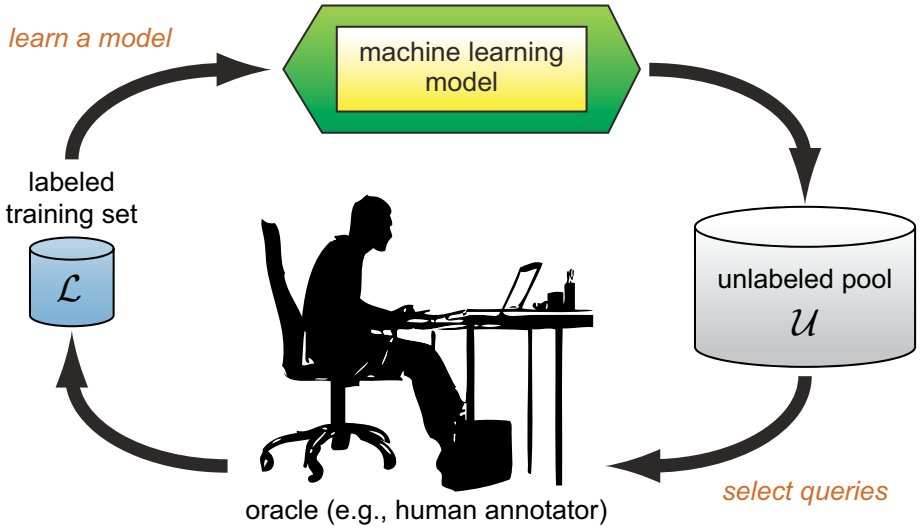


Figure 2.5: The pool-based active learning cycle.

example, Baum and Lang (1992) found that a model learning to recognize handwritten characters generated query images that were not real characters at all, but artificial combinations of existing letters and digits. Therefore, the stream-based and pool-based scenarios are often more realistic. In this thesis, I focus on the pool-based setting, since large repositories of unlabeled texts, images, and the like are usually available for these sorts of problems.

2.4.1 Pool-Based Active Learning and Query Selection

Figure 2.5 illustrates the pool-based *active learning cycle*. An active learner may begin with a small number of labeled instances in the labeled training set $\mathcal{L}$, request labels for a few carefully selected instances from the unlabeled pool $\mathcal{U}$, learn from the query results, and then leverage its newly-found knowledge to choose which instances to query next. In this way, the active learner aims to achieve high accuracy using as few labeled instances as possible. There are many ways to select query instances, most of which stem from the uncertainty principle in experimental design and statistics (Federov, 1972). One strategy for pool-based active learning, *uncertainty sampling* (Lewis and Gale, 1994), queries the instance that the model is least certain how to label. For probabilistic binary classifiers, this means querying the instance $x \in \mathcal{U}$ with the posterior probability $P(y = 1|x; \theta)$ that is closest to 0.5 (i.e., the most ambiguous instance).

Figure 2.6 illustrates the potential of active learning in a way that is easy to visualize. First, I created a toy data set from two Gaussians centered at (-2,0) and (2,0) with standard deviation $\sigma = 1$, each representing a different class distribution. Figure 2.6(a) shows the resulting data set after 400 instances are sampled (200 from each class); instances are represented as points in a 2D feature space. In a real-world setting these instances may be available, but their labels usually are not. Figure 2.6(b) illustrates the traditional supervised learning approach after randomly selecting

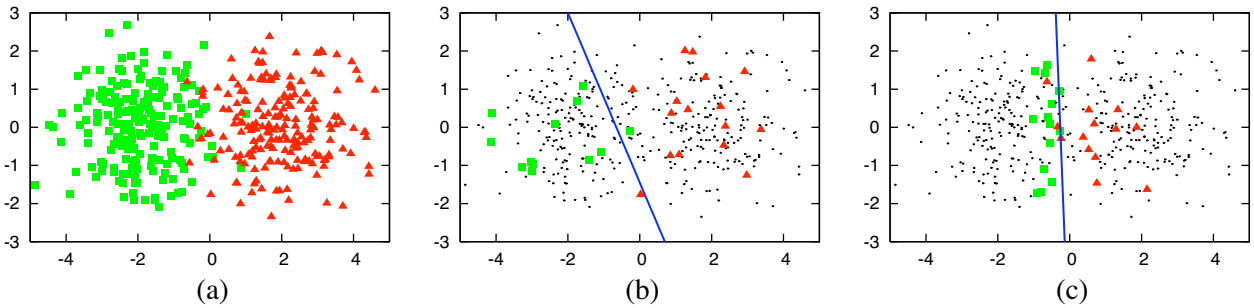


Figure 2.6: An illustrative example of pool-based active learning. (a) A toy data set of 400 instances, evenly sampled from two class Gaussians. The instances are represented as points in a 2D feature space. (b) A logistic regression model trained with 30 labeled instances randomly drawn from the pool. The line represents the decision boundary of the classifier (accuracy = 0.7). (c) A logistic regression model trained with 30 actively queried instances using uncertainty sampling (accuracy = 0.9).

30 instances for labeling, drawn i.i.d. from the problem domain. The line shows the linear decision boundary of a logistic regression model (i.e., where the posterior equals 0.5) trained using these 30 points. Notice that most of the labeled instances in this training set are far from zero on the horizontal axis, which is where the decision boundary should probably be. As a result, this classifier only achieves accuracy = 0.7 on the remaining unlabeled points. Figure 2.6(c), however, tells a very different story. The active learner uses uncertainty sampling to focus on instances closest to its decision boundary, assuming it can adequately explain instances in other parts of the instance space in $\mathcal{U}$. As a result, it avoids requesting labels for redundant or irrelevant instances, and achieves accuracy = 0.9 with a mere 30 labeled instances. That is a 67% reduction in error compared to standard supervised learning, and less than 10% of the data was labeled.

2.4.2 Example: Active Text Classification

Now let us apply the same principles of uncertainty sampling to the baseball vs. hockey example from before. In the supervised learning experiment from Section 2.1.2, we used ten-fold cross-validation. Thus 10% of the *corpus*[6] was held aside for evaluation for each fold, while the remaining 90% was used for training. In an active learning experiment, we still hold 10% aside for evaluation, but only a few of the remaining instances are placed in the labeled set $\mathcal{L}$, while all others are placed in the unlabeled pool $\mathcal{U}$. The active learner then queries instances from $\mathcal{U}$ according to the iterative active learning cycle shown in Figure 2.5. In this particular experiment, $\mathcal{L}$ is initialized with five random instances each fold.

Active learning algorithms are generally evaluated by constructing *learning curves*. These curves plot the evaluation measure of interest as a function of the number of new instances that are

[6] In text domains, a data set is usually called a corpus.

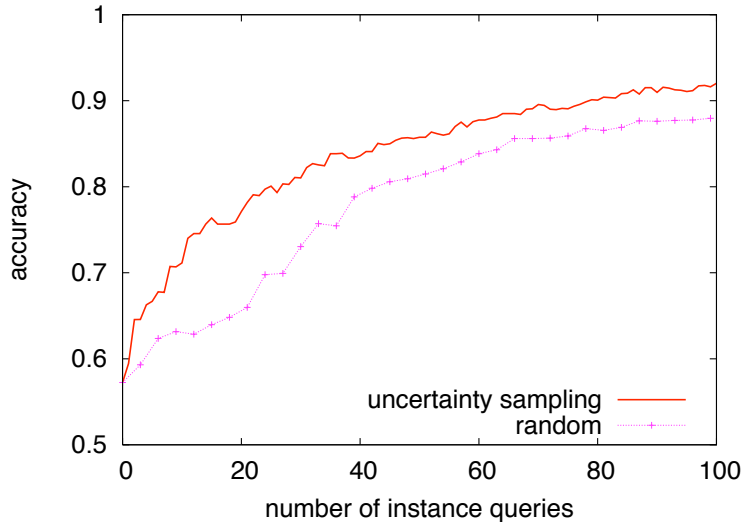


Figure 2.7: Learning curves for the baseball vs. hockey example. Curves plot the evaluation measure (e.g., accuracy) as a function of the number of new instances labeled for two selection strategies: uncertainty sampling (active learning) and random sampling (passive learning). We can see that the active learning approach is superior here because its learning curve dominates that of random sampling.

labeled and added to $\mathcal{L}$. Figure 2.7 shows learning curves for the first 100 instances labeled using uncertainty sampling vs. random sampling, i.e., traditional "passive" supervised learning in which instances are drawn i.i.d. from the unlabeled pool. After labeling 30 new instances, the accuracy of active learning is 0.810, while the supervised learning baseline is only 0.730. As can be seen, the active learning curve dominates the random baseline curve for all of the points shown in this figure. We can conclude that an active learning algorithm is superior to some other approach (e.g., a random baseline) if it dominates the other for most or all of the points along their learning curves.

2.5 Summary

This chapter has provided an introduction to the principles of supervised learning, experimental evaluation, and active learning that serve as a foundation for the rest of this document. In the chapters that follow, I explore various query selection strategies for the more complex, structured learning models I introduced in Section 2.2. I also consider cases in which the input structure itself can be exploited by actively acquiring labels at mixed levels of granularity, and how one might reduce not only the size of the training set $\mathcal{L}$, but minimize real-world annotation costs as well.

Chapter 3

Active Learning for Sequence Labeling

This chapter investigates active learning approaches to *sequence labeling*, which is a fundamental task for many real-world applications, such as in information extraction and text segmentation. I survey the methods that have been proposed for sequence models to date, motivate and present several novel algorithms to address their shortcomings, and conduct a large-scale empirical analysis to compare their performance on multiple benchmark data sets. This work has been previously published (Settles and Craven, 2008).

3.1 Introduction

Active learning is appropriate for problem domains where unlabeled data are readily available, but obtaining training labels is expensive. Such is the case with many *sequence labeling* tasks in natural language processing. For example, part-of-speech tagging (Lafferty et al., 2001; Seung et al., 1992), information extraction (Settles, 2005; Sang and DeMeulder, 2003), and document segmentation (Carvalho and Cohen, 2004) are all typically treated as sequence labeling problems. The source data for these tasks (i.e., text documents in electronic form) are often easily and inexpensively obtained. However, due to the structured nature of sequence labeling tasks, annotating these texts can be rather tedious and time-consuming, making active learning an attractive technique for building information management systems that involve these tasks.

While there has been much work on active learning for classification (Zhang and Oles, 2000; McCallum and Nigam, 1998b; Cohn et al., 1994), active learning for sequence labeling has received considerably less attention. A handful methods have been proposed, based mostly on the conventions of *uncertainty sampling*, where the learner queries the instance about which it has the least certainty, and one using *query-by-committee*; where a "committee" of models selects the instance about which its members most disagree. I describe these previously proposed methods, as well as several novel approaches, in the next section.

The comparative effectiveness of the few previously published approaches, however, has not been studied. Furthermore, it has been suggested that uncertainty sampling and query-by-committee fail on occasion for classification (Zhu et al., 2003; Roy and McCallum, 2001) by querying outliers, e.g., instances that are considered informative in isolation by the learner, but contain little information about the *rest* of the distribution of instances. Proposed methods for dealing with these shortcomings have so far only considered classification tasks.

Given: Labeled set $\mathcal{L}$, unlabeled pool $\mathcal{U}$, query strategy $\phi(\cdot)$, query batch size B

repeat
  // learn a model using the current $\mathcal{L}$
  $\theta = \text{train}(\mathcal{L})$;
  for $b = 1$ **to** B **do**
    // query the most informative instance
    $\mathbf{x}_b^* = \text{argmax}_{\mathbf{x} \in \mathcal{U}} \, \phi(\mathbf{x})$;
    $\mathbf{y}_b = \text{label}(\mathbf{x}_b^*)$;
    // move the labeled query from $\mathcal{U}$ *to* $\mathcal{L}$
    $\mathcal{L} = \mathcal{L} \cup \langle \mathbf{x}_b^*, \mathbf{y}_b \rangle$;
    $\mathcal{U} = \mathcal{U} - \mathbf{x}_b^*$;
  end
until *some stopping criterion* ;

Table 3.1: A generic pool-based active learning algorithm.

This chapter presents two major advances in active learning research for sequence labeling tasks. First, I motivate and introduce several new query strategies for probabilistic sequence models. Second, I conduct a thorough empirical analysis of previously proposed methods along with my algorithms on a variety of benchmark corpora. The remainder of this chapter is organized as follows. Section 3.2 describes in detail all the query selection strategies designed for sequence models that I consider. Section 3.4 presents the results of the empirical study, and suggests how these algorithms might best be used in practice. Section 3.5 concludes with a summary of findings and suggests future work.

3.2 Active Learning for Sequence Models

In order to select queries, an active learner must have a way of assessing how *informative* each instance is. Let $\mathbf{x}^*$ be the most informative instance according to some query strategy $\phi(\mathbf{x})$, which is a function used to evaluate each instance $\mathbf{x}$ in the unlabeled pool $\mathcal{U}$. The algorithm in Table 3.1 provides a sketch of a generic pool-based active learning approach. In the remainder of this section, I describe various query strategy formulations of $\phi(\cdot)$ that have been used for active learning with sequence models. I also point out where I think these approaches may be flawed, and propose several novel query strategies to address these issues.

Note that, while I describe these active learning algorithms in terms of linear-chain conditional random fields, or CRFs (see Section 2.2.1), they have analogs for other kinds of commonly used sequence models such as hidden Markov models, or HMMs (Rabiner, 1989), probabilistic context-free grammars (Lari and Young, 1990), and general CRFs (Sutton and McCallum, 2006).

3.2.1 Uncertainty Sampling

One of the most common general frameworks for measuring informativeness is *uncertainty sampling* (Lewis and Gale, 1994), where a learner queries the instance that it is most uncertain how to label. Culotta and McCallum (2005) employ a simple uncertainty-based strategy for sequence models called **least confidence (LC)**:

$$\phi^{LC}(\mathbf{x}) = 1 - P(\mathbf{y}^*|\mathbf{x};\theta).$$

Here, $\mathbf{y}^*$ is the most likely label sequence, i.e., the Viterbi parse. This approach queries the instance for which the current model has the least confidence in its most likely labeling. For CRFs, this confidence can be calculated using the posterior probability given by Equation 2.1.

Scheffer et al. (2001) propose an alternative uncertainty strategy, which queries the instance with the smallest margin between the posteriors for its two most likely labelings. I call this approach **margin (M)**:

$$\phi^{M}(\mathbf{x}) = -\big(P(\mathbf{y}_1^*|\mathbf{x};\theta) - P(\mathbf{y}_2^*|\mathbf{x};\theta)\big).$$

Here, $\mathbf{y}_1^*$ and $\mathbf{y}_2^*$ are the first and second most likely label sequences, respectively. These can be efficiently computed using the N-best algorithm (Schwartz and Chow, 1990), a beam-search generalization of Viterbi, with $N = 2$. The minus sign in front is simply to ensure that ϕ^M acts as a maximizer for use with the algorithm in Table 3.1.

Another uncertainty-based measure of informativeness is *entropy* (Shannon, 1948). For a discrete random variable Y, the entropy is given by $H(Y) = -\sum_i P(y_i) \log P(y_i)$, and represents the information needed to "encode" the distribution of outcomes for Y. As such, is it often thought of as a measure of uncertainty in machine learning. In active learning, we wish to use the entropy of our model's posteriors over its labelings. One way this has been done with probabilistic sequence models is by computing what I call **token entropy (TE)**:

$$\phi^{TE}(\mathbf{x}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} P_\theta(y_t = m) \log P_\theta(y_t = m), \tag{3.1}$$

where T is the length of $\mathbf{x}$, m ranges over all possible token labels, and $P_\theta(y_t = m)$ is shorthand for the marginal probability that m is the label at position t in the sequence, according to the model. For CRFs and HMMs, these marginals can be efficiently computed using the *forward* and *backward* algorithms (Rabiner, 1989). The summed token entropies have typically been normalized by sequence length T, to avoid simply querying longer sequences (Baldridge and Osborne, 2004; Hwa, 2004). However, I argue that querying long sequences should not be explicitly discouraged, if in fact they contain more information. Thus, I also propose the **total token entropy (TTE)** measure:

$$\phi^{TTE}(\mathbf{x}) = T \times \phi^{TE}(\mathbf{x}).$$

For most sequence labeling tasks, however, it is more appropriate to consider the entropy of the label sequence $\mathbf{y}$ as a whole, rather than some aggregate of individual token entropies. Thus an

alternate query strategy is **sequence entropy (SE)**:

$$\phi^{SE}(\mathbf{x}) = -\sum_{\hat{\mathbf{y}}} P(\hat{\mathbf{y}}|\mathbf{x};\theta) \log P(\hat{\mathbf{y}}|\mathbf{x};\theta), \tag{3.2}$$

where $\hat{\mathbf{y}}$ ranges over all possible label sequences for input sequence $\mathbf{x}$. Note, however, that the number of possible labelings grows exponentially with the length of $\mathbf{x}$. To make this feasible, previous work (Kim et al., 2006) has employed an approximation I call ***N*-best sequence entropy (NSE)**:

$$\phi^{NSE}(\mathbf{x}) = -\sum_{\hat{\mathbf{y}} \in \mathcal{N}} P(\hat{\mathbf{y}}|\mathbf{x};\theta) \log P(\hat{\mathbf{y}}|\mathbf{x};\theta),$$

where $\mathcal{N} = \{\mathbf{y}_1^*, \ldots, \mathbf{y}_N^*\}$, the set of the N most likely parses, and the posteriors are re-normalized (i.e., $Z(\mathbf{x})$ in Equation 2.1 only ranges over $\mathcal{N}$). For $N = 2$, this approximation is equivalent to ϕ^M, thus N-best sequence entropy can be thought of as a generalization of the margin approach.

Recently, an efficient entropy calculation via dynamic programming was proposed for CRFs in the context of semi-supervised learning (Mann and McCallum, 2007b). I use this algorithm to compute the true sequence entropy (3.2) for active learning in a constant-time factor of Viterbi's complexity. Hwa (2004) employed a similar approach for active learning with probabilistic context-free grammars.

3.2.2 Query-By-Committee

Another general active learning framework is the *query-by-committee* (QBC) approach (Seung et al., 1992). In this setting, one uses a committee of models $\mathcal{C} = \{\theta^{(1)}, \ldots, \theta^{(C)}\}$ to represent C different hypotheses that are consistent with the labeled set $\mathcal{L}$. The most informative query, then, is the instance over which the committee is in most disagreement about how to label. In particular, I employ the *query-by-bagging* approach (Abe and Mamitsuka, 1998) to learn a committee of CRFs. In each round of active learning, $\mathcal{L}$ is sampled (with replacement) L times to create a unique, modified labeled set $\mathcal{L}^{(c)}$. Each model $\theta^{(c)} \in \mathcal{C}$ is then trained using its own corresponding labeled set $\mathcal{L}^{(c)}$. To measure disagreement among committee members, I consider two alternatives.

Dagan and Engelson (1995) introduced QBC with HMMs for part-of-speech tagging using a measure called **vote entropy (VE)**:

$$\phi^{VE}(\mathbf{x}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M}\frac{V(y_t,m)}{C}\log\frac{V(y_t,m)}{C},$$

where $V(y_t, m)$ is the number of "votes" label m receives from all the committee member's Viterbi labelings at sequence position t.

McCallum and Nigam (1998b) propose a QBC strategy for classification based on *Kullback-Leibler (KL) divergence* (Kullback and Leibler, 1951), an information-theoretic measure of the difference between probability distributions. Given two distributions P_1 and P_2 over a single discrete random variable Y, the KL divergence is given by $D(P_1 \| P_2) = \sum_i P_1(y_i) \log \frac{P_1(y_i)}{P_2(y_i)}$. For

active learning, the most informative query is considered to be the one with the largest average KL divergence between a committee member's posterior label distribution and that of the consensus. I extend this approach from classification to sequence models by summing these average KL divergence scores using the marginals at each token position and, as with vote entropy, normalizing for length. I call this approach **Kullback-Leibler (KL)**:

$$\phi^{KL}(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{C}\sum_{c=1}^{C}\sum_{m=1}^{M} P_{\theta^{(c)}}(y_t = m)\log\frac{P_{\theta^{(c)}}(y_t = m)}{P_{\mathcal{C}}(y_t = m)}.$$

Here I use the shorthand again, and $P_{\mathcal{C}}(y_t = m) = \frac{1}{C}\sum_{c=1}^{C} P_{\theta^{(c)}}(y_t = m)$, denoting the "consensus" marginal probability that m is the label at position t in the sequence. Both of these disagreement measures are normalized for sequence length T. As with token entropy (3.1), this may bias the learner toward querying shorter sequences. To study the effects of normalization, I also conduct experiments with **non-normalized variants** ϕ^{TVE} and ϕ^{TKL}.

Additionally, I argue that these token-level disagreement measures may be less appropriate for most tasks than measuring the committee's disagreement about the label sequence $\mathbf{y}$ as a whole. Therefore, I propose **sequence vote entropy (SVE)**:

$$\phi^{SVE}(\mathbf{x}) = -\sum_{\hat{\mathbf{y}}\in\mathcal{N}^{\mathcal{C}}} P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C})\log P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C}),$$

where $\mathcal{N}^{\mathcal{C}}$ is the union of the N-best parses from all models in the committee $\mathcal{C}$, and $P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C}) = \frac{1}{C}\sum_{c=1}^{C} P(\hat{\mathbf{y}}|\mathbf{x};\theta^{(c)})$, or the "consensus" posterior probability for some label sequence $\hat{\mathbf{y}}$. This can be thought of as a QBC generalization of the N-best sequence entropy approach, where each committee member casts a vote for the posterior label distribution. I also explore a **sequence Kullback-Leibler (SKL)** variant:

$$\phi^{SKL}(\mathbf{x}) = \frac{1}{C}\sum_{c=1}^{C}\sum_{\hat{\mathbf{y}}\in\mathcal{N}^{\mathcal{C}}} P(\hat{\mathbf{y}}|\mathbf{x};\theta^{(c)})\log\frac{P(\hat{\mathbf{y}}|\mathbf{x};\theta^{(c)})}{P(\hat{\mathbf{y}}|\mathbf{x};\mathcal{C})}.$$

3.2.3 Expected Gradient Length

A third general active learning framework I consider is to query the instance that would impart the greatest change to the current model *if we knew its label*. Since we train discriminative models like CRFs using gradient-based optimization, this involves querying the instance which, if labeled and added to the training set, would create the greatest change in the gradient of the objective function (i.e., the largest gradient vector used to re-estimate parameter values).

Let $\nabla\ell(\mathcal{L};\theta)$ be the gradient of the log-likelihood ℓ with respect to the model parameters θ, as given by Sutton and McCallum (2006). Now let $\nabla\ell(\mathcal{L}\cup\langle\mathbf{x},\mathbf{y}\rangle;\theta)$ be the new gradient that would be obtained by adding the training tuple $\langle\mathbf{x},\mathbf{y}\rangle$ to $\mathcal{L}$. Since the query algorithm does not know the true label sequence $\mathbf{y}$ in advance, we must instead calculate the **expected gradient length (EGL)**:

$$\phi^{EGL}(\mathbf{x}) = \sum_{\hat{\mathbf{y}}\in\mathcal{N}} P(\hat{\mathbf{y}}|\mathbf{x};\theta)\Big\|\nabla\ell(\mathcal{L}\cup\langle\mathbf{x},\hat{\mathbf{y}}\rangle;\theta)\Big\|,$$

approximated as an expectation over the N-best labelings, where $\| \cdot \|$ is the Euclidean norm of each resulting gradient vector. I first introduced this approach in work on multiple-instance active learning (Settles et al., 2008b), and adapt it to query selection with sequences here. Note that, at query time, $\nabla\ell(\mathcal{L};\theta)$ should be nearly zero since ℓ converged at the previous round of training. Thus, we can approximate $\nabla\ell(\mathcal{L}\cup\langle\mathbf{x},\hat{\mathbf{y}}\rangle;\theta) \approx \nabla\ell(\langle\mathbf{x},\hat{\mathbf{y}}\rangle;\theta)$ for computational efficiency, because the training instances are assumed to be independent.

3.2.4 Fisher Information

I also introduce a query selection strategy for sequence models based on *Fisher information*, building on the theoretical framework of Zhang and Oles (2000). Fisher information $\mathcal{I}(\theta)$ represents the overall uncertainty about the estimated model parameters θ, as given by:

$$\mathcal{I}(\theta) = -\int_{\mathbf{x}} P(\mathbf{x}) \int_{\mathbf{y}} P(\mathbf{y}|\mathbf{x};\theta) \frac{\partial^2}{\partial\theta^2} \log P(\mathbf{y}|\mathbf{x};\theta).$$

For a model with K parameters, the Fisher information takes the form of a $K \times K$ covariance matrix. Our goal in active learning is to select the query that most efficiently minimizes the model variance reflected in $\mathcal{I}(\theta)$. One way to do this is to optimize the **Fisher information ratio (FIR)**:

$$\phi^{FIR}(\mathbf{x}) = -\text{tr}\left(\mathcal{I}_{\mathbf{x}}(\theta)^{-1}\mathcal{I}_{\mathcal{U}}(\theta)\right), \tag{3.3}$$

where $\mathcal{I}_{\mathbf{x}}(\theta)$ and $\mathcal{I}_{\mathcal{U}}(\theta)$ are Fisher information matrices for the unlabeled sequence $\mathbf{x}$ and the entire unlabeled pool $\mathcal{U}$, respectively. The trace function $\text{tr}(\cdot)$ is the sum of the terms along the principal diagonal of a matrix, thus ϕ^{FIR} provides us with a ratio given by the inner product of $\mathcal{I}_{\mathcal{U}}(\theta)$ and the inverse of $\mathcal{I}_{\mathbf{x}}(\theta)$. The leading minus sign again ensures that ϕ^{FIR} is a maximizer for use with the algorithm in Table 3.1.

Previously, active learning with Fisher information has only been investigated in the context of simple binary classification (Zhang and Oles, 2000; Hoi et al., 2006a). When employing the FIR strategy with sequence models such as CRFs, there are two additional computational challenges. First, we must integrate over all possible labelings $\mathbf{y}$, which can, as we have seen, be approximated as an expectation over the N-best labelings. Second, the inner product in the ratio calculation (3.3) requires inverting a $K \times K$ matrix for each $\mathbf{x}$. In most interesting natural language applications, K is very large (often hundreds of thousands of parameters), making this algorithm intractable. However, it is common in similar situations to approximate the Fisher information matrix with its diagonal vector (Nyffenegger et al., 2006). Thus we can estimate $\mathcal{I}_{\mathbf{x}}(\theta)$ using:

$$\mathcal{I}_{\mathbf{x}}(\theta) = \sum_{\hat{\mathbf{y}}\in\mathcal{N}} P(\hat{\mathbf{y}}|\mathbf{x};\theta) \left[\left(\frac{\partial \log P(\hat{\mathbf{y}}|\mathbf{x};\theta)}{\partial\theta_1}\right)^2 + \delta, \ldots, \left(\frac{\partial \log P(\hat{\mathbf{y}}|\mathbf{x};\theta)}{\partial\theta_K}\right)^2 + \delta \right],$$

and estimate $\mathcal{I}_{\mathcal{U}}(\theta)$ using:

$$\mathcal{I}_{\mathcal{U}}(\theta) = \frac{1}{U}\sum_{u=1}^{U} \mathcal{I}_{\mathbf{x}^{(u)}}(\theta).$$

A smoothing parameter $\delta \ll 1$ is added to prevent division by zero when computing the ratio. For CRFs, the partial derivative at the root of each element in the diagonal vector is given by:

$$\frac{\partial \log P(\hat{\mathbf{y}}|\mathbf{x};\theta)}{\partial \theta_k} = \sum_{t=1}^{T} f_k(\hat{y}_{t-1}, \hat{y}_t, \mathbf{x}_t) - \sum_{t=1}^{T} \sum_{y,y'} P(y, y'|\mathbf{x}) f_k(y, y', \mathbf{x}_t),$$

which is similar to the equation used to compute the training gradient for a particular instance (Sutton and McCallum, 2006), but without a regularization term. In a certain sense, the Fisher information ratio can be thought of as an extension of the expected gradient length algorithm, which tries to account for the the "representativeness" of an instance compared to the data in $\mathcal{U}$.

3.2.5 Information Density

It has been suggested that uncertainty sampling and QBC are prone to querying outliers (Roy and McCallum, 2001; Zhu et al., 2003). Figure 3.1 illustrates this problem for a binary linear classifier using uncertainty sampling. The least certain instance lies on the classification boundary, but is not "representative" of other instances in the distribution, so knowing its label is unlikely to improve accuracy on the data as a whole. QBC and EGL may exhibit similar behavior, by spending time querying possible outliers simply because they are controversial, or are expected to impart significant change in the model. I argue that this phenomenon can occur with sequence labeling tasks as well as with classification. To address this, I propose a new active learning approach called **information density (ID)**:

$$\phi^{ID}(\mathbf{x}) = \phi^{SE}(\mathbf{x}) \times \left(\frac{1}{U} \sum_{u=1}^{U} \text{sim}(\mathbf{x}, \mathbf{x}^{(u)}) \right)^{\beta}.$$

That is, the informativeness of $\mathbf{x}$ is weighted by its average similarity to all other sequences in $\mathcal{U}$, subject to a parameter β that controls the relative importance of the density term. In the formulation presented above, sequence entropy ϕ^{SE} measures the "base" informativeness, but we could just as easily use any of the instance-level strategies presented in the previous sections. Notice that this approach selects representative instances by explicitly modeling the instance distribution with a density term. This is in contrast to the Fisher information approach, which implicitly favors queries with Fisher information $\mathcal{I}_{\mathbf{x}}(\theta)$ that is not only high, but similar to that of the overall data distribution $\mathcal{I}_{\mathcal{U}}(\theta)$.

The density term requires us to be able to measure the similarity of two sequences. To do this, I first transform each $\mathbf{x}$, which is a sequence of feature vectors (tokens), into a single kernel vector denoted $\vec{\mathbf{x}}$:

$$\vec{\mathbf{x}} = \left[\sum_{t=1}^{T} f_1(x_t), \ldots, \sum_{t=1}^{T} f_J(x_t) \right], \tag{3.4}$$

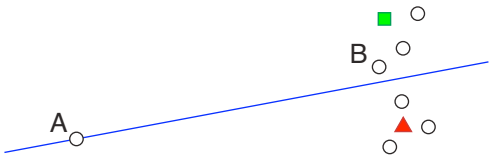


Figure 3.1: An illustration of when uncertainty sampling can be a poor strategy for classification. Shaded polygons represent labeled instances in $\mathcal{L}$, and circles represent unlabeled instances in $\mathcal{U}$. Since A is on the decision boundary, it would be queried as the most uncertain. However, querying B is likely to result in more information about the data distribution as a whole.

where $f_j(x_t)$ is the value of feature f_j for token x_t, and J is the number of features in the input representation[1]. In other words, sequence $\mathbf{x}$ is compressed into a fixed-length feature vector $\vec{\mathbf{x}}$, for which each element is the sum of the corresponding feature's values across all tokens. We can then use cosine similarity on this simplified representation:

$$\text{sim}_{cos}(\mathbf{x}, \mathbf{x}^{(u)}) = \frac{\vec{\mathbf{x}} \cdot \vec{\mathbf{x}}^{(u)}}{\|\vec{\mathbf{x}}\| \times \|\vec{\mathbf{x}}^{(u)}\|}.$$

Chapter 4 delves into the information density algorithm in more detail, e.g., examining the effectiveness of other similarity functions. For now, suffice it to say that cosine similarity performs quite well and is the only formulation we are concerned with in this chapter.

One potential drawback of information density is that the number of required similarity calculations grows quadratically with the number of instances in $\mathcal{U}$. For pool-based active learning, we often assume that the size of $\mathcal{U}$ is very large. However, these densities only need to be computed once, and are independent of the base information measure. Thus, when employing information density in a real-world interactive learning setting, the density scores can simply be pre-computed and cached for efficient lookup during the actual active learning process.

3.3 Sequence Labeling Data Sets

To compare the empirical performance of these sequence model query strategies, I conduct experiments across several benchmark information extraction and document segmentation data sets. These sequence labeling data sets are summarized in Table 3.2, and represent a variety of knowledge domains, corpus sizes, and numbers of labels or input features. The **CoNLL03** corpus (Sang and DeMeulder, 2003) is a collection of newswire articles annotated with four entities: person, organization, location, and misc. **NLPBA** (Kim et al., 2004) is a large collection of biomedical abstracts annotated with five entities of interest: protein, RNA, DNA, cell-line, and

[1]Note that $J \neq K$, and $f_j(x_t)$ here differs slightly from the feature definition given in Section 2.2.1. Since the labels y_{t-1} and y_t are unknown before querying, the K features used for model training are reduced down to the J input features here, which factor out any label dependencies.

Table 3.2: Properties of the different evaluation corpora.

Corpus	Entities	Features	Instances
CoNLL03	4	78,644	19,959
NLPBA	5	128,401	18,854
BioCreative	1	175,331	10,000
FlySlip	1	31,353	1,220
CORA:Headers	15	22,077	935
CORA:References	13	4,208	500
Sig+Reply	2	25	617
SigIE	12	10,600	250

cell-type. **BioCreative** (Yeh et al., 2005) and **FlySlip** (Vlachos, 2007) also comprise texts in the biomedical domain, annotated for gene entity mentions in articles from the human and fruit fly literature, respectively. **CORA** (Peng and McCallum, 2004) consists of two collections: a set of research paper headers annotated for entities such as title, author, and institution; and a collection of references annotated with BibTeX fields such as journal, year, and publisher. The **Sig+Reply** corpus (Carvalho and Cohen, 2004) is a set of email messages annotated for signature and quoted reply line segments. **SigIE** is a subset of the signature blocks from Sig+Reply which I have enhanced with several address book fields such as name, email, and phone (for more details, see Chapter 6). All corpora are formatted in the "IOB" sequence label representation[2] (Ramshaw and Marcus, 1995).

3.4 Experiments

I implement all fifteen query selection strategies described in Section 3.2 for use with CRFs, and evaluate them on all eight data sets. I also compare against two baseline strategies: random instance selection (i.e., "passive" supervised learning), and naïvely querying the longest sequence in terms of tokens. I use a typical feature set for each corpus based on the cited literature (including words, orthographic patterns, part-of-speech, lexicons, etc.). Where the N-best approximation is used, $N = 15$, and for all QBC methods $C = 3$. These values exhibited a good balance of accuracy and training speed in preliminary work; previous research involving QBC has also shown that small values of C often work well, and varying committee size can have surprisingly little effect (McCallum and Nigam, 1998b). For information density, I set $\beta = 1$ so that information and density terms have equal weight (in Chapter 4, I consider the effects of varying this parameter). In each experiment, $\mathcal{L}$ is initialized with five labeled instances chosen at random, and up to 150

[2]The IOB representation breaks an entity segment into two sublabels, prefixed B- for the beginning of a segment and I- for each subsequent token in a segment. For example, the words "ACME Inc." from Figure 2.1 would actually be labeled with "B-org I-org." This is standard practice for many sequence segmentation tasks.

queries are subsequently selected from $\mathcal{U}$ in batches of size $B = 5$. All results are averaged across five folds using cross-validation.

I evaluate each query strategy by constructing learning curves that plot the overall F_1 measure (as defined in Chapter 2) for all entities or segments as a function of the number of instances queried. To reduce clutter, I do not show learning curves for every experiment. Instead, Table 3.3 summarizes all results by reporting the area under the learning curve for all strategies on all data. Figure 3.2 presents a few representative learning curves for six of the corpora.

3.4.1 Discussion of Learning Curves

The first conclusion we can draw from these results is that there is no single clear winner. However, information density (ID), which I introduce in this work, stands out. It usually improves upon the base sequence entropy measure (SE), never performs poorly, and has the highest average area under the learning curve across all tasks. It seems particularly effective on large corpora, which is a typical assumption for the active learning setting. Sequence vote entropy (SVE), a QBC method I propose here, is also noteworthy in that it is fairly consistently among the top three strategies, although never the best. It is also interesting that the naïve "longest" baseline is marginally better than random, but still not competitive with the true active query strategies.

Secondly, the best uncertainty sampling strategies are least confidence (LC) and sequence entropy (SE), the latter being the dominant entropy-based method. Among the QBC strategies, sequence vote entropy (SVE) is the clear winner. We can conclude that these three methods are the best base information measures for use with information density.

Thirdly, query strategies that evaluate the entire sequence (SE, SVE, SKL) are generally superior to those which aggregate token-level information. Furthermore, the *total* token-level strategies (TTE, TVE, TKL) outperform their *length-normalized* counterparts (TE, VE, KL) in nearly all cases. In fact, the normalized variants are usually inferior even to the baselines. While an argument can be made that these shorter sequences might be easier to label from a human annotator's perspective, my work in Chapter 6 indicates that the relationship between instance length and actual labeling costs (e.g., elapsed annotation time) is not a simple one. Analysis of the experiment logs also shows that length-normalized methods are occasionally biased toward short sequences with little intuitive value (e.g., sentences with few or no entities to label). In addition, vote entropy appears to be a better disagreement measure for QBC strategies than KL divergence.

Finally, Fisher information (FIR), while theoretically sound, exhibits behavior that is difficult to interpret. It is sometimes the winning strategy, but occasionally only on par with the baselines. When it does show significant gains over the other strategies, these gains appear to be only for the first several queries (e.g., NLPBA and BioCreative in Figure 3.2). This inconsistent performance may be a result of the approximations made for computational efficiency. Expected gradient length (EGL) also appears to exhibit mediocre performance, and is likely not worth its additional computational expense.

Table 3.3: Detailed results for all query strategies on all evaluation corpora. Reported is the area under the F_1 learning curve for each strategy after 150 queries (maximum possible score is 150). For each corpus (column), the three best strategies are shown in bold. Small-font numbers in parentheses indicate the strategy's corresponding rank (one is best). The rightmost column summarizes results across all eight tasks by reporting the average area for each strategy. Sequence model query strategies that I introduce in this thesis are indicated with an asterisk (*).

Strategy	CoNLL03	NLPBA	BioCreative	FlySlip	Headers	References	Sig+Reply	SigIE	Average
				Other Methods					
ID*	**89.6** (3)	**73.1** (2)	**59.1** (1)	**126.8** (1)	80.2	88.7	131.5	88.5	**92.2** (1)
FIR*	81.7	**73.6** (1)	**58.8** (2)	118.2	79.1	87.1	**133.2** (1)	88.5	90.0
EGL*	87.3	69.3	51.5	**125.9** (2)	79.6	88.2	130.5	87.7	90.0
				Query-By-Committee					
SKL*	87.9	68.5	50.8	120.7	78.4	86.9	132.3	**89.7** (2)	89.4
SVE*	89.0	**71.8** (3)	**56.6** (3)	122.7	**80.7** (3)	**89.9** (3)	**132.8** (3)	**89.5** (3)	**91.6** (2)
TKL*	81.7	63.5	45.1	119.5	78.5	88.2	130.6	85.1	86.5
TVE*	86.7	66.9	49.2	124.1	79.7	88.7	132.1	**89.7** (2)	89.6
KL*	62.0	53.1	37.4	109.4	78.5	89.1	130.7	85.5	80.7
VE	45.9	52.4	35.2	113.3	72.8	85.1	131.4	**89.8** (1)	78.2
				Uncertainty Sampling					
NSE	89.1	68.9	50.5	124.1	80.4	89.4	**133.1** (2)	89.1	90.6
SE	**90.1** (1)	71.5	56.0	**125.4** (3)	**80.8** (2)	88.4	131.4	87.6	**91.4** (3)
TTE*	**89.7** (2)	70.9	53.0	124.9	78.5	88.6	131.6	88.3	90.7
TE	38.9	53.4	37.8	110.3	78.5	84.4	131.7	89.3	78.0
M	84.5	62.9	46.8	119.5	78.6	**91.5** (1)	132.3	87.3	87.9
LC	89.4	71.0	54.8	125.1	**81.4** (1)	89.8	132.1	88.8	**91.6** (2)
				Baselines					
Rand	78.8	59.9	34.6	112.1	76.0	**90.0** (2)	129.1	84.3	83.1
Long	79.4	67.6	26.9	121.0	78.2	86.0	129.6	82.7	83.9

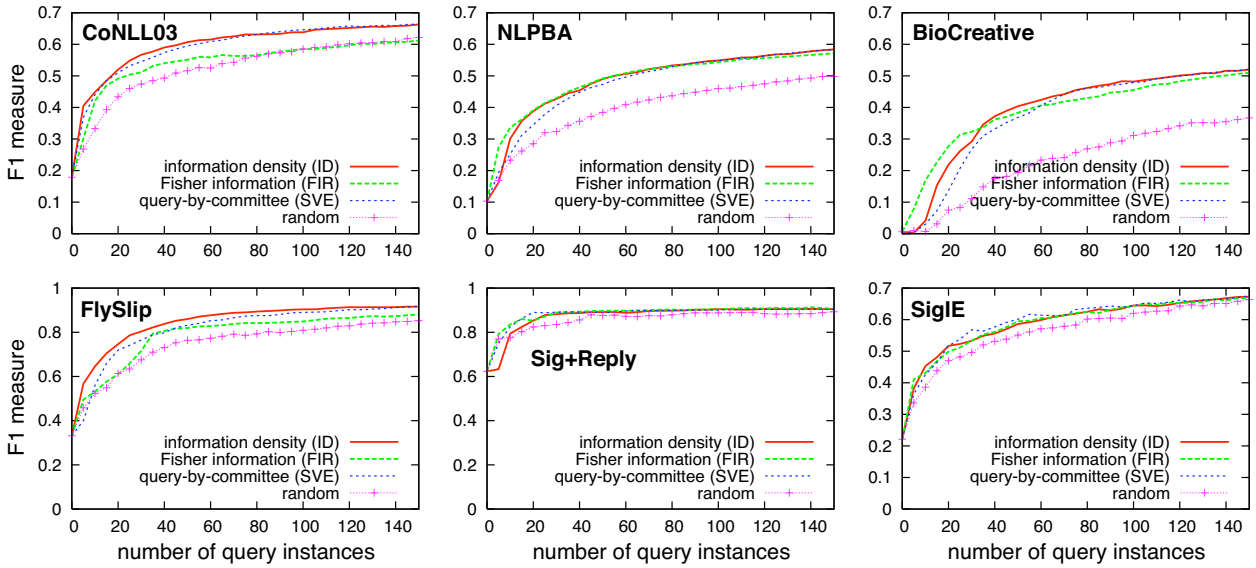


Figure 3.2: Learning curves for selected query strategies on six of the evaluation corpora.

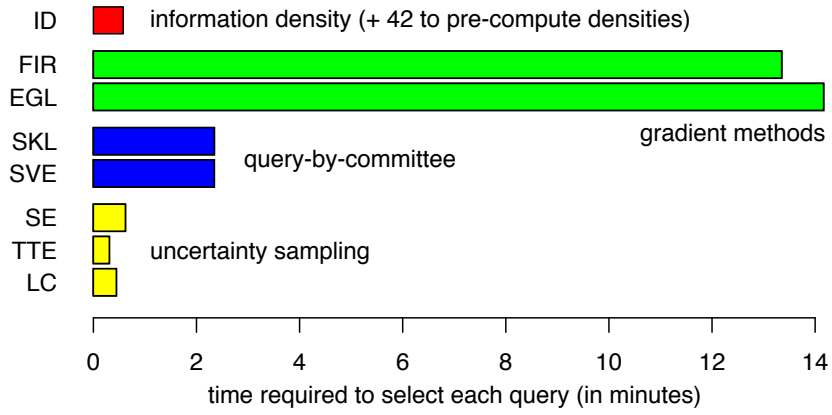


Figure 3.3: Run times for selected query strategies, averaged over the four largest corpora.

3.4.2 Discussion of Run Times

Now we turn our attention to the execution times for these query strategies using current hardware. Figure 3.3 summarizes the empirical run times for eight representative query selection algorithms. The uncertainty sampling methods are clearly the fastest (token-based variants run slightly faster), each routinely evaluating tens of thousands of sequences in under a minute. The QBC methods run about three times longer, which is sensible because of the committee size $C = 3$. These strategies must re-train multiple models with each query, resulting in a lag of around three minutes per query batch (and up to 20 minutes for corpora with more entity labels). We can conclude that the time complexity of QBC methods will scale linearly with C.

The expected gradient length and Fisher information methods are the most computationally expensive by far, because they must first perform inference over the possible labelings and then calculate gradients for each candidate label sequence. As a result, these algorithms require more than ten minutes (sometimes upwards of a half hour for the largest corpora) to select each query. Unlike the other strategies, their time complexities also scale linearly with the number of model parameters K which, in turn, increases as new sequences are labeled and added to $\mathcal{L}$.

As noted in Section 3.2.5, information density incurs a large computational cost to estimate the density weights, but these can be pre-computed and cached for efficient lookup. In my experiments, this pre-processing step takes less than one minute for the smaller corpora and about 42 minutes on average for the larger corpora. In particular, CoNLL03 and BioCreative were processed in about a half hour, and NLPBA was finished in just over one and a half hours. The density lookup causes no significant change in the run time of the base information measure (compare ID and SE in Figure 3.3). Given these results, I advocate information density with an uncertainty sampling base measure in practice, particularly for active learning with large corpora.

3.5 Summary and Future Work

In this chapter, I have presented a detailed analysis of active learning for sequence labeling tasks. Specifically, I described and criticized the query selection strategies that have been proposed for probabilistic sequence models to date, and introduced nine novel strategies to address some of their shortcomings. My large-scale empirical evaluation demonstrates that some of these newly proposed methods advance the state of the art in active learning with sequence models. Specific conclusions from this work can be summarized as follows:

- Information density, a new query framework that I introduce here, is well-suited to active learning with sequences. It performs particularly well on large corpora (a common assumption for pool-based active learning), and is quite fast if densities are pre-computed.

- Sequence vote entropy, a QBC method I introduce here, also performs reasonably well, though it is slightly more computationally expensive.

- In general, the informativeness of an instance $\mathbf{x}$ should be estimated using distributions over the full label sequence $\mathbf{y}$, as opposed to aggregating over token label distributions. Furthermore, normalizing for sequence length tends to perform poorly.

Chapter 4 explores the properties of the information density algorithm in further detail. Future work in active learning for sequence labeling tasks might include devising efficient, exact algorithms for the query strategies which currently resort to using approximations. For example, just as the dynamic programming approach for calculating the exact sequence entropy (SE) is superior to its N-best approximation (NSE) counterpart, I suspect that other strategies such as the QBC methods, expected gradient length, and Fisher information can be improved if fast and exact algorithms can be found.

Chapter 4

More on the Information Density Algorithm

This chapter explores the information density algorithm (which I introduced in the last chapter) in more detail. I experiment with and reason about different similarity functions, examine the impact of varying the β parameter, and consider the implications of using information density in the *batch-mode* active learning setting.

4.1 Similarity Functions

The information density algorithm, introduced in Section 3.2.5, can be generically written:

$$\phi^{ID}(\mathbf{x}) = \phi^{Z}(\mathbf{x}) \times \left(\frac{1}{U} \sum_{u=1}^{U} \text{sim}(\mathbf{x}, \mathbf{x}^{(u)}) \right)^{\beta}.$$

This query strategy is a *wrapper algorithm*, i.e., it is compatible with virtually any "base" informativeness measure ϕ^{Z}, and any desired similarity function. As a result, it is fairly straightforward to implement for any sort of learning problem, provided that appropriate informativeness and instance similarity measures are available for the task at hand. In this chapter, I consider both sequence labeling and text classification tasks, although for simplicity I will stick with the sequence notation of $\mathbf{x}$ for an instance and $\mathbf{y}$ for its labeling. The concepts are straightforward to extend to classification using an instance x with label y.

At the core of the information density approach is the similarity function, which is used to compute the density term. In the last chapter, I used the **cosine similarity** function:

$$\text{sim}_{cos}(\mathbf{x}, \mathbf{x}^{(u)}) = \frac{\vec{\mathbf{x}} \cdot \vec{\mathbf{x}}^{(u)}}{\|\vec{\mathbf{x}}\| \times \|\vec{\mathbf{x}}^{(u)}\|},$$

where $\vec{\mathbf{x}}$ is a fixed-length feature vector that represents a "kernelized" version of the sequence $\mathbf{x}$, as described in Equation 3.4[1]. However, in this section I will examine and compare against two other similarity measures that have been used for estimating density in machine learning applications.

McCallum and Nigam (1998b) also developed a density-weighted approach to active learning for text classification. In their work, a QBC strategy is employed for a naïve Bayes classifier, and

[1] For classification, the vector $\vec{x}$ is simply the input feature vector, as described in Section 2.1.

the informativeness of each instance is weighted by its average similarity to the rest of the pool $\mathcal{U}$ using an exponentiated **KL divergence similarity** function:

$$\text{sim}_{KL}(\mathbf{x}, \mathbf{x}^{(u)}) = \exp\left(-\gamma_1 \sum_{j=1}^{J} P(f_j|\vec{\mathbf{x}}) \log \frac{P(f_j|\vec{\mathbf{x}})}{\gamma_2 P(f_j|\vec{\mathbf{x}}^{(u)}) + (1-\gamma_2)P(f_j)}\right).$$

The first parameter γ_1 controls the "sharpness" of the divergence measure, and the second γ_2 determines how much smoothing to use on the encoded distribution in the denominator. We can estimate the probability $P(f_j|\vec{\mathbf{x}}) = \vec{\mathbf{x}}_j/|\vec{\mathbf{x}}|_1$, which is the value of input feature f_j divided by the "Manhattan norm" of the vector $\vec{\mathbf{x}}$ (i.e., the sum of all feature values in the vector). The smoothing term $P(f_j)$ is simply the marginal probability of feature f_j over all instances in the pool $\mathcal{U}$. Note that, unlike the other similarity functions I consider here, KL divergence is non-symmetric and must be re-computed in both directions, incurring twice the computational cost.

Another common similarity measure used to estimate density (Zhu, 2005a; Lee, 1999) is an exponentiated Euclidean distance, which defines a **Gaussian similarity** function:

$$\text{sim}_{Gauss}(\mathbf{x}, \mathbf{x}^{(u)}) = \exp\left(-\sum_{j=1}^{J} \frac{(\vec{\mathbf{x}}_j - \vec{\mathbf{x}}_j^{(u)})^2}{\alpha^2}\right).$$

Here, α^2 is the variance in the shape of the Gaussian. It is possible to use a different parameter α_j for each input feature f_j, but tuning these parameters individually can be difficult since we do not necessarily know what to optimize them for. Furthermore, results using the simpler formulation (reported on shortly) suggest that tuning these extra parameters may not be worth the effort. Therefore, I only consider the simpler formulation presented here.

I conduct experiments to compare these different similarity functions for both sequence labeling and classification tasks. For the sequence labeling experiments, I use the four largest information extraction corpora from the previous chapter: CoNLL03, NLPBA, BioCreative, and FlySlip. The learning algorithm is again a CRF (Section 2.2.1). For the classification experiments, I use four topical subsets of the 20 Newsgroups corpus: `rec.*` (four labels: baseball, hockey, autos, and motorcycles), `sci.*` (four labels: crypt, electronics, med, and space), `talk.*` (four labels: guns, mideast, politics, and religion), and `comp.*` (five labels: graphics, ibm, mac, ms-windows, and windows.x). The `comp.*` subset was also used by McCallum and Nigam (1998b) in their work using the KL divergence similarity function. The learning algorithm used in my classification experiments is a MaxEnt model (Section 2.1.1).

For the sequence labeling experiments, the labeled set $\mathcal{L}$ is initialized with five randomly selected sequences; 10 instances are used for classification. For the sequence tasks, learning curves are constructed using the overall F_1 measure; accuracy is used for classification. In all experiments, queries are selected from $\mathcal{U}$ one at a time, and results are averaged across ten folds using cross-validation[2]. For the KL divergence similarity function, I use parameter values $\gamma_1 = 3$ and $\gamma_2 = 0.5$,

[2]Note that the setup for sequence labeling experiments here differs from the previous chapter in that batches are of size $B = 1$, and there are twice as many folds in the cross-validation. The evaluations in this chapter involve fewer experiments, thus we can afford the extra accuracy this newer methodology provides.

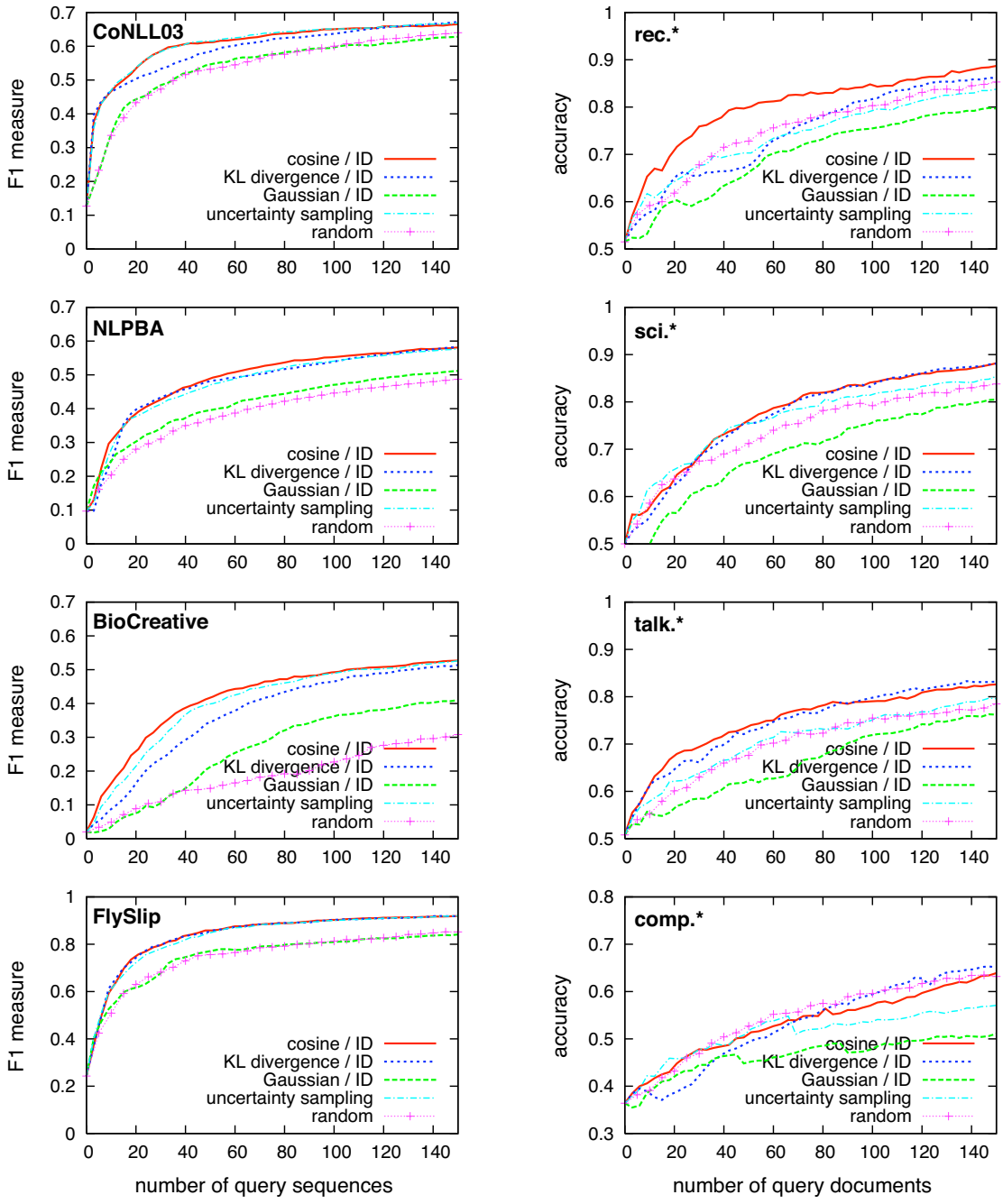


Figure 4.1: Learning curves for information density (ID) using the three different similarity functions compared to uncertainty sampling and random selection: (left) the four sequence labeling tasks, (right) the four text classification tasks.

following the work of McCallum and Nigam (1998b). For the Gaussian similarity function, I use $\alpha = 5$ based on some preliminary work to tune it. I again set $\beta = 1$ so that the information and density terms have equal weight. In all experiments, the base information measure ϕ^Z is an uncertainty sampling strategy. For the sequence tasks, I again use sequence entropy (3.2), and for the classification tasks I use the posterior label entropy. The three information density variants are compared against two baselines: the unweighted uncertainty strategy, and random sampling.

Figure 4.1 present learning curves for the three information density formulations compared to the two baselines for the four sequence labeling and four text classification corpora. We can see that, as in the previous chapter, information density with a cosine similarity function generally improves upon the base uncertainty sampling strategy for these data sets (both sequence labeling and classification tasks). The improvements over uncertainty sampling are in fact statistically significant for the majority of points on the horizontal axis (using a two-tailed t-test at 95%) for all but the CoNLL03 and FlySlip corpora. One surprising result is that uncertainty sampling actually performs worse than random sampling for the `comp.*` task, but the cosine-based information density appears to compensate for the base measure's shortcomings, becoming a competitive strategy again.

If we compare the performance of the different similarity functions, we can see that cosine is the clear winner. KL divergence exhibits somewhat erratic behavior, often on par with cosine similarity but occasionally much worse, and Gaussian similarity is rarely better than random (and consistently worse on the classification tasks). One possible explanation for this has to do with the particular problem domain. These experiments explore natural language applications, for which instances are represented as feature vectors which may be quite sparse (i.e., features represent words, thus most feature values tend to be zero because not all words appear in all documents). It has been shown that cosine similarity is an effective similarity measure for these sparse, high-dimensional feature spaces (Manning and Schütze, 1999; Lee, 1999). It stands to reason, then, that it should work well for information density as well.

These results underscore the importance of choosing the right similarity function when employing information density for active learning. One must first consider the properties of the task and possibly the learning model as well, and choose a similarity measure that has appropriate properties. For example, if we were learning in a domain where Gaussian classifiers are shown to work well, employing information density with the Gaussian similarity function might be the best active learning approach.

4.2 Weighting the Density Term

Recall that the β parameter controls the relative importance of the density term. For example, as $\beta \to 0$, information density gracefully degrades to the base query selection strategy, while as $\beta \to \infty$, the density term becomes predominantly more important. So far, I have only discussed formulations of information density where the $\beta = 1$, i.e., both the information and density terms have equal weight. However, it may be that higher or lower values of this exponent are more appropriate in some domains.

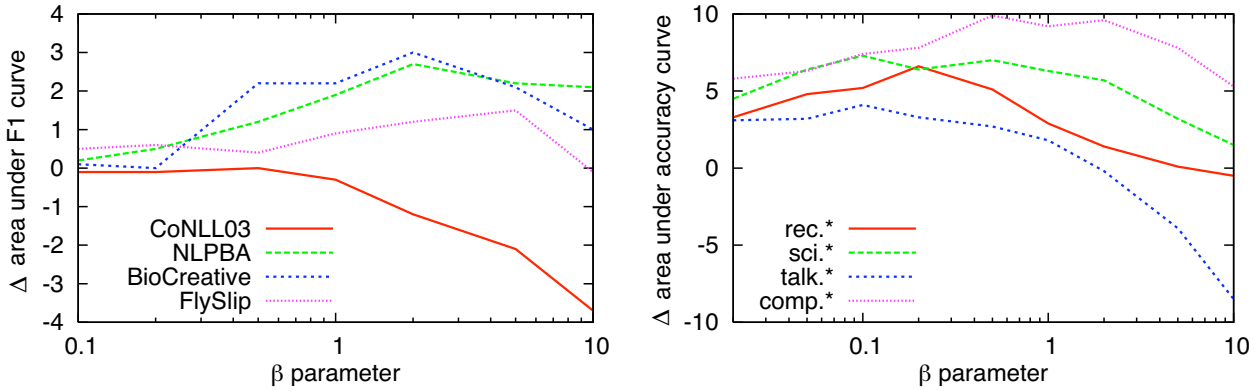


Figure 4.2: The effect of varying the β parameter on: (left) sequence labeling and (right) text classification tasks. These plots show the change in the area under the learning curve relative to the base strategy (after 150 queries) as a function of β. Note the log-scale on horizontal axes, and that plots for the two problems are scaled differently.

To study this, I conduct several experiments using the various information extraction and text classification tasks from the previous section. For each task, I again use entropy-based uncertainty sampling as a base measure, coupled with the cosine similarity function. Other aspects of experimental setup are identical to Section 4.1. Figure 4.2 illustrates the effect of varying β for each of these eight data sets. The plots show the relative increase (or decrease) of the area under the learning curve for the first 150 queries as a function of β. We can see that, with the exception of the CoNLL03 information extraction corpus, there is some value of β that improves upon the base query strategy for all of these tasks. As expected, the change approaches zero for smaller values of β, and for all tasks there is some value at which the gains begin to decrease (eventually resulting in negative effects), typically around $\beta = 2$ or greater. Note that all text classification tasks show gains even for very small values (e.g., $\beta = 0.02$), and that the "optimal" value of β appears to be different for the two problem domains. For the sequence labeling tasks, the value exhibiting the highest overall gain appears to be around $\beta = 2$ (for the tasks that show any improvement), while for text classification it appears to be in the range $\beta = 0.2$ to $\beta = 0.5$. In both domains, the initial value of $\beta = 1$ (before any tuning) appears to work relatively well.

4.3 Batch-Mode Active Learning

Most work in active learning, including the majority of this thesis, is concerned with *serial* active learning. That is, queries are selected in serial one at a time (or in small batches). The experiments in Chapter 3 used small batches of size $B = 5$ for computational efficiency due to the sheer number of experiments that needed to be run, and because some query strategies are slow to compute. Even with efficient query strategies, however, we may want to query in batches for other reasons, such as the speed of the actual training procedure. Sutton and McCallum (2006)

Given: Labeled set $\mathcal{L}$, unlabeled pool $\mathcal{U}$, query strategy $\phi(\cdot)$, query batch size B

repeat
  // learn a model using the current $\mathcal{L}$
  $\theta = \text{train}(\mathcal{L})$;
  // initialize the empty query batch
  $\mathcal{Q} = \{\}$;
  for $b = 1$ **to** B **do**
    // select the most informative instance
    $\mathbf{x}_b^* = \text{argmax}_{\mathbf{x} \in \mathcal{U}} \, \phi(\mathbf{x})$;
    // move the selected query from $\mathcal{U}$ *to* $\mathcal{Q}$
    $\mathcal{Q} = \mathcal{Q} \cup \mathbf{x}_b^*$;
    $\mathcal{U} = \mathcal{U} - \mathbf{x}_b^*$;
  end
  // query each instance in $\mathcal{Q}$ *and add to* $\mathcal{L}$
  foreach $\mathbf{x} \in \mathcal{Q}$ **do**
    $\mathbf{y} = \text{label}(\mathbf{x})$;
    $\mathcal{L} = \mathcal{L} \cup \langle \mathbf{x}, \mathbf{y} \rangle$;
  end
until *some stopping criterion* ;

Table 4.1: A greedy batch-mode active learning algorithm.

report that, for example, a typical information extraction corpus with five entities requires less than two hours to estimate CRF model parameters. However, a part-of-speech tagging corpus with 45 labels can take over a week to train. While these numbers reflect learning from very large labeled sets with hundreds of thousands of words (rather than the smaller labeled sets that hopefully result from active learning), this illustrates the fact that as the learning task becomes sufficiently complex, the re-training process in-between queries can become very slow. This may make "interactive" learning impractical.

One way of addressing this limitation is with *batch-mode* active learning, in which a large batch of queries are selected at once from $\mathcal{U}$, then labeled as a group, and added to the training set $\mathcal{L}$ together. In this way, the active learning process need not be highly interactive, as the learner can re-train using a large set of new labeled instances and the annotator can move on to other things until the model is ready with its new batch of queries (perhaps overnight). It has been shown that greedily querying the "N most informative" queries—according to, say, uncertainty sampling—can often be a poor strategy for batch-mode active learning, and several more sophisticated batch-mode methods have been proposed for classification (Guo and Schuurmans, 2008; Hoi et al., 2006a). More specifically, these approaches are based on heuristics that greedily construct query batches that are not only informative and representative, but contain a diverse set of instances as well.

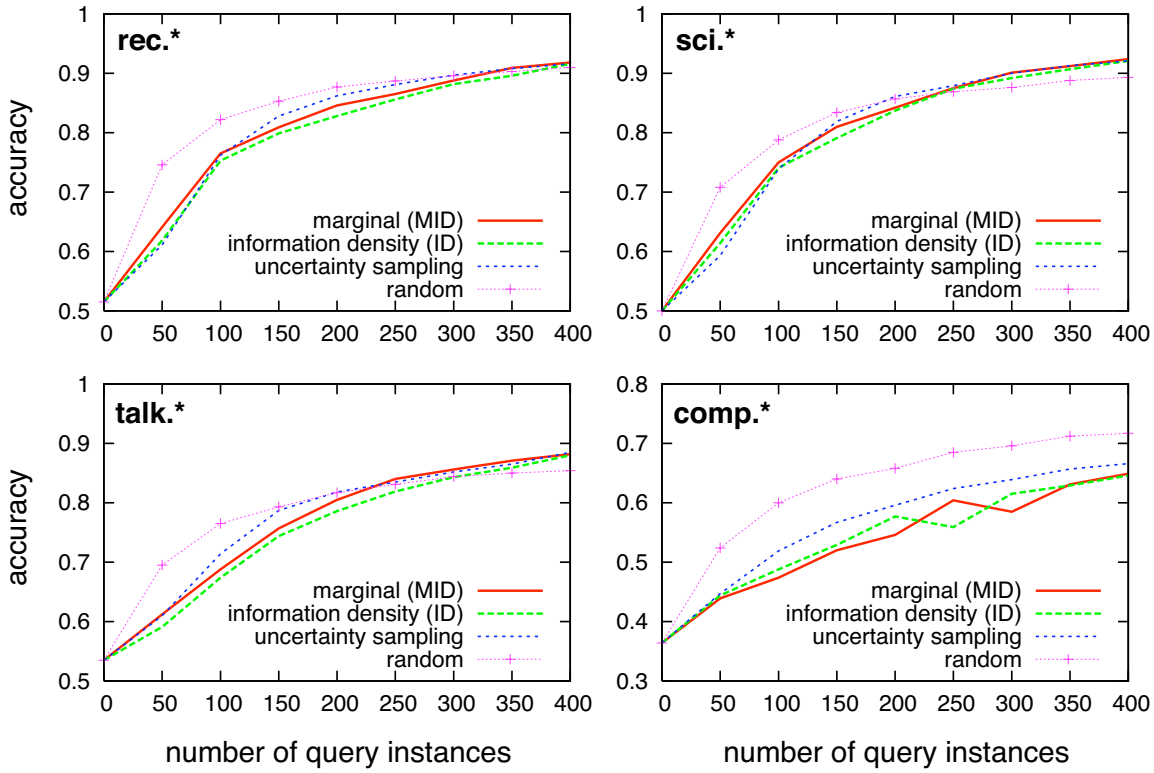


Figure 4.3: Learning curves for batch-mode active learning using batch size $B = 50$.

To study to implications of the information density approach to batch-mode active learning, I propose a variant that incorporates batch diversity into its selection strategy. This results in a strategy I call **marginal information density (MID)**:

$$\phi^{MID}(\mathbf{x}) = \phi^{Z}(\mathbf{x}) \times \left(\lambda \frac{1}{U} \sum_{u=1}^{U} \text{sim}(\mathbf{x}, \mathbf{x}^{(u)}) + (1 - \lambda) \frac{1}{Q} \sum_{q=1}^{Q} \text{diff}(\mathbf{x}, \mathbf{x}^{(q)}) \right)^{\beta},$$

because we wish to querying informative instances that are in marginally dense regions with respect to instances in the current batch. Here, Q is the size of the current query batch $\mathcal{Q}$, the $\text{diff}(\cdot, \cdot)$ function computes the difference (dissimilarity) between two instances, and λ is a parameter that controls the tradeoff between an instance's average similarity to the pool $\mathcal{U}$ (density) and its average dissimilarity to the current batch $\mathcal{Q}$ (diversity). If $\lambda = 1$, for example, this approach is equivalent to the original information density formulation from before. As $\lambda \to 0$, the diversity among instances in the batch becomes more important. Table 4.1 presents a greedy batch-mode active learning algorithm that can be used with any query strategy, including marginal information density.

To study the effectiveness of marginal information density in a batch-mode active learning setting, I conduct several experiments using the text classification tasks from this chapter. Because the cosine similarity function is always between zero and one, we can simply define the dissimilarity function to be $\text{diff}_{cos}(\cdot, \cdot) = 1 - \text{sim}_{cos}(\cdot, \cdot)$. For the density vs. diversity tradeoff, I use $\lambda = 0.5$

in these experiments, i.e., both terms have equal weight. For the exponent I again use $\beta = 1$. Marginal information density is compared to three baseline batch-mode strategies: information density, uncertainty sampling, and random sampling.

Figure 4.3 presents learning curves for these strategies using a batch of size $B = 50$ (similar results are obtained for batch sizes 10, 25, and 100). When queries are limited to being made in batches, random sampling outperforms the active approaches for all four classification tasks, which is consistent with some previously reported results for batch-mode active learning (Guo and Schuurmans, 2008). One explanation for this is that greedy selection prefers very similar instances in the query batch (after all, since these instances are deemed highly informative it is likely that they inhabit similar regions of the feature space). Information density may actually exacerbate this effect, by biasing instances toward dense regions. However, we do see that the marginal approach, which encourages diversity in the batch, improves slightly upon regular information density and sometimes the base uncertainty measure as well, although it is still nowhere near random sampling for batches. It may be that more appropriate values of β and λ could result in better learning curves. Exploring other notions of "diversity" in queries might also be useful. For example, Kim et al. (2006) employ a different strategy favoring queries that are dissimilar, on average, to instances already in the labeled set $\mathcal{L}$.

4.4 Summary and Future Work

In this chapter, I have more deeply analyzed the information density algorithm introduced in Chapter 3. This analysis includes exploring different similarity measures used in computing the density term, and a discussion of why and when different measures are appropriate. I also examined the effect of varying the β exponent, and considered a preliminary variant for batch-mode active learning which encourages diversity in addition to density and informativeness. Specific conclusions from this work can be summarized as follows:

- The appropriate similarity function for information density appears to depend on the task at hand. For example, cosine similarity should be used for large, sparse feature spaces such as text domains.
- The best value for the β exponent may depend on the problem domain, but values between 0.5 and 1.0 appear to be fairly robust for the tasks considered here.
- Information density, like most base query strategies, is not well-suited to greedy batch-mode active learning. However, a marginal variant which incorporates batch diversity into its value assessment may provide a simple and scalable solution for the batch setting.

There are many possibilities for future work regarding information density. One direction involves approximating the density term to help scale the approach to very large data sets (e.g., millions of instances). I have so far only considered an approach that exhaustively compares all instances, the time complexity of which is quadratic in the size of $\mathcal{U}$. Solutions to this problem may entail sub-sampling the pool, or sparsely computing density using only the nearest neighbors

for each instance. Another direction includes active learning with a gradually decaying value for β. The intuition here is that early in active learning the model should focus on representative instances, but as learning proceeds it should be allowed to explore the margins more freely. Another important direction for future work is the design of better similarity measures for structured instances, such as sequences. In this thesis, I use a simple kernel function that compresses a sequence $\mathbf{x}$ into a single fixed-length feature vector $\vec{\mathbf{x}}$. However, even this method loses some potentially important information about features that co-occur at the token level, which may explain why information density is not as highly effective for sequence learning tasks as it is for simpler classification tasks. Finally, developing a better generalization of information density for the batch-mode active learning setting may hold some promise for large-scale active learning with structured instances, by allowing the learner to query large sets of instances at once.

Chapter 5

Multiple-Instance Active Learning

This chapter presents a novel framework for active learning in *multiple-instance* (MI) settings. In MI learning problems, instances are naturally organized into bags and it is the bags, instead of individual instances, that are labeled for training. This chapter considers active learning in MI settings as a way to even further reduce the labeling burden in problem domains where labels can be acquired at both bag-level and instance-level granularities. An approach like this is well motivated in learning settings where it is inexpensive to acquire bag labels and possible (but expensive) to acquire more fine-grained instance labels. I propose and explore four different active learning scenarios for MI problems, and present a training algorithm that learns from labels at these mixed levels of granularity. I also introduce and evaluate several active query selection strategies motivated by the MI setting. Experiments show that active learning with instance labels can significantly improve the performance of an MI learning algorithm. Much of the work in this chapter has been previously published (Settles et al., 2008b).

5.1 Introduction

A limitation of traditional supervised learning algorithms is that they require instance labels which are often difficult or expensive to obtain. The *multiple-instance* (MI) learning framework (Dietterich et al., 1997) can, in some cases, address this handicap by relaxing the granularity at which labels are given. The MI setting was introduced in Section 2.2.2, and is briefly reviewed here. In the MI setting, instances are grouped into *bags* (i.e., multi-sets) which may contain an arbitrary number of instances. A bag $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$ is labeled negative if every instance x_n it contains is actually negative. A bag is labeled positive, however, if at least one of its instances is positive (note that positive bags may also contain negative instances).

The main challenge of multiple-instance learning is that, to induce an accurate model of the target concept, the learner must determine which instances in positive bags are actually positive, even though the ratio of negatives to positives in these bags can be arbitrarily high. For many MI problems, such as the content-based image retrieval (CBIR) and text classification tasks illustrated in Figure 2.2, it is possible to obtain labels both at the bag level and directly at the instance level. Labeling all instances, however, would be expensive. Again, the rationale for formulating a learning task as an MI problem is that it allows us to take advantage of coarse-granularity labelings that may be available at low cost, or even for free. Instead, consider a family of approaches that

Table 5.1: Four multiple-instance active learning scenarios.

	Scenario I	**Scenario II**	**Scenario III**	**Scenario IV**
Query unit	unlabeled bag	unlabeled instance from positive bag	labeled positive bag	labeled positive bag
Label(s) obtained for	query bag	query instance	*any positive* instance in the query bag	*all* instances in the query bag

involves selectively obtaining the labels of only certain instances in the context of MI learning. In particular, consider obtaining labels for selected instances in positive bags, since the labels for instances in negative bags are implicitly known.

I argue that whereas multiple-instance learning reduces the burden of labeling data by getting labels at a coarse level of granularity, we may also benefit from selectively labeling some part of the training data at a finer level of granularity. Hence, the aim of this work is to explore various approaches to *multiple-instance active learning* as a way to efficiently overcome the inherent ambiguity of the MI framework, while still keeping label costs low.

Table 5.1 presents several MI active learning scenarios we might consider. Scenario I, which is analogous to standard supervised active learning, simply allows the learner to query for labels of unlabeled bags. Scenario II is one in which the bags in the training set are assumed to be already labeled, and the learner is allowed to query and obtain labels for selected instances from positive bags. For example, the learner might query on particular image segments in a CBIR task, or passages of text in a document classification task (e.g., the highlighted instances shown in Figure 2.2). If the instance label is positive, the learner now has direct evidence for the positive class. If the label is negative, the learner knows to focus its attention to other instances from that bag, also reducing ambiguity. Scenario III also assumes that bag labels are available, but relaxes *which* instance label must be obtained. In this setting, the learner may query a positive bag and the oracle provides a label for any positive instance in that bag. For example, the learner might query a document in a text classification task and allow the annotator to label the first positive passage he or she finds, without having to read through the entire document. Scenario IV similarly queries positive bags, but asks the oracle to label all instances in the bag. For example, the learner might query a positive image in the CBIR domain, and ask the oracle to label all segments that belong to the target object (and the remaining segments are therefore implicitly negative). An additional scenario might assume that some bags are labeled and some are not, and the learner is able to query on (i) unlabeled bags, (ii) unlabeled instances in positive bags, or (iii) some combination thereof. In this work, I focus on the four scenarios summarized in Table 5.1, and assume all queries for a given learning application are of the same type.

This chapter presents three main contributions to research in MI active learning. First, I propose several bag-level and instance-level query strategies designed for each of the four active learning

scenarios. Second, I empirically evaluate all of these scenarios and query strategies on a large collection of multiple-instance tasks, and offer a discussion of when certain approaches may be more appropriate in practice. The rest of this chapter is organized as follows. Section 5.2 presents the mixed-granularity learning algorithm I have developed for MI learning models like MILR. Section 5.3 defines several bag-level and instance-level query selection algorithms for the various MI active learning scenarios described above. Section 5.4 describes several MI data sets, and Section 5.5 presents experiments to evaluate the various MI active learning scenarios and algorithms. Finally, Section 5.6 offers a summary of findings and suggestions for future work for MI active learning.

5.2 Learning from Labels at Mixed Granularities

For most of the MI active learning scenarios described in the last section, labels can be provided at the instance level of granularity. For example, in Scenario II an active MI learner may query an instance x_n and the corresponding instance label y_n is provided by the oracle. We would like to include a direct training signal for this instance in the training procedure for the model, in this case log-likelihood optimization for MILR (see Section 2.2.2). However, the objective function $\ell(\mathcal{L};\theta)$ is defined in terms of bag-level likelihood, not instance-level likelihood. Consider, though, that in MI learning a labeled instance is effectively the same as a labeled bag that contains only that instance. So when the label for an instance is known, we can transform the training set by adding a new training tuple $\langle\{x_n\}, y_n\rangle$ to $\mathcal{L}$, where $\{x_n\}$ is a new *singleton bag* containing only a copy of the labeled instance, and y_n is its corresponding instance label. If x_n is positive, it may seem reasonable to simply substitute it for the original bag $\mathcal{X}$ in the training set, but doing so assumes that a bag contains one and only one positive instance. This assumption is too strong for many MI learning tasks, as can be seen later. Instead, a copy of the query instance also remains in the original bag, enabling the learner to compute the remaining instance gradients as described below.

Let us assume that the oracle labels instance x_n as positive. Because the objective function ℓ will guide the learner toward classifying the singleton query instance in the new positive tuple $\langle\{x_n\}, 1\rangle$ as positive, it will tend to classify the original bag $\mathcal{X}$ positive as well. Conversely, if the oracle's answer is negative, the tuple $\langle\{x_n\}, 0\rangle$ will be added and the learner will tend to classify the instance negative in the original bag. This will affect the other instance gradients via the combining function and will guide the learner to focus on other potentially positive instances in that bag.

It may seem that this effect on the original bag could be achieved by simply clamping the instance output $o_n \equiv y_n$ during training, rather than creating new singleton bags. However, this has the undesirable property of eliminating the training signals for both bags and instances. If $o_n \equiv 1$, the combining function output o for the bag would be extremely high, making the log-likelihood nearly one, thus maximizing the objective function without actually updating any model parameters. If $o_n \equiv 0$, the instance would contribute nothing to the combining function, thus the learner would get no training signal at all for that instance (though in this case the learner can still focus its calculations on other positive-looking instances in the bag). It is possible to combine clamped instance outputs with the singleton bag approach to overcome this problem, but

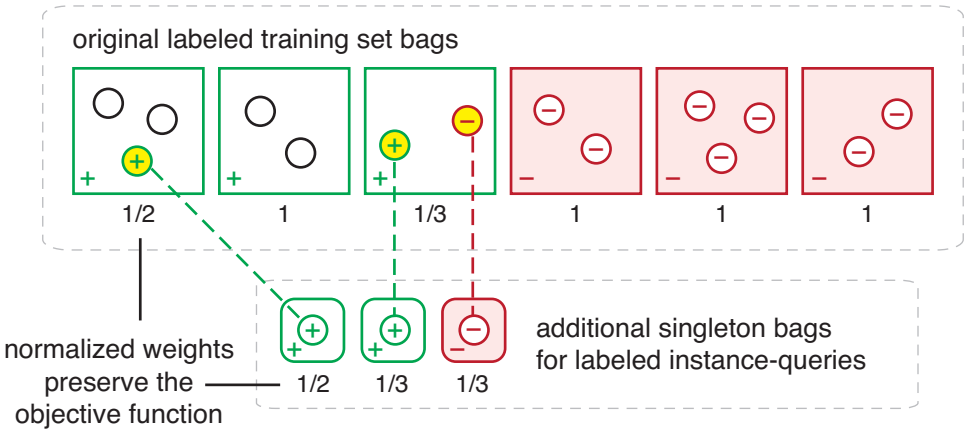


Figure 5.1: An example MI training set after three instances have been labeled and added to $\mathcal{L}$. Squares represent labeled bags, circles represent instances (some instance labels are implicitly known), and boxes with rounded corners represent labeled singleton bags. The labeled instances are highlighted, shown with dashed lines to their corresponding singleton bags. The fractions below each bag indicate that bag's training weight.

my research indicates that this has no empirical advantage over adding singleton bags using the method described above.

Figure 5.1 shows what an example MI training set might look like after three instances have been labeled. Note that simply adding singleton bags will alter the objective function by adding weight, albeit indirectly, to bags that have been queried more often. To control for this effect, each bag and all its queried singleton bags are uniformly weighted to sum to one when computing the value and gradient during training. Let $Q \leq N$ be the number of queried instances from a bag, and let the weight of said bag (and thus all of its queried instances) be $\frac{1}{Q+1}$. Therefore, an unqueried bag has a weight of one, a bag with one instance query and its derived singleton bag both have weight 1/2, a bag with two instance queries has weight 1/3, and so on.

While this mixed-granularity learning algorithm (and the active learning algorithms in the next section) are described in terms of a log-likelihood formulation of multiple-instance logistic regression, it is important to note that these approaches generalize to any multiple-instance classifier that outputs instance-level probabilities used with differentiable combining and objective functions. For example, the original formulation of Diverse Density (Maron and Lozano-Perez, 1998) couples a Gaussian instance model with a noisy-or combining function.

5.3 Query Selection Algorithms

In order to select queries, an active learner must have a way of assessing how *informative* each bag or instance is. Again, let $\phi(\cdot)$ be an query selection strategy that evaluates the informativeness

of potential queries. The remainder of this section presents several formulations of $\phi(\cdot)$ that can be used for the various MI active learning scenarios described in Section 5.1.

5.3.1 Querying Bags

First let us consider selecting bags for labeling. The methods in this section apply to MI active learning Scenario I from Table 5.1, where bags are queried from an unlabeled pool $\mathcal{U}$ and bag-level labels are provided by the oracle. These query selection strategies can also be employed for Scenarios III and IV, where positive bags are queried and instance-level labels are obtained for any or all unlabeled instances in the query bag.

We begin with query strategies in the uncertainty sampling framework, which queries the bag about which the model is least certain how to label. Using the posterior label entropy to evaluate uncertainty, let us define **bag uncertainty (BU)** to be:

$$\phi^{BU}(\mathcal{X}) = -\big(o \log o + (1-o)\log(1-o)\big),$$

where $o = P(y = 1|\mathcal{X}; \theta)$ is shorthand for the posterior probability that $\mathcal{X}$ is positive under the model. Note that the particular uncertainty measure we use (entropy in this case) is not especially critical; the important properties are that its minima are at zero and one, its maximum is at 0.5, and it is symmetric about 0.5.

Estimating uncertainty at the bag level can possibly obscure the model's underlying uncertainty about the instances within the bag. This effect is undesirable for Scenarios III and IV, where we wish to query positive bags (whose labels are known), but in order to obtain labels for one or more of its constituent instances. To study the differences between using bag-level vs. instance-level uncertainty in such cases, I also explore several query selection strategies that evaluate bags by aggregating instance-level uncertainties. The first such method is **combined bag uncertainty (CBU)**:

$$\phi^{CBU}(\mathcal{X}) = \text{softmax}_\alpha \left(\phi^{IU}(x_1), \ldots, \phi^{IU}(x_N)\right),$$

where $\phi^{IU}(x_n)$ is the instance-level uncertainty for each constituent instance in $\mathcal{X} = \{x_1, \ldots, x_N\}$, given below by Equation 5.1 in Section 5.3.2. Here, the informativeness of a bag is evaluated by using the model's combining function to aggregate instance-level uncertainties. Two other aggregate uncertainty strategies I consider are **maximum bag uncertainty (MBU)**:

$$\phi^{MBU}(\mathcal{X}) = \max_{n=1}^{N} \phi^{IU}(x_n),$$

which is similar, but uses the hard max function, and **total bag uncertainty (TBU)**:

$$\phi^{TBU}(\mathcal{X}) = \sum_{n=1}^{N} \phi^{IU}(x_n).$$

In addition to these uncertainty sampling approaches, I also examine the "expected gradient length" (EGL) framework (introduced in Section 7.3.3), which considers the informativeness of

a query to be the amount of change it would impart on the current model *if we knew its label*. Since we train MILR via gradient optimization (2.5), this involves querying the bag $\mathcal{X}$ which, if $\langle \mathcal{X}, y \rangle$ is added to the training set $\mathcal{L}$, would create the greatest expected change in the gradient of the objective function (i.e., the highest-magnitude gradient vector used to re-estimate values for θ). Let $\nabla \ell(\mathcal{L}; \theta)$ be the gradient of the objective function ℓ with respect to θ; it is a vector whose components are the partial derivatives of ℓ with respect to each model parameter: $\nabla \ell(\mathcal{L}; \theta) = [\frac{\partial \ell}{\partial \theta_0}, \ldots, \frac{\partial \ell}{\partial \theta_K}]$.

Now, let $\nabla \ell(\mathcal{L} \cup \langle \mathcal{X}, 1 \rangle; \theta)$ be the new gradient obtained by adding the positive tuple $\langle \mathcal{X}, 1 \rangle$ to the training set, and likewise let $\nabla \ell(\mathcal{L} \cup \langle \mathcal{X}, 0 \rangle; \theta)$ be the new gradient if a query results in the negative tuple $\langle \mathcal{X}, 0 \rangle$ being added. Since we do not know which label the oracle will provide in advance, we instead calculate the *expected* length of the gradient based on the learner's current estimate o of the outcome. More precisely, let the expected **bag gradient length (BGL)** criterion be defined as:

$$\phi^{BGL}(\mathcal{X}) = o \Big\| \nabla \ell(\mathcal{L} \cup \langle \mathcal{X}, 1 \rangle; \theta) \Big\| + (1 - o) \Big\| \nabla \ell(\mathcal{L} \cup \langle \mathcal{X}, 0 \rangle; \theta) \Big\|.$$

Note that, since the model converged on its current parameters θ after the previous round of training, $\|\nabla \ell(\mathcal{L}, \theta)\| \approx 0$. That is to say, the length of the gradient using $\mathcal{L}$ with no new bags added should be nearly zero. Because the model assumes that labeled bags are independent, $\nabla \ell(\mathcal{L} \cup \langle \mathcal{X}, y \rangle, \theta) \approx \nabla \ell(\langle \mathcal{X}, y \rangle, \theta)$. In other words, the expected gradients above only need to be calculated for the new bag $\mathcal{X}$ currently being evaluated for querying. This approach simplifies the implementation and reduces any unnecessary computation.

The formulation of ϕ^{BGL} above is only appropriate for the Scenario I, in which unlabeled bags are queried and bag labels are provided by the oracle. For the Scenario IV, in which positive bags are queried and labels are obtained for all of its constituent instances, I instead use the **total bag gradient length (TBGL)** strategy:

$$\phi^{TBGL}(\mathcal{X}) = \sum_{n=1}^{N} \phi^{IGL}(x_n),$$

which is the sum of expected gradient lengths $\phi^{IGL}(x_n)$ for each instance $x_n \in \mathcal{X}$, given below by Equation 5.3 in Section 5.3.2. This is an approximation to the true expected gradient length for this scenario, which would otherwise require an expectation over all possible permutations of the potential instance labels. Because the model assumes that instance-based singleton bags are independent, this is a reasonable approximation. Observe that when computing these expected gradients, each singleton bag is given weight $\frac{1}{N+1}$, as explained in Section 5.2. This is because ϕ^{TBGL} strictly expects that all the instances in the query bag will be labeled (i.e., $N = Q$). I do not explore any approaches based on EGL for Scenario III, in which a positive bag is queried but a label is obtained only for one of its positive instances. This is because EGL computes an expectation over possible labels it explicitly expects; however in Scenario III the query and label granularities are not the same. Furthermore, the BGL and TBGL formulations do not work well for Scenario III in my experience.

5.3.2 Querying Instances

Now let us turn our attention to strategies that select unlabeled query instances directly from positive bags. The methods in this section are appropriate for use in the MI active learning Scenario II from Table 5.1. Returning to the uncertainty sampling framework, one approach is to query instances based on the aforementioned **instance uncertainty (IU)**:

$$\phi^{IU}(x_n) = -\big(o_n \log o_n + (1 - o_n) \log(1 - o_n)\big), \tag{5.1}$$

where again entropy is used to measure uncertainty.

I argue that when querying for instance labels in a multiple-instance setting, the selection criterion should take into account not just uncertainty about a given instance's class label, but also the extent to which the learner can adequately "explain" the bag to which the instance belongs. For example, the instance that the learner finds most *uncertain* may belong to the same bag as the instance it finds most *positive*. In this case, the bag will have a high probability $P(y = 1|\mathcal{X}; \theta)$ of being positive, because the combining function output will be dominated by the positive-looking instance. Thus, naïvely querying the most uncertain instance may only have a small impact on the model overall. To address this shortcoming, I propose an uncertainty-based query strategy that weights the uncertainty of x_n in terms of how much it contributes to the classification of its parent bag $\mathcal{X}$. I call this approach **multiple-instance uncertainty (MIU)**:

$$\phi^{MIU}(x_n) = \phi^{IU}(x_n) \times \left(\frac{\partial o}{\partial o_n}\right)^{\beta}, \tag{5.2}$$

which is the instance uncertainty (5.1) times a "relevance" term, which is the derivative of the bag output o with respect to the instance output o_n (i.e., the derivative of the softmax combining function, which is given in the Appendix). The parameter β controls the relative importance of the relevance term (e.g., if $\beta = 1$ both terms have equal weight). This criterion encourages the learner to query instances that are both uncertain and influential in the labeling of their respective parent bags. Note that this approach is reminiscent of the information density algorithm (Chapters 3 and 4), except that here we are balancing the uncertainty of an instance with its influence over the bag (rather than similarity to other instances).

I also consider a query strategy for instances based on expected gradient length, which is called **instance gradient length (IGL)**:

$$\phi^{IGL}(x_n) = o_n \Big\| \nabla \ell(\mathcal{L} \cup \langle \{x_n\}, 1 \rangle; \theta) \Big\| + (1 - o_n) \Big\| \nabla \ell(\mathcal{L} \cup \langle \{x_n\}, 0 \rangle; \theta) \Big\|. \tag{5.3}$$

For efficiency reasons, the gradient here again only needs to be calculated over the expectation of the label for $\{x_n\}$. Also, each instance gradient is given weight $\frac{1}{Q+2}$, where Q is the number of instances in the same parent bag that have already been queried.

5.4 Multiple-Instance Data Sets

In Scenario I, only bag labels are necessary because instance labels are never queried. Therefore, we can evaluate these active learning strategies using several benchmark MI data sets. Drug

activity prediction was the original motivating application for the MI representation, so we begin with the **MUSK** data set from Dietterich et al. (1997). For this task, a bag represents a molecule which can assume multiple *conformations* (three-dimensional shapes) in solution. These conformations are described by various positional features, and are represented as instances. A molecule (bag) is positive if at least one of its conformations (instances) binds to a target of interest, and negative if none of its conformations bind. There are two versions of the data set which contain approximately the same number of bags, but the second version contains many more instances per bag. Active learning via Scenario I for drug activity prediction can be thought of as an automated experimental selection methodology. The model attempts to learn how a molecule (or one of its conformations) binds to a target by conducting chemical binding affinity experiments, with the goal of minimizing the number of molecules to analyze. A related idea has been explored for active learning with inductive logic programming to autonomously discover metabolic pathways in yeast, with the goal of minimizing the cost of laboratory materials (King et al., 2004). Note that it is probably not possible to chemically determine *which* conformations (instances) bind to a target. Therefore, drug activity prediction is not a good candidate for MI active learning via Scenarios II through IV.

Protein family modeling has also been framed as a multiple instance problem. I explore MI active learning in this application using the **TrX** data set (Tao et al., 2004), where the task is to classify given protein sequences according to whether they belong to the family of TrX proteins. The proteins are first aligned with respect to a motif that is known to be conserved in members of the family. A bag then represents an aligned protein, and an instance corresponds to a position in a fixed-length sequence around the conserved motif. Each instance is described by properties of the amino acid at that position, and smoothed using the same properties from its 16 neighbors. A protein (bag) is positive if it belongs to the family, and negative otherwise. Labeling instances for this application may be possible, though extremely expensive. Therefore, as with drug activity prediction, Scenario I is the most appropriate MI active learning setting for this task.

I also investigate text classification with the **TST-OHSUMED** corpus, prepared by Andrews et al. (2003) and derived from the OHSUMED corpus (Hersh et al., 1994). This data set comprises biomedical article abstracts, and is divided into seven subtasks. Each subtask involves assigning abstracts to a specific MeSH (Medical Subject Heading) term. Similar to the MI text classification setting discussed in Section 5.1, a bag represents a document and instances are passages of text. More specifically, in this corpus, instances are overlapping windows of 50 words each. A document (bag) is positive if at least one of its passages (instances) asserts that the MeSH term is relevant. Instances are characterized using a bag-of-words feature representation. Each subtask contains 200 positive and 200 negative bags, which have about 16 instances per bag on average.

For the content-based image retrieval task I use the **SIVAL** repository, which is a collection of 1500 images, each labeled with one of 25 class labels. The images contain complex objects photographed in a variety of positions, orientations, locations, and lighting conditions. As explained in Section 2.2.2, in this setting a bag represents an image, which has been transformed and segmented into approximately 30 segments. An image (bag) is positive if at least one of its

segments (instances) belongs to an object of interest. Each instance segment is described by a 30-dimensional feature vector describing color and texture attributes of the segment and its cardinal neighbors. For more details, see Rahmani and Goldman (2006).

Conducting experiments for active learning Scenarios II, III, and IV requires data sets that have instance-level labels available. Since no MI data sets with instance-level labels previously existed, I created several, including an augmented version of the **SIVAL** repository. Using a web-based graphical interface I developed, a few members of my research group helped manually annotate all positive instance segments belonging to the labeled object of interest for each image. We used this system to annotate all 1500 images in the repository. More details on the annotation methodology for this data set are provided in Chapter 6 (Section 6.2.2)

For text classification, I created a semi-synthetic MI data set using the **20 Newsgroups** corpus as a base. This corpus was chosen because it is an established benchmark for text classification, and because the source documents—Usenet posts from the early 1990s—are relatively short (in the MI setting, instances are usually paragraphs or short passages). For each of the 20 news categories, I generate artificial bags of approximately 50 posts (instances) each by randomly sampling from the target class (i.e., newsgroup category) at a rate of up to 10% for positive bags, with remaining instances (and all instances for negative bags) drawn uniformly from the other 19 classes. The documents are processed by first stripping out all headers (e.g., from and subject lines), email addresses (that match a regular expression), and reply quotes (i.e., lines that begin with ">," or ":" characters). This step helps prevent content in one message from bleeding into other messages which may quote it verbatim. The resulting documents are then processed with stemming, stop-word filtering, and term-frequency ranked feature selection. Frequency counts for the 500 most common words are used as features to represent the instance documents. I construct a data set of 100 bags (50 positives and 50 negatives) for each of the 20 label classes. The corpus described here differs slightly from the semi-synthetic corpus used in previous work (Settles et al., 2008b). This is because I found the previous version to be trivial for MILR after switching to the log-likelihood objective function (see Section 2.2.2). This new corpus is more difficult and, I feel, somewhat more realistic.

I also created a semi-synthetic MI data set using the **Handwritten Digits** collection (Hull, 1994) as a base. This data set contains thousands of handwritten digits that belong to ten label classes (zero through nine), each represented as 8×8 pixel images (resulting in a 64-feature vector). To treat this as an MI learning problem, I generate random integers of up to 15 digits in length. A bag, then, is one of these randomly generated integers, and an instance is a randomly chosen image from the collection corresponding to a digit position in the integer. An integer (bag) is positive if it contains at least one digit (instance) of the target label class (e.g., zero), and is negative otherwise. This results in ten subtasks, one for each digit, for which I generate 5,000 total bags each.

5.5 Experiments

This section presents experimental results using the various query strategies from Section 5.3, using MILR as the base learning algorithm, for each of the four active learning scenarios. Methods are evaluated by constructing learning curves that plot the area under the ROC curve, or AUROC (as defined in Section 2.3) as a function of the number of queries made for each data set and query selection strategy. In all experiments, I use $\alpha = 3$ for the softmax combining function following previous work (Ray and Craven, 2005), and because this value seems to provide a good empirical balance between the "average" and "hard max" extremes. Further experimental details for each scenario are given the subsections below.

5.5.1 Scenario I: Label a Query Bag

The goal of these experiments is to determine how MILR and related models can best select unlabeled bags for querying. This setting is analogous to standard active learning, in that labels are not acquired at mixed levels of granularity. In these experiments, results are averaged over over 20 independent runs for which the data set is randomly split into two halves: one for the pool of unlabeled bags $\mathcal{U}$, and the other held aside for evaluation. For each run, four bags (two positive and two negative) are randomly drawn from $\mathcal{U}$ to comprise the initial labeled set $\mathcal{L}$. Randomization is done using seed numbers to ensure that all query strategies receive the same initial partitioning of the data for each run. For each query strategy, bags are queried one at a time from $\mathcal{U}$ to be labeled and added to $\mathcal{L}$, the model is re-trained, and the process repeats. I compare the active learning algorithms presented in Section 5.3.1 against a simple baseline that randomly chooses an unlabeled bag from $\mathcal{U}$.

Figure 5.2 presents selected learning curves for for each data set. For legibility, the figures only show the random baseline and three of the five query strategies: BU, CBU, and BGL (the other approaches all aggregate instance-level uncertainty, and either perform poorly or very similarly to CBU). Since the TST-OHSUMED, SIVAL, and Handwritten Digits collections are made up of many subtasks, I present plots for only three representative subtasks each. Table 5.2 summarizes the curves for these larger data sets by reporting the average improvement across all subtasks made by each strategy over the initial model (i.e., before any queries are made). Curves for 20 Newsgroups are omitted because they are flat and uninformative, but the results are summarized in Table 5.2.

For the MUSK, TST-OHSUMED, and 20 Newsgroups tasks, active learning does not appear to be helpful under these experimental conditions. While accuracy does generally improve as new labeled bags are added to the training set, the active query strategies are often on par with (or even inferior to) the random baseline. The 20 Newsgroups tasks seem particularly difficult to learn in general. For the TrX data set, the BU and BGL strategies show some improvement over random, but the differences are not statistically significant. With the SIVAL and Handwritten Digits tasks, however, active learning does appear to help fairly consistently, mainly using the bag uncertainty (BU) strategy. The AUROC gains over random for several of these subtasks are statistically significant as well (for the majority of sample sizes along the learning curve, using

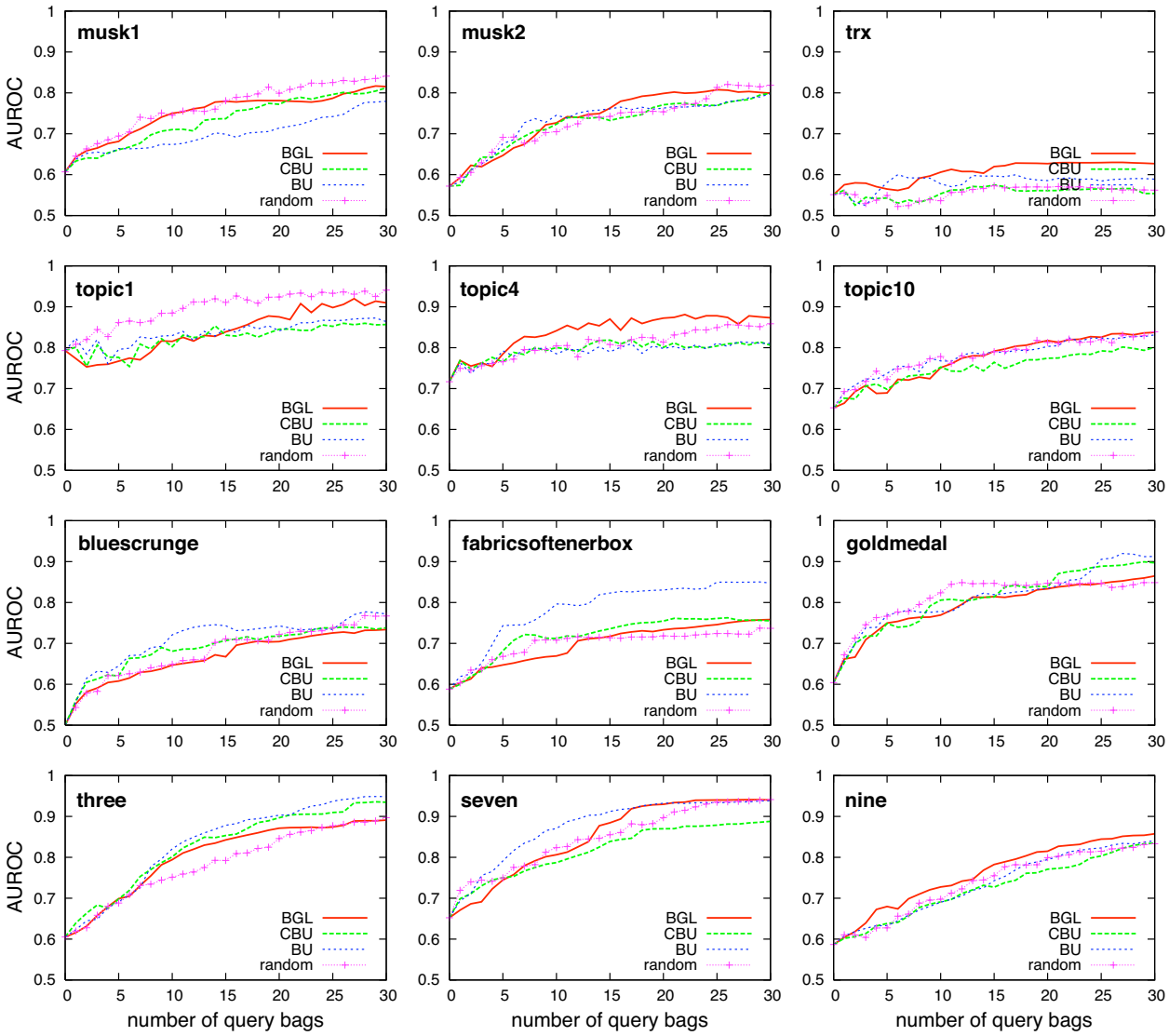


Figure 5.2: Selected learning curves for Scenario I: MUSK and TrX data sets (top row), TST-OHSUMED (second row), SIVAL (third row), and Handwritten Digits (bottom row).

a two-tailed t-test at 95%). The BU strategy is consistently better than the other bag-level query algorithms for this scenario, indicating that it is better to consider the informativeness of the bag as a whole rather than aggregating instance-level information for this active learning setting.

Why active learning under Scenario I should be helpful for certain tasks and not others might have something to do with the underlying ratio of positive instances in positive bags. We know that positive images from the SIVAL repository contain 15–28% positive segments, and integer-bags from the Handwritten Digits collection contain about 16% positive digits. These high figures

Table 5.2: Summary of Scenario I learning curves for large data sets. Reported is the average improvement in AUROC over the initial model (i.e., before any queries are made) for each query strategy. Numbers are averaged across all subtasks for each data set at various points during active learning, and the algorithm with the highest average improvement at a given point is shown in bold. Small numbers in parentheses represent the number of subtasks for which the strategy was the "winner" at that point in the curve.

Queries	Random	BU	CBU	MBU	TBU	BGL
			TST-OHSUMED			
5	+0.062 (2)	+0.047 (1)	+0.034	+0.039 (1)	**+0.066** (3)	+0.033
10	**+0.093** (1)	+0.066 (1)	+0.071	+0.069 (1)	+0.087 (2)	+0.086 (2)
20	+0.116 (2)	+0.089	+0.088	+0.112 (2)	+0.116 (1)	**+0.126** (2)
40	+0.164 (3)	+0.126	+0.128	+0.139	+0.143	**+0.166** (4)
			SIVAL			
5	+0.054 (5)	**+0.064** (3)	+0.057 (4)	+0.058 (3)	+0.056 (5)	+0.051 (5)
10	+0.070 (3)	**+0.082** (5)	+0.076 (6)	+0.071 (1)	+0.069 (4)	+0.067 (6)
20	+0.085 (4)	**+0.099** (4)	+0.096 (4)	+0.087 (1)	+0.091 (4)	+0.091 (8)
40	+0.098 (2)	**+0.125** (9)	+0.119 (2)	+0.113	+0.118 (6)	+0.113 (6)
			Handwritten Digits			
5	+0.084 (1)	+0.090 (3)	+0.096 (2)	+0.076 (1)	+0.069	**+0.097** (3)
10	+0.143 (1)	**+0.169** (5)	+0.160 (2)	+0.136	+0.119	+0.159 (2)
20	+0.219 (1)	**+0.251** (5)	+0.237 (2)	+0.198	+0.190	+0.234 (2)
40	+0.286 (1)	**+0.297** (2)	+0.288 (2)	+0.273 (1)	+0.260 (1)	+0.279 (3)
			20 Newsgroups			
5	–0.009 (1)	–0.002 (5)	**+0.000** (6)	–0.004 (1)	–0.012 (2)	–0.003 (5)
10	**+0.004** (5)	–0.002 (2)	**+0.004** (4)	–0.003 (2)	–0.009 (2)	–0.002 (5)
20	+0.009 (4)	–0.001 (3)	**+0.013** (7)	+0.004 (1)	–0.009 (2)	+0.002 (3)
40	**+0.021** (4)	+0.014 (2)	**+0.021** (8)	+0.014	+0.018 (5)	+0.010 (1)

correlate with with their higher gains during active learning. Bags in the 20 Newsgroups corpus, however, are more sparsely positive at 10% or less. While we know nothing about the abundance of positive instances in the MUSK, TrX, and TST-OHSUMED data sets, it is possible that positive instances in these problem domains are also rare. In previous work with MUSK and TrX (Ray and Craven, 2005), models with strong assumptions of a single positive instance performed well, suggesting that this is indeed the case. It may also be that the ratio of positive *bags* in the overall data set influences the utility of active learning. The 20 Newsgroups, TST-OHSUMED and MUSK1 data sets are fairly balanced, while the SIVAL (4% positive) and Handwritten Digits (61%) are more skewed. However, MUSK (38%) and TrX (13%) are also skewed and show little improvement with active learning. It is more likely, then, that the active learning algorithms I consider for this scenario are only useful when the ratio of positive instances in positive bags is relatively high.

One possible explanation stems from the use of the combining function. If a bag contains many instances, only a few of which are positive, the softmax function may underestimate the overall bag-level output. If this is the case, the bags with the highest posteriors may in fact result in the highest entropy values (which may not be the desired effect). To study this, I have experimented with other query strategies that measure uncertainty as proximity to a dynamic threshold, as estimated by the mean or median of the bag label posteriors in $\mathcal{U}$. However, these approaches perform no better in practice. It is possible that more carefully tuning the α parameter might resolve some of these issues for highly ambiguous tasks, however, my preliminary investigation of this yielded no improvements and I have not explored it any further.

I feel that these results underscore the rationale for exploring the other three active learning scenarios, examined in the following subsections. For problem domains in which instance-level labels can be reasonably acquired, queries that obtain these ambiguous instance labels directly may lead to more substantial gains in accuracy while keeping labeling costs low.

5.5.2 Scenario II: Label a Query Instance from a Positive Bag

The goal of these experiments is to assess whether instance-level queries are of value to MI learning algorithms, and how to best select them. In these experiments, results are again averaged over 20 independent runs; I draw 20 bags (ten positive and ten negative) randomly from the data set to comprise $\mathcal{L}$, and hold aside all remaining bags for evaluation. This is because, in this active learning setting, the learner queries unlabeled *instances* in positive bags and acquires labels for these instances. Thus the active learning pool is actually comprised of instances from the bags in $\mathcal{L}$. Using the query strategies from Section 5.3.2, the unlabeled instances are queried directly, labeled, and added as singleton bags to $\mathcal{L}$. The model is re-trained with these mixed-granularity labels using the method described in Section 5.2, and the active learning process repeats. For the MIU strategy, I use $\beta = 1$ (i.e., both the uncertainty and relevance terms have equal weight). I compare the three active query approaches to a baseline that chooses an unlabeled instance at random from any positively labeled bag.

Figure 5.3 presents selected learning curves for the three data sets with instance labels available: SIVAL, 20 Newsgroups, and Handwritten Digits. Notice that the horizontal axis is now plotted in terms of query instances rather than bags. Table 5.3 summarizes learning curves across all subtasks for each data set. Note that the active query strategies reported here are different than with Scenario I, because we are now interested in selecting instance queries directly.

We can draw several interesting conclusions from these results. First and most germane to this MI active learning setting, is that MI learners benefit from instance-level labels. Note that, unlike Scenario I, no new labeled *bags* are added to the training set; each query instead helps by disambiguating which *instances* in the positively labeled bags are truly positive. With the exception of random selection on the 20 Newsgroups data, these instance-level labels almost always improve the accuracy of the learner, often with statistical significance after only a few queries.

Second, we see that all three active query strategies (IU, MIU, and IGL) perform better than passive (random) instance labeling. On SIVAL tasks, random querying steadily improves accuracy, but very slowly. As Table 5.3 shows, random selection at 40 queries is roughly comparable to the

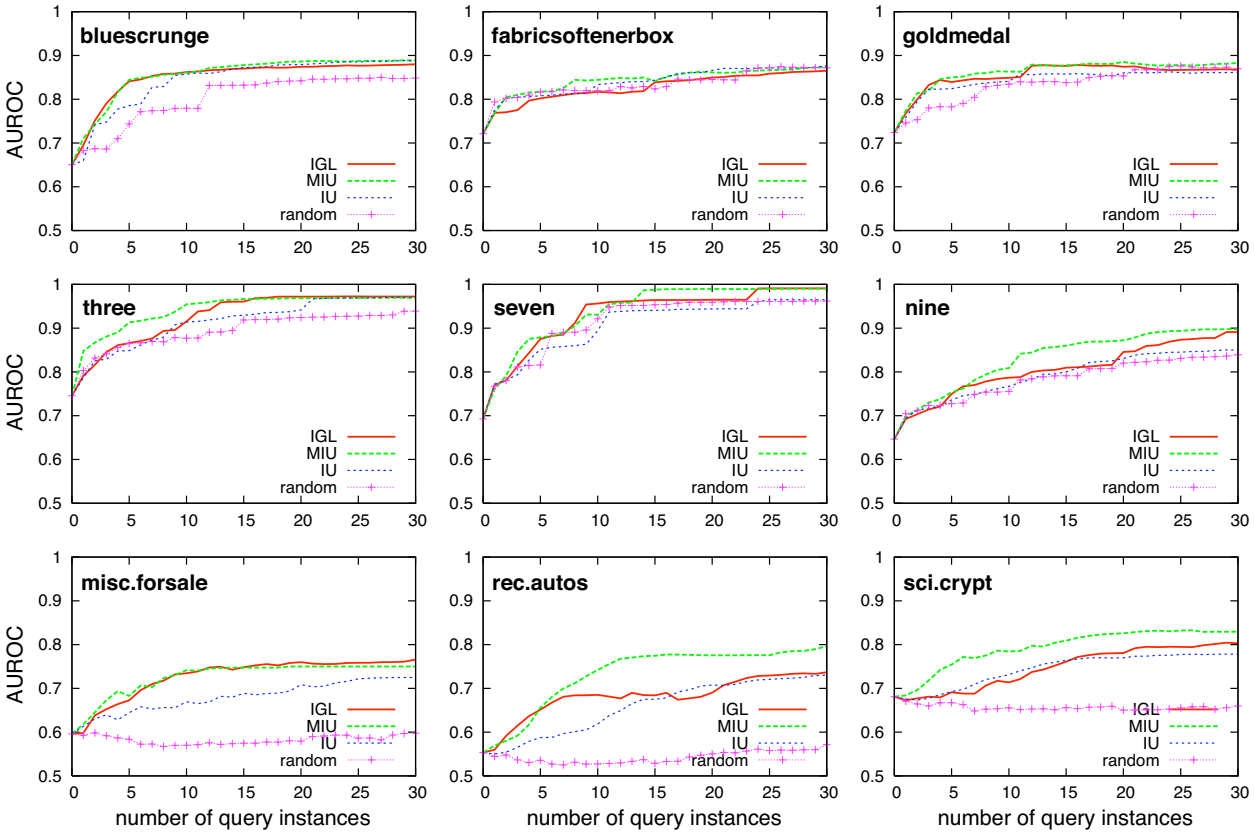


Figure 5.3: Selected learning curves for Scenario II: SIVAL (top row), Handwritten Digits (second row), and 20 Newsgroups (bottom row).

active query strategies after a quarter as many queries. On the 20 Newsgroups tasks, random selection has a slight negative effect (if any) early on, possibly because it lacks a focused search for the positive and/or ambiguous instances (of which there are only a few per bag). All three active selection methods, on the other hand, show significant gains fairly quickly on all data sets. This is particularly important for the 20 Newsgroups tasks, for which we know that positive instances are sparse. Unlike Scenario I, which queries unlabeled bags, these results demonstrate that active learning can help even in these highly ambiguous domains, as long as the learner is allowed to query for specific instance labels.

Finally, MIU appears to be a particularly well-suited query strategy for Scenario II. On all data sets, it consistently improves the initial MI learner, usually with statistical significance, and often approaches the asymptotic level of accuracy with fewer labeled instances than the other two active methods (sometimes in as few as ten query instances). IGL in turn outperforms the IU strategy in most cases. It is true that MIU's gains over IGL are not usually statistically significant, and in the long run it is generally matched or slightly surpassed it. However, MIU shows the greatest

Table 5.3: Summary of Scenario II learning curves. See Table 5.2 for an explanation of notation.

Queries	Random	IU	MIU	IGL
		SIVAL		
5	+0.056 (1)	+0.072 (8)	**+0.079** (12)	+0.073 (4)
10	+0.070 (2)	+0.090 (4)	**+0.096** (12)	+0.090 (7)
20	+0.083 (1)	+0.105 (7)	+0.106 (9)	**+0.107** (8)
40	+0.090 (3)	+0.117 (9)	**+0.118** (7)	**+0.118** (6)
		Handwritten Digits		
5	+0.087 (2)	+0.088 (1)	**+0.102** (5)	+0.098 (2)
10	+0.113 (1)	+0.121 (1)	**+0.140** (5)	+0.130 (3)
20	+0.141	+0.149 (1)	**+0.176** (6)	+0.160 (3)
40	+0.160	+0.176 (2)	**+0.184** (3)	+0.181 (5)
		20 Newsgroups		
5	–0.010	+0.014	**+0.043** (14)	+0.032 (6)
10	–0.011	+0.035 (1)	**+0.085** (16)	+0.051 (3)
20	–0.007	+0.071 (4)	**+0.112** (14)	+0.091 (2)
40	+0.007 (1)	+0.097 (1)	+0.116 (6)	**+0.130** (12)

advantage early in the active learning, perhaps because it is the only method tested that explicitly encodes the MI assumption by taking advantage of the combining function in its selection process.

5.5.3 Scenario III: Label Any Positive Instance from a Positive Query Bag

The goal of these experiments is to evaluate the utility of querying a bag and obtaining the label of any positive instance in that bag (which may be more natural in some domains). The initial partitioning of the data is the same as in Scenario II. However, in this setting the learner queries a labeled positive bag, and acquires an instance label for a randomly chosen positive instance from that bag. The labeled instance is then added as a singleton bag to $\mathcal{L}$. The instance is chosen at random because, from the point of view of the annotator, we can assume that any labeled positive instance is as reasonable as any other from the same bag. To account for the randomness in which instance is actually labeled, however, I replicate each of the 20 independent runs 20 times (thus, results are averaged over 400 runs). Once all the positive instances in a bag have been queried, the parent bag is marked such that it will not be queried again (this simulates an annotator flagging, for example, an image for which all the positive segments have now been labeled). I compare the bag-level active query strategies (Section 5.3.1) to a baseline that chooses a positive query bag at random.

Figure 5.4 presents selected learning curves for this scenario. Note that, since the active learner does not know which instance will actually be labeled and added to the training set, I do not consider an "expected gradient length" query strategy variant, which inherently assumes that the

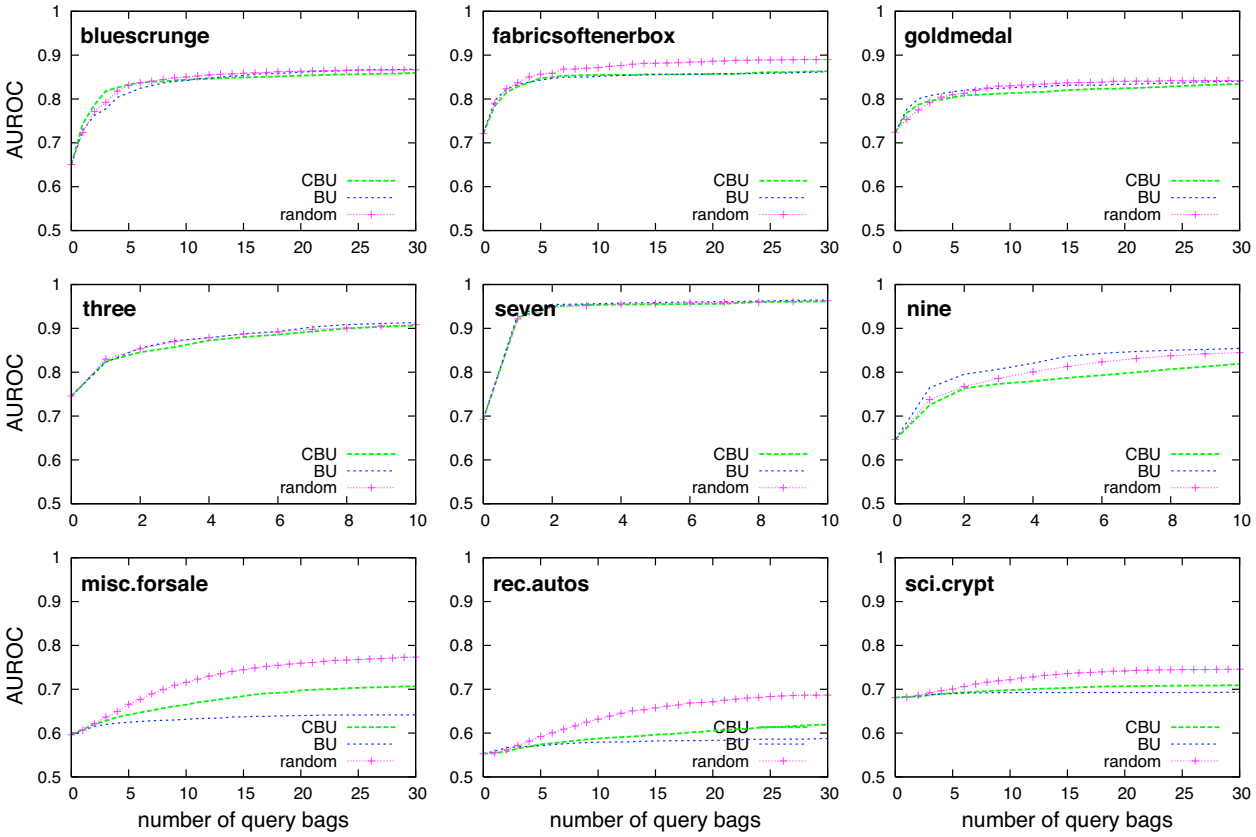


Figure 5.4: Selected learning curves for Scenario III.

instance to be labeled is known. Note also that the horizontal axis is again plotted in terms of query bags (however, only one instance is labeled per query). Table 5.4 summarizes learning curves across all subtasks. Since queries can no longer be made once all positive instances in $\mathcal{L}$ are labeled, some data sets have fewer possible queries. As a result, Handwritten Digits experiments only run out to ten queries, and 20 Newsgroups experiments run out to 30 queries.

The first thing to note from these results is that learning with positive instance labels still shows improvement over the base MI model for all data sets. This is again even true for the 20 Newsgroups corpus. For some tasks, the learner even approaches its asymptotic accuracy after seeing only five to ten positive instances (and merely one instance for the seven subtask).

However, unlike Scenario II, the active query strategies perform no better than random selection in this setting. In fact, for most tasks they are somewhat worse. While all methods generally improve the base MI learner, querying bags randomly does so with fewer labeled instances. I surmise that this is because the active strategies implicitly expect to obtain labels for the instances that most influence the uncertainty of the parent bag's labeling. However, there is no guarantee as to exactly which positive instance will be labeled and added to $\mathcal{L}$, thus the attempts of these active strategies are ultimately thwarted. Furthermore, the most informative instance may in fact be negative.

Table 5.4: Summary of Scenario III learning curves. See Table 5.2 for an explanation of notation.

Queries	Random	BU	CBU	MBU	TBU
			SIVAL		
5	**+0.078** (11)	+0.072 (3)	+0.072 (5)	+0.072 (2)	+0.071 (4)
10	**+0.091** (16)	+0.084 (3)	+0.083 (5)	+0.083	+0.082 (1)
20	**+0.098** (18)	+0.092 (3)	+0.091 (3)	+0.092 (1)	+0.091
40	**+0.101** (16)	+0.095 (4)	+0.096 (3)	+0.097 (1)	+0.095 (1)
			Handwritten Digits		
5	+0.134 (1)	**+0.139** (8)	+0.126	+0.124	+0.125 (1)
10	**+0.150** (3)	**+0.150** (4)	+0.143	+0.142	+0.144 (3)
			20 Newsgroups		
5	**+0.024** (19)	+0.010	+0.014 (1)	+0.013	+0.009
10	**+0.049** (20)	+0.013	+0.024	+0.017	+0.012
20	**+0.081** (20)	+0.015	+0.039	+0.020	+0.015

We also see that the gains under this scenario are not as pronounced or as rapid as they were with Scenario II. We can conclude from these results that there is value in having the learner choose the specific instance to be labeled directly. It also appears advantageous to obtain both positive and negative instance labels, despite the fact that positives are less abundant (and thus intuitively more important). This is possible with Scenario II but not with Scenario III.

5.5.4 Scenario IV: Label All Instances in a Positive Query Bag

The goal of these experiments is to evaluate the benefit of obtaining labels for every instance in an informative query bag. The setup for these experiments is similar to Scenario III; the only difference is that once a bag is queried, all instances (both positive and negative) are labeled. Each time a bag is queried, it is marked as such—so that it will not be queried again—and all of its instances are labeled and added as singleton bags to $\mathcal{L}$. The model is re-trained with these mixed-granularity labels and the active learning process repeats. I again compare the active query strategies to a baseline that randomly chooses a positive bag.

Figure 5.5 presents selected learning curves for this scenario. Notice that the horizontal axis in these plots only goes to ten, since $\mathcal{L}$ only contains ten positive bags. Thus, the rightmost point in each plot means that all instances in the training set have been fully labeled. The reason the algorithms to not always converge to the exact same AUROC value is likely due to local optima. Table 5.5 summarizes learning curves across all subtasks for each data set.

We can see that fully labeling all instances from positive bags can improve accuracy. However, there are several other interesting observations we can make with these results. For one thing, once again the active strategies appear to show no consistent gains over the random baseline. While labeling instances appears to be generally valuable, the proper way to choose which bags should

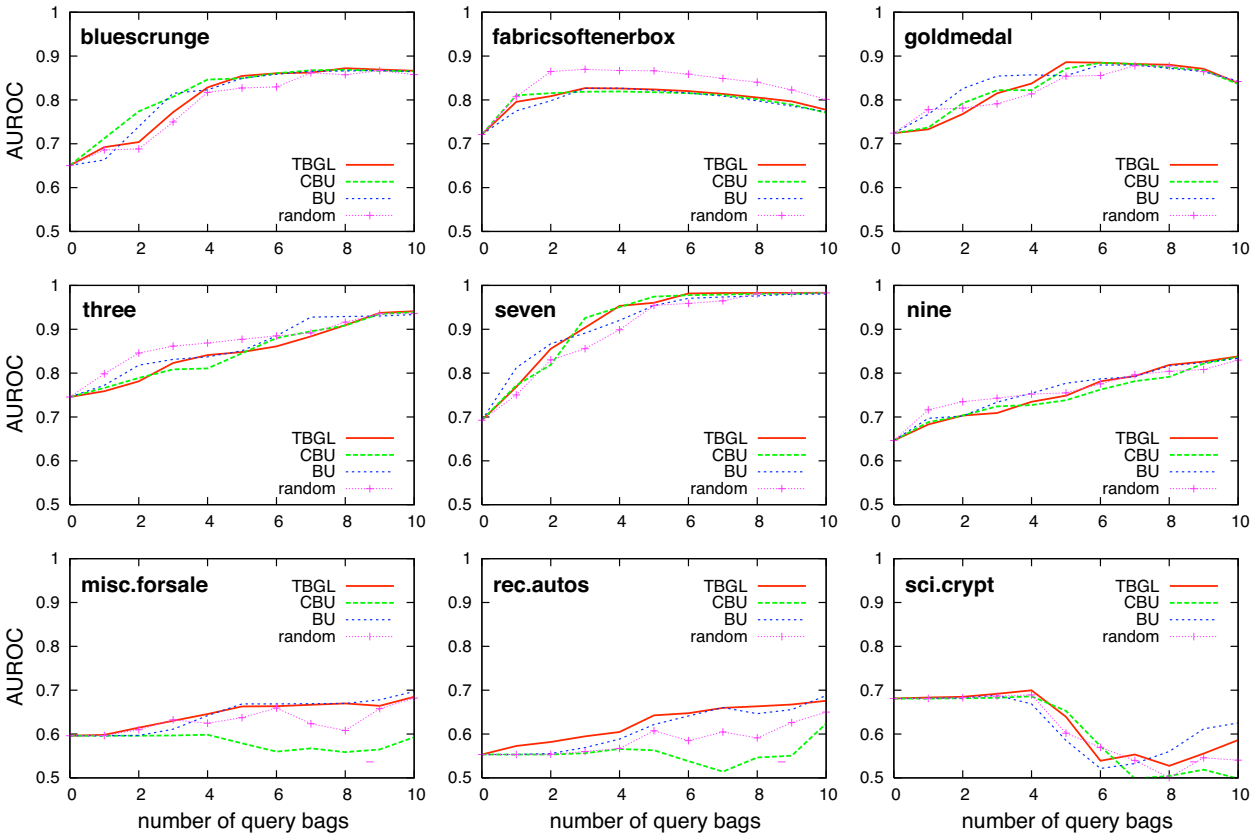


Figure 5.5: Selected learning curves for Scenario IV.

Table 5.5: Summary of Scenario IV learning curves. See Table 5.2 for an explanation of notation.

Queries	Random	BU	CBU	MBU	TBU	TBGL
			SIVAL			
5	+0.081 (4)	+0.074 (3)	**+0.088** (6)	+0.085 (2)	+0.086 (5)	+0.086 (5)
10	**+0.062** (5)	+0.057 (6)	+0.062 (4)	+0.062 (3)	+0.061 (5)	+0.059 (2)
			Handwritten Digits			
5	+0.112 (2)	**+0.114** (3)	+0.108 (1)	+0.105 (1)	+0.098	+0.110 (3)
10	+0.158 (1)	+0.157 (1)	+0.159	+0.159 (1)	+0.159 (3)	**+0.160** (4)
			20 Newsgroups			
5	+0.001 (2)	+0.006 (7)	–0.010 (2)	+0.004 (3)	–0.012 (1)	**+0.010** (5)
10	+0.021 (3)	**+0.045** (9)	–0.010	+0.018 (2)	+0.025 (2)	+0.030 (4)

be fully labeled is unclear. This may be an artifact of the experimental setup, because $\mathcal{L}$ (which comprises the query pool in this scenario) is so small. However, in many real-world applications of MI active learning there may only be a handful of labeled bags available.

Another interesting result is that, for a handful of SIVAL subtasks, accuracy increases with the first several queries but decreases again once nearly all bags have been queried. When the annotators were manually labeling instances for this data set, we noticed that some of the image segments did not clearly belong to the target object (e.g., part of the object and its shadow were lumped together into a single segment). As a result, different annotators may have judged these borderline segments differently, creating a certain amount of noise in the instance labels. I conjecture that most of the observed decrease in accuracy is due to the learner trying to model this noise.

A final observation is that, for a few curves in the 20 Newsgroups subtasks, model accuracy actually drops after four or five queries, and in rare cases (e.g., sci.crypt) does not recover. I suspect that this is an artifact of the modifications that are made to the learning algorithm to account for labels at mixed bag and instance granularity. If all instances in a bag are labeled, they are uniformly weighted by the training algorithm, which means that each positive receives a weight of $\approx 1/50$. Since positive instances are so sparse in this corpus, the training signal for these valuable instances might actually be diminished relative to the case where only bags (which have a weight of 1) are labeled. I have experimented with non-normalized variants of the training algorithm, but such approaches tend to severely over-fit the bags that have been queried.

I argue that one lesson to be learned from these results is that fully labeling all instances in a bag may not be appropriate for all problems. Not only is labeling all these instances the most expensive query scenario I have considered, but doing so completely as in Scenario IV might force the learning algorithm to (i) model a certain amount of inherent instance-level noise, or (ii) not sufficiently weight the valuable positive instances when they are particularly sparse.

5.5.5 Comparison of Instance-Labeling Scenarios

Now let us take a comparative look at MI active learning Scenarios II, III, and IV, all of which can obtain instance-level labels in an effort to reduce the inherent ambiguity of the MI representation. Figure 5.6 presents a comparison of the best learning curves for each of these three scenarios for selected SIVAL, Handwritten Digits, and 20 Newsgroups subtasks. Note that the horizontal axes represent the number of queries made, even though the number of actual label(s) obtained can vary, i.e., the query instance (Scenario II), any positive instance in the query bag (Scenario III), or all instances from the query bag (Scenario IV). The Scenario III and IV curves may not extend as far as Scenario II, because for some tasks there are fewer positive instances or bags available to query.

Scenarios II and III can be reasonably compared in terms of labeling cost, since they both label a single positive instance per query. There are three main observations we can make regarding the learning curves for these two approaches. First, the best query strategy for Scenario II is consistently the MIU algorithm, whereas for Scenario III, it is variously BU, CBU, or even random. Second, there is no clear "winner" between the two, although Scenario II dominates more often (particularly for the 20 Newsgroups corpus, in which positives are most sparse). Second, even

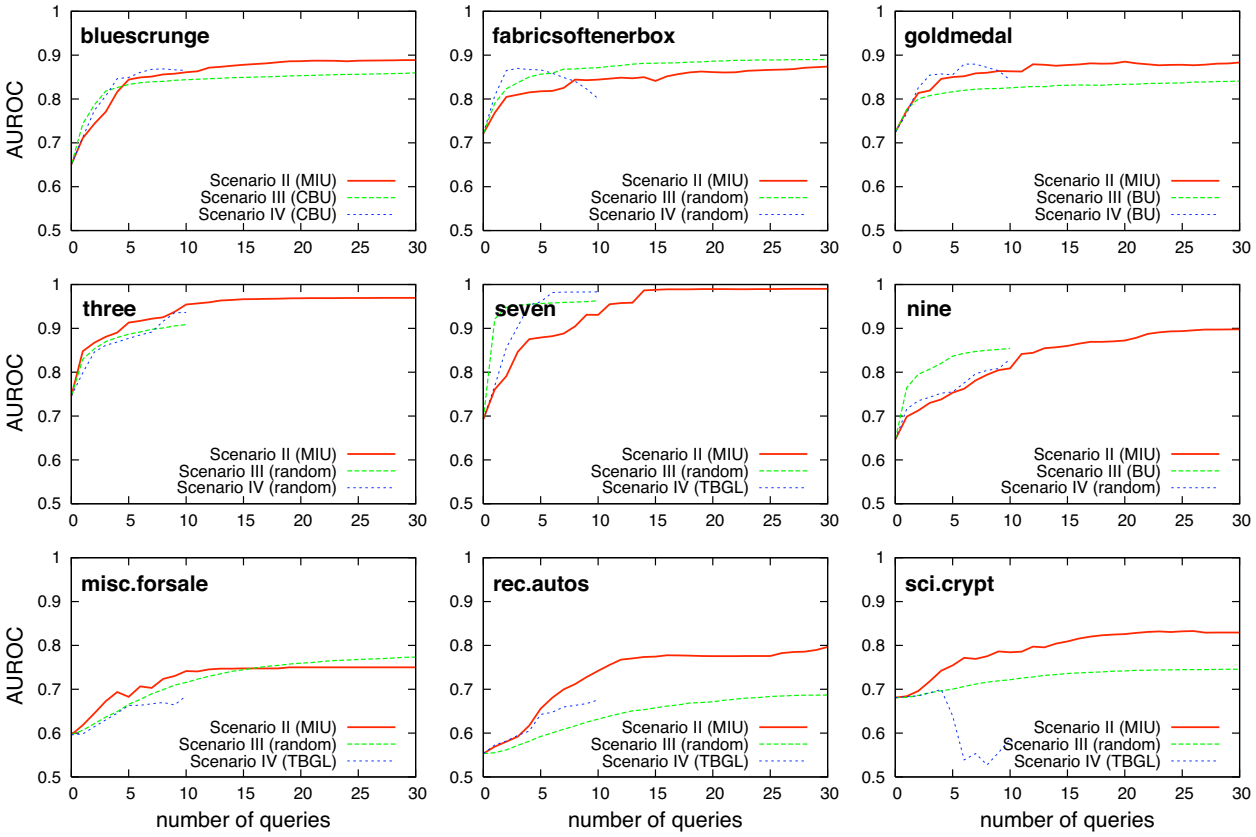


Figure 5.6: A comparison of the best learning curves for each scenario on several sample tasks.

in cases where Scenario III dominates, the curves are cut short in some domains, such as the Handwritten Digits tasks. This is because Scenario III is limited to only labeling positive instances, which may be exhausted before the learner is able to find a better set of parameters. Scenario II, however, is able to continue asking questions about informative negative instances as well, and ultimately surpass the accuracy of Scenario III approaches, e.g., for the seven and nine subtasks.

To see why querying negative instances can be important, consider Figure 5.7. This figure shows the first six queries asked by the MIU algorithm in a run of the goldmedal SIVAL task. In the first three queries, the learner requests labels for (1) a shadow near the medal, (2) a prominent part of the ribbon, and (3) the medal itself against a background which is similar in color and texture. The next three queries (and the majority thereafter) focus on the negative space near the object—such as inside a loop of the ribbon—in a wide variety of different background settings. In essence, the learner seems to have learned the general concept of a gold medal, and is now verifying that all these near-misses are indeed part of the background. By doing this, it can better avoid retrieving images from the repository that contain different objects in these same background settings.

Scenario IV is somewhat difficult to compare to the other two in terms of labeling cost, since a query in this scenario involves labeling all instances from a positive bag. However, we see that the

Figure 5.7: The first six queries in a Scenario II run of the goldmedal task, using the MIU strategy.

other scenarios are often superior even if we assume that the query types are of equal cost. In short, labeling all instances is no better (and occasionally even worse) than labeling only one instance per query. This is likely a benefit of using the MILR model, which is able to resolve the inherent ambiguity of an entire bag with only a few instance labels.

I conclude that Scenario II is the preferable MI active learning approach in cases where instance labels can be obtained. However, Scenario III has its advantages as well. In CBIR tasks, for example, there is often little difference in accuracy between the two scenarios (possibly because the ratio of positive instances is so high). It may be more natural for these tasks for a human annotator to label segments that belong to the target object, instead of labeling particularly informative segments that may or may not be positive, e.g., negative spaces in Figure 5.7. (In fact, CBIR tasks may require little or no extra effort to label a single positive segment while labeling images, thus mixed-granularity learning might also be leveraged for active learning with Scenario I.) For text domains, on the other hand, Scenario II is clearly preferable because the annotator is only required to read and label a single informative passage, rather than an entire document. Furthermore, if real-world MI text classification tasks are as sparsely positive as the 20 Newsgroups corpus here (as the TST-OHSUMED results from Section 5.5.1 seem to indicate), Scenario II is also likely to produce better learning curves, and can do so with the known superior query strategy: the MIU algorithm.

5.5.6 Active Learning with Diverse Density

I have also conducted some active learning experiments using the original formulation of Diverse Density, which couples a Gaussian instance model with a noisy-or combining function (Maron and Lozano-Perez, 1998). However, I do not report those results here. In my experience, Diverse Density is much more prone to becoming trapped in local minima than MILR, thus adding new query bags (or instances) to $\mathcal{L}$ often results in little or no change in the model or its accuracy. One can get around this problem with a "N random restarts" approach after each query is made, allowing the algorithm to better utilize the newly labeled parts of the instance space and achieve learning curves similar to MILR. However, this tactic is very slow and computationally expensive in practice. I find that MILR learns faster, and is often more accurate on these data sets than Diverse Density (a comparison is provided in the Appendix).

5.6 Summary and Future Work

In this chapter, I presented *multiple-instance active learning*, a novel framework for reducing the labeling burden by acquiring labels at a coarse granularity, and then allowing the learner to selectively query the data at potentially finer levels. This approach is useful when bag labels are easily acquired, and instance labels can be obtained but are more expensive. I have formulated, implemented, and compared query strategies designed for four different MI active learning scenarios, in which: (i) unlabeled bags may be queried, (ii) unlabeled instances in positive bags may be queried, (iii) any positive instance label may be obtained from a labeled, positive query bag, or (iv) all instance labels may be obtained from a positive query bag. Specific conclusions of this work can be summarized as follows:

- These active learning scenarios can, each in their own way, reduce the inherent ambiguity of the MI representation while keeping label costs low. Specifically, obtaining instance-level labels through active learning is beneficial in several real-world problem domains, including content-based image retrieval and text classification.

- Scenario II, in which an active learner may query specific instances in labeled positive bags, shows the most promise for learning with mixed levels of granularity. In particular, the MIU query strategy is well-suited to MI active learning. This scenario appears to be especially useful in domains where positive bags contain few positive instances.

- Scenario I, in which an active learner queries unlabeled bags from a pool, appears to be most useful in domains where positive bags contain many positive instances.

Future work in MI active learning might include investigating other ways to combine bag-level and instance-level queries for MI problem domains. One possibility requires that, when labels are obtained for positive bags, at least one positive instance label be provided as well. For many tasks such as content-based image retrieval, this extra labeling step requires little or no additional effort for the annotator, and is likely of great value to the learner. Another setting of particular interest

is where some bags are initially labeled and others are not, and the learner is allowed to query on (i) unlabeled bags, (ii) unlabeled instances from positively labeled bags, or (iii) some combination thereof. I hypothesize that this family of active learning approaches can even further reduce the labeling effort for many real-world multiple-instance learning applications. Vijayanarasimhan and Grauman (2009) have recently extended my active learning framework to support vector machines, and demonstrated that such mixed-granularity queries can be effective in content-based image retrieval tasks.

Finally, I have argued that instance labels are more expensive to acquire than bag labels. Another key direction of future work in MI active learning (and active learning in general) involves quantifying those differences in cost and evaluating the tradeoffs between these mixed granularities. In the next chapter, I consider ways of quantifying and predicting one such real-world labeling cost: annotation time.

Chapter 6

Accounting for Real-World Annotation Costs

This chapter addresses the issue of active learning with variable annotation costs. Most research in active learning to date has assumed that the cost of acquiring labels is the same for all queries. However, there are many problem domains in which labeling costs may vary, and reducing the number of labeled instances may not guarantee a reduction in cost. To better understand the nature of active learning with actual labeling costs, I present a detailed empirical study of annotation costs in four real-world domains involving human annotators. Much of this work has been previously published (Settles et al., 2008a).

6.1 Introduction

Most previous work in active learning has assumed a fixed cost for acquiring each label, i.e., all queries are equally expensive for the oracle. However, consider the information extraction tasks from Chapters 3 and 4: the sentences in these corpora can vary considerably in length and the complexity of language used. These variables most likely affect the amount of work required to label different instances. Also consider that the queries that are most valuable to the learner may be the most difficult or ambiguous cases, and therefore the most expensive for an oracle to label accurately. These issues have serious implications for using active learning in practice. I argue that, in order to truly reduce the labeling cost required to build an accurate model, the notion of annotation cost must be better understood and incorporated into the active learning process.

In some problem domains, the cost required to label an instance is known before the learner makes a query. For example, if labels are acquired by executing a biological experiment, then the cost of a query might be the price of the materials used (King et al., 2004), which is presumably fixed and available to the learner. In this work, I am primarily concerned with reducing costs that are *not known* in advance. Specifically, I investigate the nature of reducing the annotation time for tasks involving human annotators. Time is the natural currency for labeling costs in such domains, since the scarce resource is usually the time available for annotators to label instances.

The vast majority active learning research has not considered that instances may vary in labeling cost. Some methods have been developed for the situation in which an *active classifier* may incur a cost to obtain additional feature values at classification time (Greiner et al., 2002). This work, in contrast, is focused on settings in which unlabeled instances (and their feature descriptions) are readily available, but the labeling process incurs a cost at training time. One proposed approach

for reducing human annotation effort in active learning involves using the current learned model to assist in the labeling of query instances in structured learning tasks like parsing (Baldridge and Osborne, 2004) or information extraction (Culotta and McCallum, 2005). However, these methods do not actually represent or reason about costs. Instead, they attempt to reduce the number of annotation actions required for a query that has already been selected.

The prior work that is most closely related to the present chapter is a group of approaches that explicitly account for varying label costs in active learning. One such cost-sensitive query selection algorithm was proposed by Margineantu (2005), but differs from ours in that labeling costs are assumed to be known for each instance, and the paper does not provide any empirical evaluation using real-world data sets. Kapoor et al. (2007) have developed a related approach that takes into account both labeling costs and estimated misclassification costs. In this setting, each candidate query is evaluated by summing the labeling cost for the instance and the expected future misclassification costs that would be incurred if the instance were added to the training set. Kapoor et al. applied their method in a voicemail classification task, but instead of using real cost information, their experiments make the simplifying assumption that the cost of labeling a voicemail message is a linear function of its length (e.g., ten cents per second). King et al. (2004) present the only work that, to my knowledge, uses active learning in an attempt to reduce real labeling costs. They describe a "robot scientist" which can execute a series of autonomous biological experiments to discover metabolic pathways, with the objective of minimizing the cost of materials used. In contrast to the tasks I consider, the labeling cost of each instance in this previous work is known prior to querying the instance.

This chapter presents a detailed analysis of four data sets for which I have measured the annotation cost (labeling time, and in some cases labeling actions) incurred by humans annotators. This investigation attempts to answer several important questions about the role of such annotation costs in real-world active learning. Are annotation times variable for a given task or domain? Do these times change from one annotator to the next? Are they stationary, or do they change over time? How stochastic are they? Can the annotation times, if not known or trivially estimated, be accurately predicted? And finally, can we produce better active learning systems by incorporating cost information into the query selection process?

6.2 Data Sets and Annotation Methodology

Because most active learning research has not been concerned with reducing real annotation cost, I am not aware of any data sets with real cost information. I contacted the organizers for at least five benchmark efforts involving human annotators to try to obtain cost data for their respective data sets. While some could provide rough estimates about the average annotation time per instance, none of them logged actual annotation times (or any other form of cost) for individual instances. Therefore, I conducted several annotation experiments of my own in which these costs are recorded.

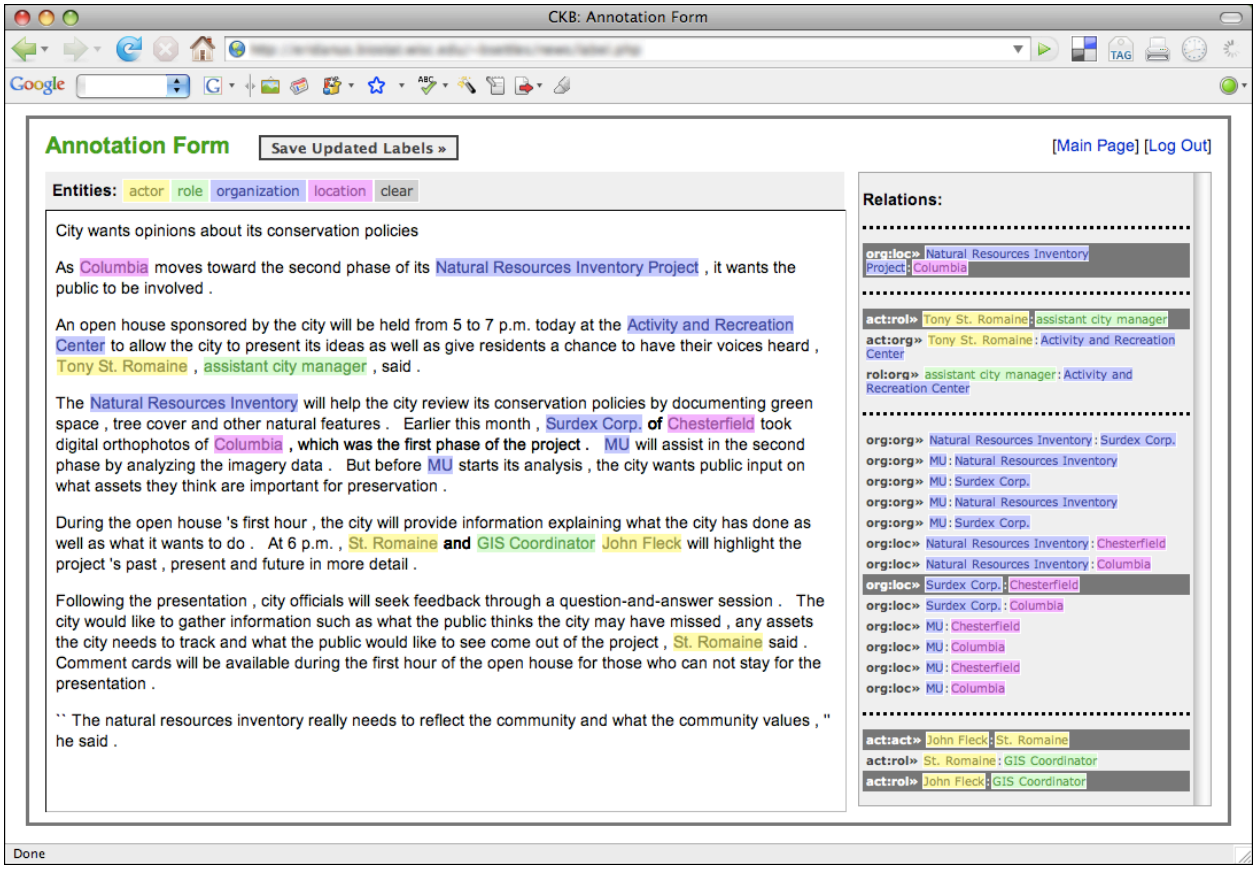


Figure 6.1: A screenshot of the CKB labeling interface. Article text appears in the window on the left, where annotators can highlight entities and label them with a word-processor style formatting menu. As the entities are labeled, candidate relations among them are dynamically generated in the window to the right, grouped by paragraph. Users then click on these relations to indicate which ones are true (in dark grey).

6.2.1 CKB News Corpus

This work was partially motivated by a collaborative project called Community Knowledge Base (CKB), headed by Lewis Friedland at the University of Wisconsin-Madison Department of Journalism and Mass Communication. The goal of the project is to build a software system for local newsrooms that can monitor local and regional news feeds, automatically extract information, and maintain a database of key players in the local community. The system provides access to a structured model of the community's social network for journalists researching news stories.

For an initial version of the CKB system, I focused on learning to extract four entities (actor, role, organization, location) and six binary relations among them (actor:actor, actor:role, actor:organization, role:organization, organization:organization, and organization:location). I

began by training a CRF to extract named entities using the CoNLL03 corpus (Sang and DeMeulder, 2003), augmented with a small set of articles annotated for the additional role entity (which is not part of that corpus). For this version of the system, I was able to collaborate with the Reynolds Journalism Institute at the University of Missouri. Textual sources consisted of articles published over a year in the *Columbia Missourian*, a working daily newspaper published by the school. After filtering documents for text encoding errors and outliers in length (keeping all articles between 200 and 600 words), the final pool consisted of 1,984 articles. The CRF described above was then used to automatically pre-annotate this pool of articles.

Each article was then labeled by one of five University of Missouri undergraduate journalism students. Figure 6.1 shows the interactive web-based annotation system used for (i) adding entity annotations or editing automatic pre-annotations and (ii) adding relation annotations. Articles were selected from the pool in a random order and each was presented to one annotator, thus no two annotators labeled the same article. The annotation system logged both the time elapsed during the labeling process, and the number of labeling "actions" taken for each article (label an entity, clear an incorrect entity, mark a putative relation as positive, etc.). The resulting corpus consists of 358 labeled articles.

6.2.2 SIVAL Image Repository

In Chapter 5, I presented a framework for active learning in *multiple-instance* (MI) problem domains. An important aim of that research was to study the value of actively selecting finer-granularity labels for objects in an MI representation. One application for MI active learning is content-based image retrieval (CBIR), in which images are represented as bags and instances correspond to processed, segmented regions of the image. MI active learning, therefore, would allow the learner to query image segments that correspond to parts of an object of interest.

Since no MI data sets with instance-level labels existed, I augmented the SIVAL repository by manually adding instance labels (see Section 5.4). Figure 6.2 shows the web-based interface I developed for the annotation process. This interface allows users to visually select which image segments belong to the indicated object, such as the WD40 can shown on the right-hand side of the screen. This repository consists of 1500 images, which were all manually labeled by three members of the machine learning research group at the University of Wisconsin, Madison. The annotations were done in an arbitrary order, and annotation times for each image were logged. As in the CKB corpus, images were not redundantly labeled by multiple annotators.

6.2.3 Speculative Text Corpus

There has been a growing interest recently in handling subjectivity in natural language tasks. Following previous work (Light et al., 2004), I annotated a corpus of biomedical abstracts for statements that use speculative vs. definite language. Consider the following excerpts from various PubMed[1] abstracts (with ID numbers):

[1] http://www.ncbi.nlm.nih.gov/pubmed/

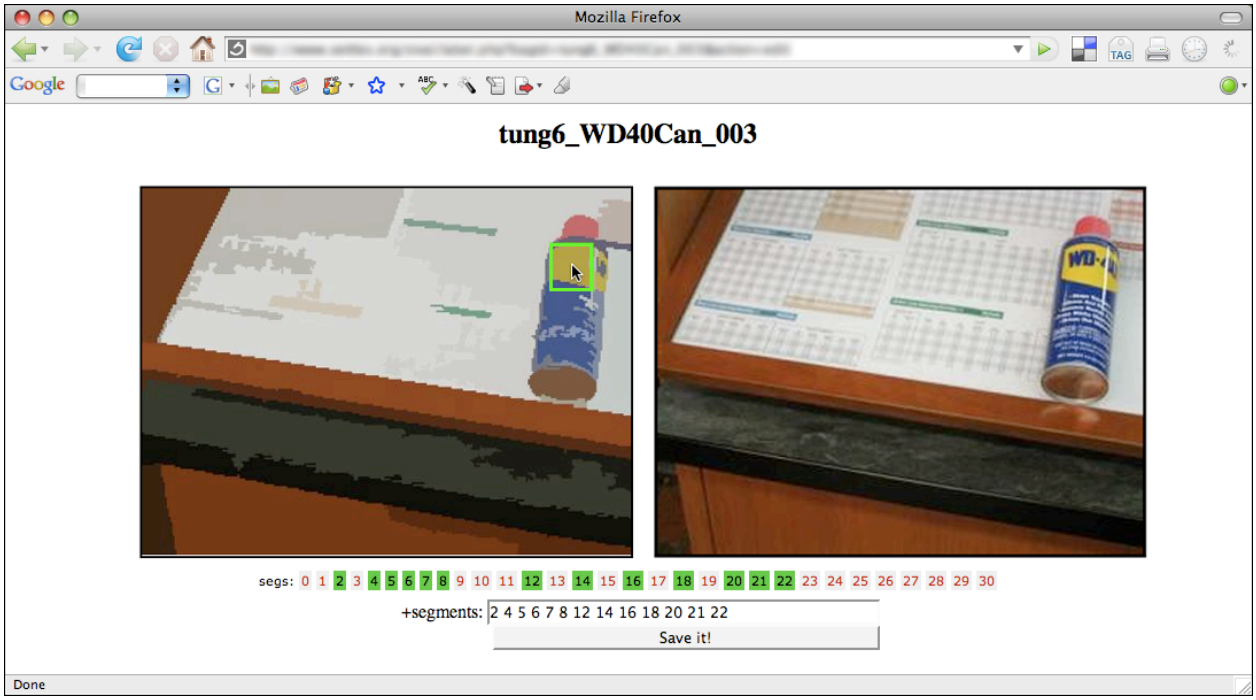


Figure 6.2: A screenshot of the SIVAL labeling interface. Annotators move the cursor over segments in the processed image on the left, clicking on those belonging to the target object. In this example, the highlighted segment belongs to the product label of a WD40 can. An original reference image is shown on the right.

1. "LBF4 has a molecular mass of 105 kDa and is probably unrelated to PU.1." (95074873)
2. "These observations suggest that de novo EBV infection of thymocytes differs from infection of B cells." (95266275)
3. "This study illustrates the almost complete tolerance of mice for human TCF-1 and demonstrates that this tolerance is readily broken by gene knock-out." (96030053)

Sentences 1 and 2 represent scientific conclusions that are speculative in nature. That is, they do not necessarily follow directly from observed data, but are expressed by the authors with some reserved degree of belief. Sentence 3 is definite, however, asserting quite firmly the author's belief in the results. For the speculative text corpus, I followed the guidelines used in previous work (Light et al., 2004) with two exceptions. First, since they found no significant inter-annotator agreement between "high" and "low" levels of speculation, I treated this as a binary classification task. Second, sentences were annotated individually to simulate queries being made at the granularity level of a single sentence.

I randomly selected 100 PubMed abstracts from the GENIA corpus (Kim et al., 2003) for manual annotation. Since previous work indicates that many speculative statements appear toward the

end of abstracts, all abstracts truncated for length by PubMed were excluded. All 850 resulting sentences were then labeled by three members of the machine learning research group using a simple web-based interface, and annotation times for each sentence were logged. Unlike the previous two data sets, all sentences were redundantly labeled by all three annotators in order to gather data on inter-annotator agreement. The ordering of instances was randomized, and annotators saw the instances in the same sequence.

6.2.4 SigIE Email Corpus

I also created a new corpus for the task of extracting contact details from email signature lines. For this, I randomly selected 250 signatures from the Sig+Reply corpus and manually added annotations for twelve address book fields such as name, phone, jobtitle, and postal address information. All annotations were performed in a random order, using a modified version of the CKB labeling interface. As with CKB, both annotation times and actions were logged. This is the same SigIE data set used in experiments from Chapter 3.

6.3 Analysis and Experiments

In this section, I consider six questions that are aimed at understanding how real annotation costs can be learned and exploited by active learning systems.

6.3.1 Are Annotation Times Variable for a Given Task or Domain?

If the goal of active learning is to reduce the total time required to train an accurate model, then this first question is critical. If times are approximately constant, the goal can be achieved by simply minimizing the number of labeled instances, as most work in active learning has done. If these times vary significantly, however, then these differences should be taken into account by the learner.

The answer to this question, however, is complicated. Figure 6.3 shows histograms that characterize the distribution of annotation times for each domain. For the CKB corpus, the majority of articles took from 56 seconds to just over 16 minutes ($\approx$1000 seconds), but ranged up to 1.73 hours (6275 seconds). SIVAL appears to have two peaks in its distribution, possibly because some objects such as apple are simple (with fewer segments to be labeled), while others like wd40can are more complex (composed of many segments, requiring more time). Most of these images required less than a minute, but some took as long as 3.4 minutes (204 seconds). The Spec corpus went very quickly, with only 7.6 seconds on average and no sentence requiring more than a minute. The distribution of SigIE is similar to SIVAL, but with a single mode and fewer apparent outliers. For all data sets, the standard deviation is more than half the mean (in the case of CKB, even greater), suggesting a fairly large degree of variability. But where does this variance come from? Is it dependent on the annotator, the nature of the task, or is it simply due to random noise? The

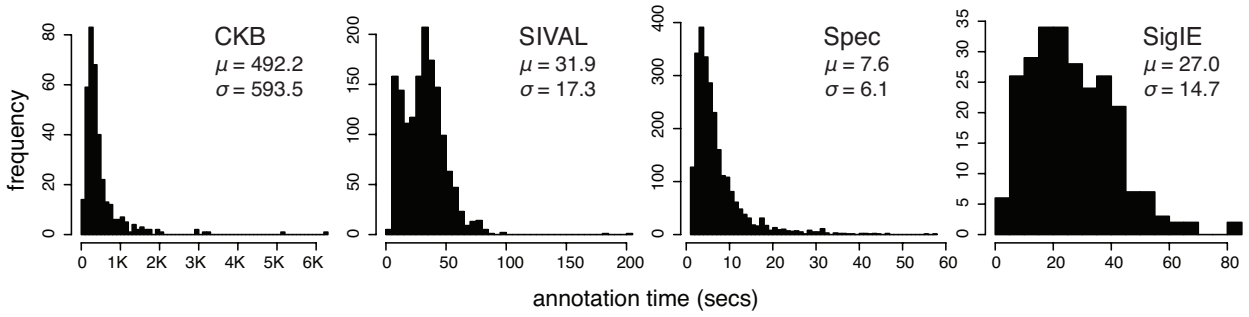


Figure 6.3: Histograms illustrating the distribution of annotation times for each data set in this study. The mean annotation time μ and standard deviation σ for each data set is also reported.

rest of this section is aimed at better understanding this variance and, more importantly, how it can be utilized by the active learner.

6.3.2 Do Times Vary from One Annotator to the Next?

Figure 6.4 provides a more detailed look at the annotation time distributions for each annotator. Individual annotators are identified by a unique ID, e.g., CKB1 and CKB2 are both annotators for the CKB corpus. At a glance, we can see that some annotators look quite different from their neighbors. Some are generally faster or slower, some have a larger spread of annotation times, and some appear more prone to outliers.

I conduct two-sided Kolmogorov-Smirnov (KS) significance tests to see if these apparent differences are real. For the CKB corpus, four distribution pairs result in statistically significant differences at the 95% level: CKB1/CKB2, CKB1/CKB3, CKB2/CKB4, and CKB2/CKB5. (After Bonferroni correction, however, only CKB1/CKB2 remains significant.) For the SIVAL and Spec data sets, all differences are significant. We can conclude from these results that annotation behavior can indeed vary substantially from one annotator to the next. If we wish to leverage annotation cost information into the active learning process, and annotators exhibit these unique trends, then perhaps annotation cost should be modeled on a per-annotator basis (I return to this idea in Section 6.3.5).

6.3.3 Are Annotation Times Stationary?

It is possible that annotator behavior changes over time. If this is the case, any modeling of annotation time should be able to account for this variation. Figure 6.5 plots each annotator's average labeling time per instance as a function of the number of instances labeled thus far (all instances are considered in the order they were actually labeled). As the figure shows, most annotators are able to work somewhat faster as they progress, although the most significant gains seem to be during

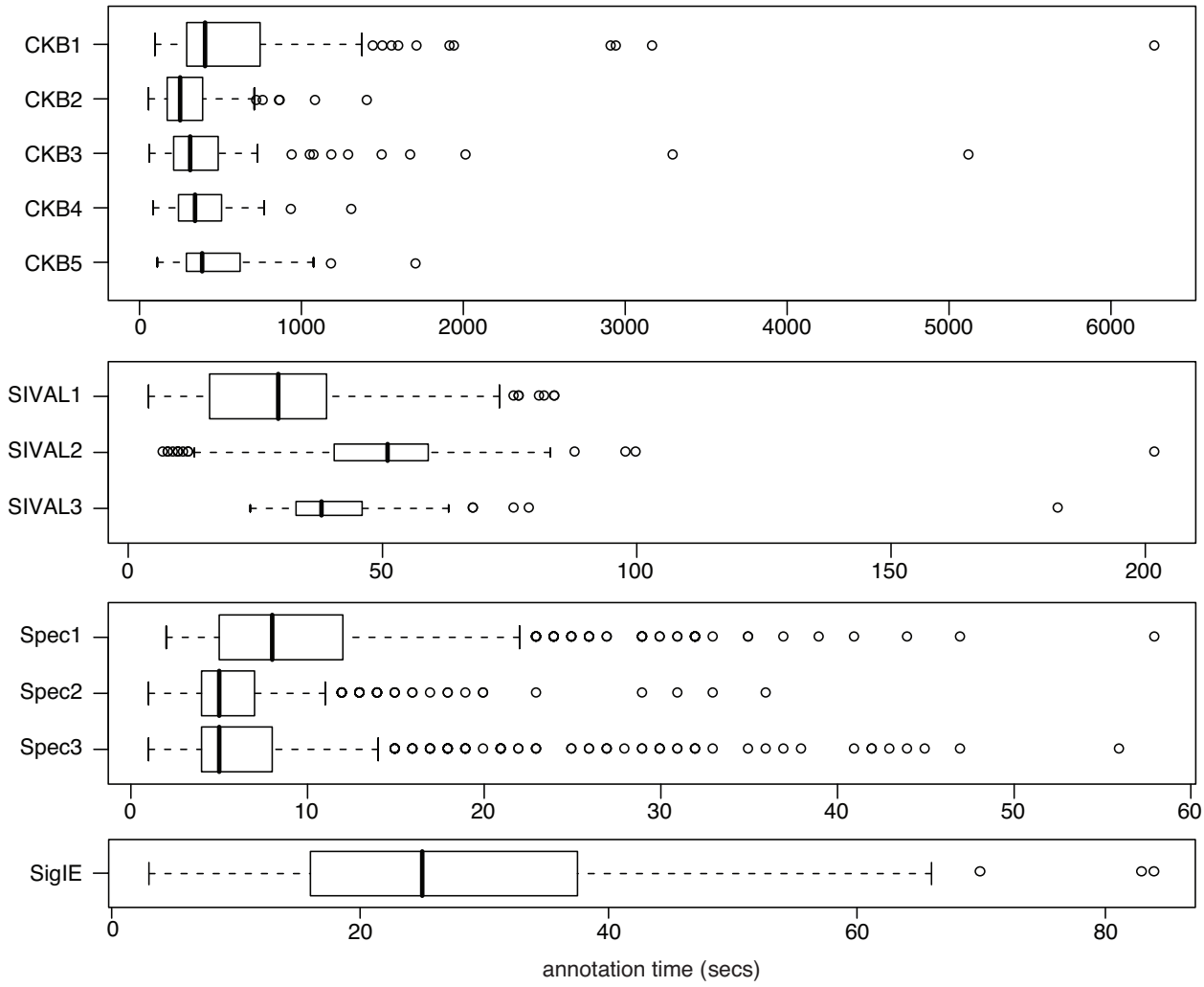


Figure 6.4: Box plots showing per-annotator labeling time distributions for each data set. A box represents the middle 50% of annotation times, and the median is marked with a thick black line. Box heights are scaled in proportion to the number of instances labeled. Whiskers on either side span the 1st and 4th quartiles of each distribution, up to 1.5 times the inner-quartile range (i.e., box width). Circles indicate possible outliers. Note that the range of the horizontal axis varies across data sets.

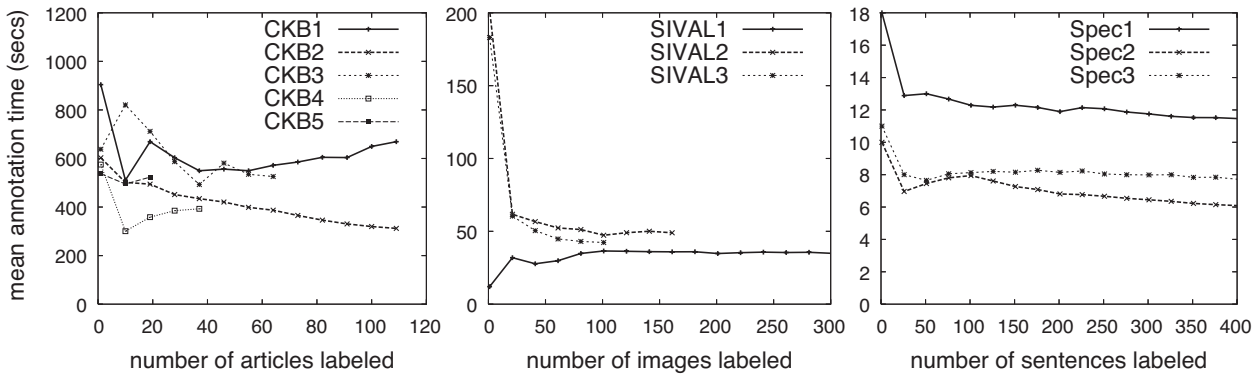


Figure 6.5: Average annotation time per instance versus the number of instances labeled. SigIE is omitted here, but exhibits a similar pattern.

the first few annotations. Presumably, this is because they are unfamiliar with both the task and the annotation interface early on, but are able to adapt quickly. The notable exception of SIVAL1 slowing down is because the annotator's first few images depicted simple objects that generally took less time than more typical images. CKB1 and CKB4 decelerate slightly as well, although they remain much faster on average than at the beginning. These data suggest that, while most annotators demonstrate a rapid speed-up early during labeling, the "burn-in" period is brief, and annotation times are relatively stationary thereafter.

6.3.4 How Stochastic Are Annotation Times?

There are two kinds of noise one might encounter when measuring annotation time. The first, which I call *jitter*, is the cumulative effect of small human and/or machine delays, such as momentary fatigue or network latency. If the same instance were labeled multiple times under similar conditions, we would expect to see minor differences in annotation time due jitter. The second type of noise, which I call *pause*, arises from unexpected interruptions such as a phone call or taking a long email break. Labeling times subject to pause should be faster under normal circumstances.

I consider two analyses to try to determine the extent to which jitter and pause factor into two of the data sets in this study. First, since each instance in the Spec corpus was labeled by all three annotators, we can assess how well their labeling times correlate with one another. One might think of these redundant labelings as a surrogate for instances being labeled multiple times under similar conditions (although the annotator does vary). Figure 6.6(a) shows a 3D scatterplot of annotation times for this corpus among the three annotators. Although there are positive correlations between the annotators, the correlation is not strong (pairwise correlation coefficients are between 0.258 and 0.328). This result suggests that jitter has a fairly large effect on labeling times for this data set. This is probably because labeling times are very quick in this domain, leaving more room for such stochastic effects.

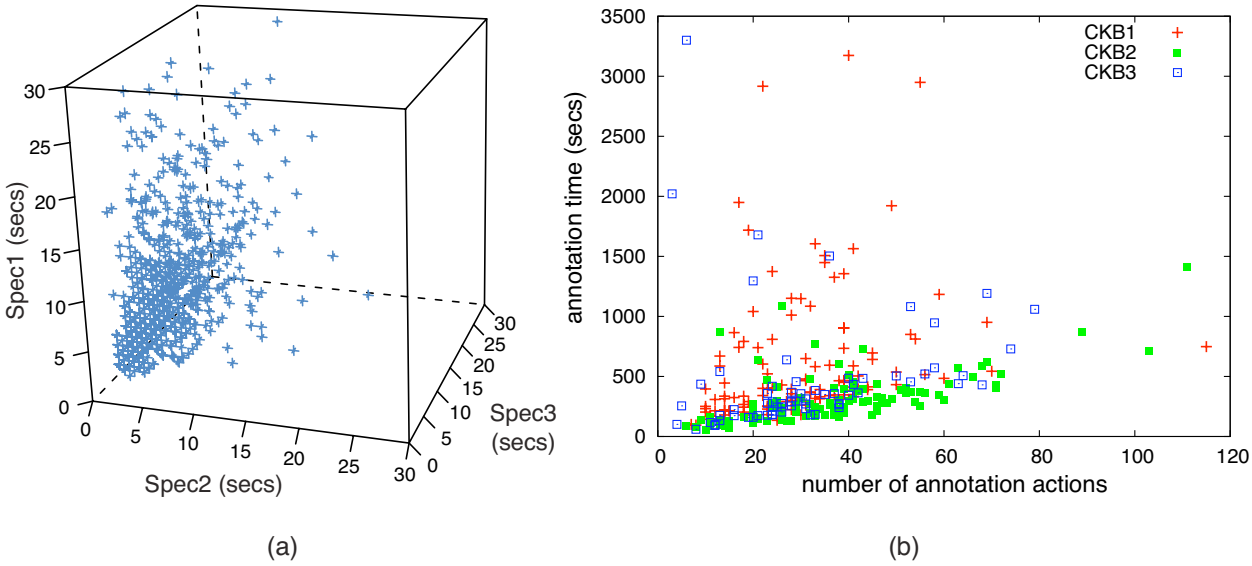


Figure 6.6: Stochastic artifacts of annotation time. (a) 3D scatterplot of the Spec corpus. Each point represents a sentence, plotted in a 3D space whose axes correspond to labeling times for each annotator. (b) Annotation time vs. actions in the CKB corpus. Each point is an article, and the shape of each point indicates the annotator (only three shown, for clarity).

The second analysis considers the relationship between annotation time and the number of annotation actions performed for each query in the CKB corpus, as shown in Figure 6.6(b). If we assume that the time required to annotate an article is proportional the the number of labeling actions, we would expect a strong correlation between them. Indeed, there is a fairly linear relationship. However, a few articles took much longer than the number of actions would imply, suggesting that the annotator was somehow distracted for an extended period. I argue that these large departures from the expected annotation time are indicative of pause. Note that some annotators seem more prone to pause than others. For example, CKB1 and CKB3 annotated all of the extreme points in the above figure (as well as the extreme outliers for the CKB corpus in Figure 6.4). The correlation coefficient between actions and time for both of these annotators is only 0.108, whereas the other three annotators are all between 0.624 and 0.744.

In practice, there is little that can be done to correct for jitter, and I conjecture that its effect on annotation time is minimal anyway. However, if we wish to reason about annotation time and utilize this information in the active learning process, then the methods we use should be able to detect and remain robust to pause.

6.3.5 Can Annotation Times Be Accurately Predicted?

To reduce the total annotation cost in active learning, I argue that query costs should be taken into account by the learner. Unlike the forms of cost previously considered by others (Kapoor et al., 2007; King et al., 2004), knowledge about the labeling time for each instance in these domains is not available to the learner at query time. Therefore, I consider whether or not these unknown annotation times can be accurately predicted.

This problem can be approached as a *regression* learning task (i.e., the predicted label is a real-valued number, rather than from a discrete class distribution). The annotation time can be predicted for each query candidate based on a few simple numerical input features. For the CKB corpus, I use seven features, such as the number of words, entities, candidate relations, paragraphs, etc. Note that some of these features depend on other quantities which are unknown at query time, such as the number of entities or relations in an article. To handle this, I use the current task-model predictions to estimate these quantities (details for task models, which are trained alongside the regression cost-models, are given in Section 6.3.6). For SIVAL, using five features: the min, max, mean, and standard deviation of the image segment sizes (in pixels), plus the task model's predicted number of positive segments. For Spec, I use four features: the number of ASCII characters, words, and unique features used by the classification task model (I use a "bag of words" representation, subject to stop-word removal and stemming), plus this model's uncertainty about the class label (in these experiments, entropy). For SigIE, I use four features: the number of entities, lines, and characters, plus the percentage of characters that are non-alphanumeric. I emphasize that, for all domains, I made little effort to "engineer" these features. Predicting annotation times could valuable if it can be done with few training examples, and with a minimum of human effort. Therefore, I run these experiments using my initial intuitions about easy-to-compute, domain-independent features.

After using a variety of regression learning algorithms, I have found the SMO algorithm (Smola and Schölkopf, 1998) for support vector regression to be most accurate for the representations I consider here. The accuracy of a cost-model's prediction p against the true annotation time t can be evaluated using the correlation coefficient $r = \frac{\sum_i (p_i - \mu_p)(t_i - \mu_t)}{(n-1)\sigma_p \sigma_t}$ and relative absolute error $E = 100\% \times \frac{\sum_i |p_i - t_i|}{\sum_i |t_i - \mu_t|}$, where μ_t and σ_t are the mean and standard deviation of t, respectively. I plot learning curves averaged using ten-fold cross-validation (five-fold for CKB). As with previous work on the SIVAL data set, experiments in that domain are done on a binary per-class basis and averaged over 20 independent runs. In each run, half the positive images are used for training and the other half are held aside for evaluation. Note that MI active learning in these experiments corresponds to Scenario IV from Chapter 5, since the learner queries an image, and the annotator labels all segments.

Learning curves for the annotation-time prediction experiments are shown in Figure 6.7. These plots show that, in general, annotation times appear to be quite learnable. In particular, the CKB2 cost-model achieves a correlation coefficient of 0.626 and error of 63%. The combined Spec cost-model also does well, reaching 0.587 correlation and 63% error. The SigIE cost-model is the most accurate by far, ultimately achieving error below 45% and correlation of 0.852. I also emphasize that where annotation times are learnable, they appear to be learnable from only a few instances.

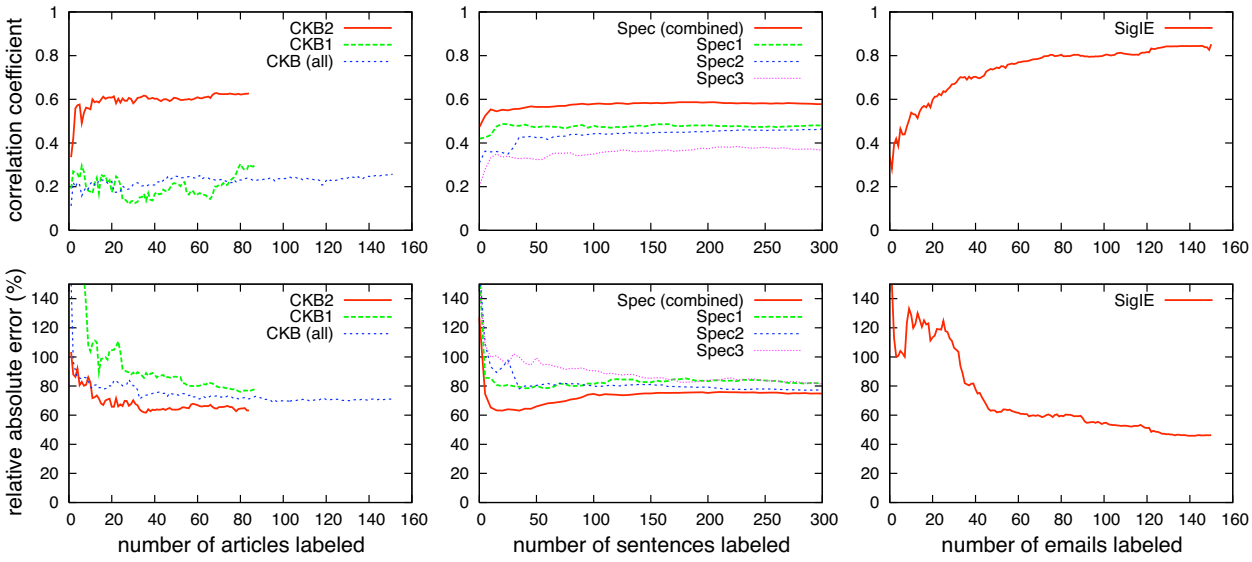


Figure 6.7: Learning curves for predicting annotation time in terms of (top) correlation coefficient and (bottom) relative absolute error. CKB plots show three curves: all annotator times pooled together, plus annotator-specific models for CKB1 and CKB2. Spec plots include a model that predicts the combined annotation time for all annotators, compared to each annotator-specific model.

To show that these learned models are better than simple linear functions of length (as considered by Kapoor et al. (2007)), I report correlation coefficients between annotation time and the document length (in characters) for the following: CKB2 = 0.291, SPEC(combined) = 0.572, and EMAIL = 0.455. Other simple estimators (e.g., number of words, lines, or sentences) produce almost identical results. This demonstrates that at least the CKB2 and SigIE cost-models produce significantly more accurate estimates of cost than a simple heuristic approach.

We can also see that the CKB2-specific cost-model is more accurate at predicting annotation times than the one tailored to CKB1, or for all annotators pooled together. Recall from Section 6.3.2 that labeling behavior varies greatly from one annotator to the next. We can also conclude that these differences can have an acute impact on how learnable that labeling behavior is. In Section 6.3.4, I noted that CKB1 was prone to a type of noise called pause, while CKB2 is not. This probably widens the gap in accuracy between their two models: since the CKB1-specific cost-model does not detect and handle pause, it may be prone to try and learn from this noise. We see some variation in learnability among SPEC annotators as well, though it is far less profound. Interestingly, the cost-model that aims to predict the combined annotation time of all SPEC annotators is the most accurate. I hypothesize that this is because the combined annotation time factors out some of the effects of jitter, the other type of noise.

Annotation times for SIVAL tasks, on the other hand, seem generally unlearnable using the representation described here. For most objects tested, error is greater than 100% and correlation

is near zero. Recall from Figure 6.3 that the distribution of SIVAL labeling times appears multi-modal, and I hypothesized that some objects simply require more or less time to label than others. To confirm this, I conduct KS tests between all these distributions, and find that 65% (37% after Bonferroni correction) are significantly different. This is evidence that most of the cost differences in the SIVAL repository are object-specific, and there is little other variation (other than noise or general annotator speed) within a particular class.

6.3.6 Can We Improve Active Learning by Utilizing Cost Information?

So far, this chapter has presented an extensive analysis of the nature and learnability of annotation time as a labeling cost in four problem domains. I now consider whether or not predicted annotation times can benefit an active learner by reducing the amount of time required to achieve a certain level of accuracy. To do this, I consider cost-sensitive variants of entropy-based uncertainty sampling to query informative but inexpensive instances.

For the CKB corpus, I use a CRF trained using a typical feature set to extract entities from newswire text (i.e., the same features used for the CoNLL03 corpus throughout this thesis[2]). The CRF parses one sentence at a time, and dynamically generates a set of candidate relations based on the predicted entities. These relations are described with contextual features (e.g., entity labels and the "bag of words" between them), and then classified with a MaxEnt model. I treat relation extraction as a seven-label classification task: the six legal relations plus null-relation. Unlike the typical active learning setting, a query in the CKB domain is an *article*, which in fact is a *set* of instances: an entity sequence (plus several candidate relations) per sentence. For simplicity, I treat each instance in article $\mathcal{X}$ as independent, thus the article uncertainty is given by $\phi(\mathcal{X}) = \sum_{x \in \mathcal{X}} \phi(x)$, the sum of all its instance entropies. For the Spec corpus, I train a MaxEnt model using a "bag of words" feature set subject to stop-word filtering and stemming. Since a query in this domain corresponds to a single instance x, estimating the uncertainty for $\phi(x)$ is straightforward. For SigIE, I use the same feature set as in CKB (minus syntactic features, such as part-of-speech tags). A query for this task is a single sequence $\mathbf{x}$, thus estimating $\phi(\mathbf{x})$ is straightforward; for these experiments I use the N-best sequence entropy approach (Section 3.2.1) to estimate the uncertainty of sequences.

I compare standard entropy-based uncertainty sampling with a simple "bang for your buck" cost-sensitive heuristic approach, where entropy is divided by the predicted labeling time for an instance. I employ the cost-models from Section 6.3.5, which are trained alongside these task-models (using only the labeled instances) to predict the labeling time. I also compare against two baselines: the cost-sensitive method using *known* annotation times, and random sampling. The task-models are evaluated using F_1 learning curves, using ten-fold cross-validation (five-fold for CKB).

Results for these active learning experiments are shown in Figure 6.8. For the CKB and SigIE corpora, the standard entropy query scheme does not reduce the time required to achieve the same

[2]Syntactic features, such as part-of-speech tags and phrasal boundaries, are predicted using an syntactic parser for English available from http://nlp.stanford.edu/software/lex-parser.shtml.

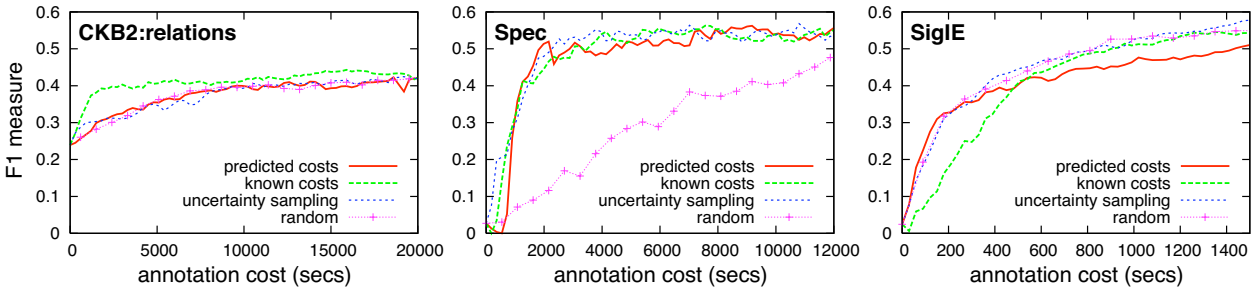


Figure 6.8: Active learning curves. Cost-sensitive variants using predicted and known annotation times are compared to standard entropy-based uncertainty sampling and a random sampling baseline. Note that the horizontal axis represents actual annotation cost.

accuracy as random sampling. However, it is important to note that when these curves are instead plotted as a function the number of queries (not shown), entropy does produce better learning curves. This indicates that naïve uncertainty sampling is prone to select informative, but time-consuming queries in these domains, resulting in no net reduction in cost.

While this cost-sensitive approach using predicted annotation times does not outperform the random baseline for CKB, we do see significant gains when the annotation time is known. This suggests that better learning curves can be achieved if labeling time can be predicted accurately and utilized appropriately. I note that, while I only report the CKB2 relation subtask here, results for CKB1 and both their entity subtasks are nearly identical. For the Spec corpus, standard entropy-based active learning does produce better curves, and in this case the cost-sensitive variants are roughly equivalent. I surmise that this is because annotation times are indeed approximately constant (with observed variations being due to jitter), thus cost information is of little value. Curves for the SigIE task are interesting because cost-sensitive active learning with known costs actually performs worse than random. I suspect that this is because, in this domain, shorter instances actually contain less valuable information. For example, a brief email signature may only contain name and email fields and therefore be very quick and easy to annotate, but lacks important rare fields such as jobtitle or phone. This result underscores the importance of understanding the relationship between the annotation cost of an instance and its overall value to the learner.

6.4 Summary and Future Work

To date, most research in active learning has assumed that the cost of acquiring a label is the same for all instances. Some recent work has considered cases where labeling costs are variable, but these have either assumed that the cost is known for each instance (King et al., 2004; Margineantu, 2005) or can be approximated using a simple estimator (Kapoor et al., 2007). In this chapter, I have presented an extensive empirical study of annotation costs in four real-world text and image domains. To my knowledge, this is the first empirical investigation of annotation

costs in a real setting. This analysis provides several conclusions that have implications for active learning in domains where labeling is done by human annotators:

- In most of the domains considered here, annotation costs are not (approximately) constant across instances, and can instead vary considerably.
- Consequently, active learning approaches which ignore cost information may perform no better than random instance labeling. However, improved learning curves are achievable if an active learner can take these variable costs into account appropriately.
- In some domains, the cost for annotating an instance may not be intrinsic, but instead vary according to the person doing the annotation.
- In some domains, the measured cost for an annotation may include a stochastic component. The effects of this seem to depend, in part, on the typical time required to label an instance, and the proficiency of the annotators.
- In some domains, we can accurately learn to predict annotation costs, even after seeing only a few training instances. Moreover, these learned cost-models can be significantly more accurate than heuristics (e.g., document length).

This last result suggests that, even when annotation costs are not known before querying, an active learner may be able to profitably reason about them. I have attempted to exploit this by training a cost-model to predict annotation costs while simultaneously training the actual task-model. However, the simple "bang for your buck" cost-sensitive approach considered here does not appear to capture the necessary aspects of the problem. A main focus in future work will be to investigate cost-sensitive active learning strategies that are more robust when given approximate, predicted annotation costs.

Chapter 7

Active Learning Literature Survey

This chapter provides a general review of the literature on active learning. There have been a host of algorithms and applications for learning with queries over the years. This review is by no means a comprehensive look at active learning, as the field (like so many subfields in computer science) is evolving rapidly. I apologize in advance for any omissions or inaccuracies, and welcome comments or corrections at `bsettles@cs.wisc.edu`. To make this survey more useful in the long term, an expanded, online version will be maintained and updated indefinitely at:
`http://pages.cs.wisc.edu/~bsettles/active-learning/`

7.1 What is Active Learning?

Active learning (also called "query learning," or sometimes "experimental design" in the statistics literature) is a subfield of machine learning and, more generally, artificial intelligence. The key hypothesis is that a learning algorithm can achieve greater accuracy with fewer labeled training instances in cases where it is allowed to selectively choose the data from which is learns. An active learner is allowed to ask *queries* in the form of unlabeled instances to be labeled by an *oracle* (e.g., a human annotator). Active learning is well-motivated in many modern machine learning problems where data may be abundant but labels are scarce or expensive to obtain. Note that this kind of active learning is related in spirit, though not to be confused, with instructional techniques of the same name in the education literature (Bonwell and Eison, 1991).

7.2 Scenarios

Generally speaking, active learning is an iterative process. A learner first requests a label of the oracle, who labels the query and adds it to the learner's training set, the learner re-trains, and the process repeats. Once a query has been made, there are usually no additional assumptions on the part of the learning algorithm. The new labeled instance is simply added to the labeled set $\mathcal{L}$, and the learner proceeds from there in a standard supervised way. There are a few exceptions to this, such as when the learner is allowed to make alternative types of queries (Section 7.9), or when active learning is combined with semi-supervised learning (Section 7.10.1).

There are several different problem scenarios in which the learner may be able to ask queries. The three main settings that have been considered in the literature so far are (i) membership query synthesis, (ii) stream-based selective sampling, and (iii) pool-based active learning. The remainder of this section provides an overview of these three active learning scenarios.

7.2.1 Membership Query Synthesis

One of the first active learning scenarios to be investigated is learning with *membership queries* (Angluin, 1988). In this setting, the learner may request labels for any unlabeled instance in the input space, including (and typically assuming) queries that the learner generates de novo, rather than those sampled from some underlying natural distribution. Efficient query synthesis is often tractable and efficient for finite problem domains (Angluin, 2001). The idea of synthesizing queries has also been extended to regression learning tasks, such as learning to predict the absolute coordinates of a robot hand given the joint angles of its mechanical arm as inputs (Cohn et al., 1996).

Query synthesis is reasonable for many problems, but labeling such arbitrary instances can be awkward if the oracle is a human annotator. For example, Baum and Lang (1992) employed membership query learning with human oracles to train a neural network to classify handwritten characters. They encountered an unexpected problem: many of the query images generated by the learner contained no recognizable symbols, only artificial hybrid characters that had no natural semantic meaning. Similarly, one could imagine that membership queries for natural language processing tasks might create streams of text or speech that amount to gibberish. The stream-based and pool-based scenarios (described below) have been proposed to address these limitations.

However, King et al. (2004) describe an innovative and promising real-world application of the membership query scenario. They employ a "robot scientist" which can execute a series of autonomous biological experiments to discover metabolic pathways in yeast. Here, an instance is a mixture of chemical solutions that constitute a growth medium, as well as a particular yeast (*Saccharomyces cerevisiae*) mutant. A label, then, is whether or not the mutant thrived in the growth medium. All experiments are autonomously synthesized using an active learning approach based on inductive logic programming, and physically performed using a laboratory robot. This active method results in a three-fold decrease in the cost of experimental materials compared to naïvely running the least expensive experiment, and a 100-fold decrease in cost compared to randomly generated experiments. In domains where labels come not from human annotators, but from experiments such as this, query synthesis may be a promising direction for automated scientific discovery.

7.2.2 Stream-Based Selective Sampling

An alternative to synthesizing queries is *selective sampling* (Cohn et al., 1994). The key assumption is that obtaining an unlabeled instance is free (or inexpensive), so it can first be sampled from the actual distribution, and then the learner can decide whether or not to request its label. This approach is sometimes called *stream-based* or *sequential* active learning, as each unlabeled

instance is typically drawn one at a time from the data source, and the learner must decide whether to query or discard it. If the input distribution is uniform, selective sampling may well behave like membership query learning. However, if the distribution is non-uniform and (more importantly) unknown, we are guaranteed that queries will still be sensible, since they come from a real underlying distribution.

The decision whether or not to query an instance can be framed several ways. One approach is to evaluate samples using some "informativeness measure" or "query strategy" (see Section 7.3 for examples) and make a biased random decision, such that more informative instances are more likely to be queried (Dagan and Engelson, 1995). Another approach is to compute an explicit *region of uncertainty* (Cohn et al., 1994), i.e., the part of the instance space that is still ambiguous to the learner, and only query instances that fall within it. A naïve way of doing this is to set a minimum threshold on an informativeness measure which defines the region. Instances whose evaluation is above this threshold are to be queried. Another more principled approach is to define the region that is still unknown to the overall model class, i.e., to the set of hypotheses consistent with the current labeled training set called the *version space* (Mitchell, 1982). In other words, if any two models of the same model class (but different parameter settings) agree on all the labeled data, but disagree on some unlabeled sample, then that sample lies within the region of uncertainty. Calculating this region completely and explicitly is computationally expensive, however, and it must be maintained after each new query. As a result, approximations are used in practice (Cohn et al., 1994; Dasgupta et al., 2008).

The stream-based scenario has been studied in several real-world tasks, including part-of-speech tagging (Dagan and Engelson, 1995), sensor scheduling (Krishnamurthy, 2002), and learning ranking functions for information retrieval (Yu, 2005). Fujii et al. (1998) employ selective sampling in active learning for word sense disambiguation, e.g., determining if the word "bank" means land alongside a river or a financial institution in a given context (only they study Japanese words in their work). The approach not only reduces annotation effort, but also limits the size of the database used in nearest-neighbor learning, which in turn expedites the classification algorithm.

It is worth noting that some authors (e.g., Thompson et al., 1999; Moskovitch et al., 2007) use "selective sampling" to refer to the pool-based scenario described in the next subsection. Under this interpretation, the term merely signifies that queries are made with select instances sampled from a real data distribution. However, in most of the literature selective sampling refers to the stream-based scenario discussed here.

7.2.3 Pool-Based Active Learning

For many real-world learning problems, large collections of unlabeled data can be gathered at once. This motivates *pool-based active learning* (Lewis and Gale, 1994), which assumes that there is a small set of labeled data $\mathcal{L}$ and a large pool of unlabeled data $\mathcal{U}$ available. Queries are selectively drawn from the pool, which is usually assumed to be closed (i.e., static or non-changing), although this is not strictly necessary. Typically, instances are queried in a greedy fashion, according to an informativeness measure used to evaluate all instances in the pool (or, perhaps if $\mathcal{U}$ is very large, some subsample thereof).

The pool-based scenario has been studied for many real-world problem domains in machine learning, such as text classification (Lewis and Gale, 1994; McCallum and Nigam, 1998b; Tong and Koller, 2000; Hoi et al., 2006a), information extraction (Thompson et al., 1999; Settles and Craven, 2008), image classification and retrieval (Tong and Chang, 2001; Zhang and Chen, 2002), video classification and retrieval (Yan et al., 2003; Hauptmann et al., 2006), speech recognition (Tur et al., 2005), and cancer diagnosis (Liu, 2004) to name a few.

The main difference between stream-based and pool-based active learning is that the former scans through the data sequentially and makes query decisions individually, whereas the latter evaluates and ranks the entire collection before selecting the best query. While the pool-based scenario appears to be much more common among application papers, one can imagine settings where the stream-based approach is more appropriate. For example, when memory or processing power may be limited, as with mobile and embedded devices.

7.3 Query Strategy Frameworks

All active learning scenarios involve evaluating the informativeness of unlabeled instances, which can either be generated de novo or sampled from a given distribution. There have been many proposed ways of formulating such *query strategies* in the literature. This section provides an overview of the general frameworks used to date. From this point on, I use the notation $\phi(\cdot)$ to represent a query strategy, which is a function used to evaluate the informativeness of a query. The "best" query instance x is the one which maximizes this function.

7.3.1 Uncertainty Sampling

Perhaps the simplest and most commonly used query framework is *uncertainty sampling* (Lewis and Gale, 1994). In this framework, an active learner queries the instances about which it is least certain how to label. This approach is often straightforward for probabilistic learning models. For example, when using a probabilistic model for binary classification, an uncertainty sampling strategy simply queries the instance whose posterior probability of being positive is nearest 0.5 (Lewis and Gale, 1994; Lewis and Catlett, 1994).

A more general uncertainty sampling strategy uses *entropy* (Shannon, 1948) as an uncertainty measure:

$$\phi^{ENT}(x) = -\sum_i P(y_i|x) \log P(y_i|x),$$

where y_i ranges over all possible labelings. For binary classification, entropy-based uncertainty sampling is identical to choosing the instance with posterior closest to 0.5. However, the entropy-based approach can be generalized easily to probabilistic multi-label classifiers and probabilistic models for more complex structured instances, such as sequences (Settles and Craven, 2008) and trees (Hwa, 2004). An alternative to entropy in these more complex settings involves querying the instance whose best labeling is the least confident:

$$\phi^{LC}(x) = 1 - P(y^*|x),$$

where $y^* = \text{argmax}\, P(y|x)$ is the most likely class labeling. This sort of strategy has been shown to work well, for example, with conditional random fields or CRFs (Lafferty et al., 2001) for active learning in information extraction tasks (Culotta and McCallum, 2005; Settles and Craven, 2008).

Uncertainty sampling strategies may also be employed with non-probabilistic models. One of the first works to explore uncertainty sampling used a decision tree classifier (Lewis and Catlett, 1994) by modifying it to have probabilistic output. Similar approaches have been applied to active learning with nearest-neighbor (a.k.a. "memory-based" or "instance-based") classifiers (Fujii et al., 1998; Lindenbaum et al., 2004), by allowing each neighbor to vote on the class label of x, with the proportion of these votes representing the posterior label probability. Tong and Koller (2000) also experiment with an uncertainty sampling strategy for support vector machines, or SVMs (Cortes and Vapnik, 1995), that involves querying the instance closest to the linear decision boundary. This last approach is analogous to uncertainty sampling with a probabilistic binary linear classifier, such as logistic regression or naïve Bayes.

7.3.2 Query-By-Committee

Another, more theoretically-motivated query selection framework is the *query-by-committee* (QBC) algorithm (Seung et al., 1992). The QBC approach involves maintaining a committee $\mathcal{C}$ of models which are all trained on the current labeled set $\mathcal{L}$, but represent competing hypotheses. Each committee member is then allowed to vote on the labelings of query candidates. The most informative query is considered to be the instance about which they most disagree.

The fundamental premise behind the QBC framework is minimizing the version space, which is (as mentioned in Section 7.2.2) the set of hypotheses that are consistent with the current labeled training data $\mathcal{L}$. If we view machine learning as a search for the "best" model within the version space, then our goal in active learning is to constrain the size of this space as much as possible (so that the search can be more precise) with as few labeled instances as possible. This is exactly what QBC does, by querying in controversial regions of the version space.

To use the QBC framework, one must (i) construct a committee of models that approximate different regions of the version space and (ii) have some measure of disagreement among them. Seung et al. (1992) accomplish the first task simply by sampling a committee of two random hypotheses that are consistent with $\mathcal{L}$. For generative model classes, this can be done more generally by randomly sampling models from some posterior distribution $P(\theta|\mathcal{L})$. For example, McCallum and Nigam (1998b) do this for naïve Bayes by using the Dirichlet distribution over model parameters, whereas Dagan and Engelson (1995) sample HMMs by using the Normal distribution. For other model classes, such as discriminative or non-probabilistic models, Abe and Mamitsuka (1998) have proposed *query-by-boosting* and *query-by-bagging*, which employ the well-known ensemble learning methods boosting (Freund and Schapire, 1997) and bagging (Breiman, 1996) to construct committees. Melville and Mooney (2004) propose another ensemble-based method which encourages diversity among committee members. For measuring the degree of disagreement, two main approaches have been proposed: vote entropy (Dagan and Engelson, 1995) and average KL-divergence (McCallum and Nigam, 1998b). There is no consensus on the appropriate committee size to use, which may in fact vary by model class or application. However, even small

committee sizes (e.g., two or three) have been shown to work well in practice (Seung et al., 1992; McCallum and Nigam, 1998b; Settles and Craven, 2008).

Aside from the QBC framework, several other query strategies attempt to minimize the version space as well. For example, Cohn et al. (1994) describe a related selective sampling algorithm for neural networks using a combination of the "most specific" and "most general" models, which lie at two extremes the version space given the current labeled examples in the training set $\mathcal{L}$. Tong and Koller (2000) propose a pool-based query strategy that tries to minimize the version space for support vector machine classifiers directly. The membership query algorithms of Angluin (1988) and King et al. (2004) can also be interpreted as synthesizing de novo instances that limit the size of the version space. However, Haussler (1994) shows that the size of the version space can grow exponentially with the size of $\mathcal{L}$. This means that, in general, the version space of an arbitrary model class cannot be explicitly represented in practice. The QBC framework, rather, uses a committee which is a subset-approximation of the full version space.

7.3.3 Expected Model Change

Another general active learning framework is to query the instance that would impart the greatest change to the current model *if we knew its label*. An example query strategy in this framework is the "expected gradient length" (EGL) approach for discriminative probabilistic model classes. This strategy was introduced by Settles et al. (2008b) for active learning in the multiple-instance setting (Dietterich et al., 1997), and has also been applied to probabilistic sequence models like CRFs (Settles and Craven, 2008).

Since discriminative probabilistic models are usually trained using gradient-based optimization, the "change" imparted to the model can be measured by the length of the training gradient (i.e., the vector used to re-estimate parameter values). In other words, the learner should query the instance x which, if labeled and added to $\mathcal{L}$, would result in the new training gradient of the largest magnitude. Let $\nabla\ell(\mathcal{L};\theta)$ be the gradient of the objective function ℓ with respect to the model parameters θ. Now let $\nabla\ell(\mathcal{L} \cup \langle x, y\rangle;\theta)$ be the new gradient that would be obtained by adding the training tuple $\langle x, y\rangle$ to $\mathcal{L}$. Since the query algorithm does not know the true label y in advance, we must instead calculate the length as an expectation over all possible labelings:

$$\phi^{EGL}(x) = \sum_i P(y_i|x;\theta)\Big\|\nabla\ell(\mathcal{L} \cup \langle x, y_i\rangle;\theta)\Big\|,$$

where $\|\cdot\|$ is the Euclidean norm of each resulting gradient vector. Note that, at query time, $\|\nabla\ell(\mathcal{L};\theta)\|$ should be nearly zero since ℓ converged at the previous round of training. Thus, we can approximate $\nabla\ell(\mathcal{L} \cup \langle x, y_i\rangle;\theta) \approx \nabla\ell(\langle x, y_i\rangle;\theta)$ for computational efficiency, because the training instances are assumed to be independent.

The intuition behind this framework is that will prefer instances that are likely to most influence the model (i.e., have greatest impact on its parameters), regardless of the resulting query label. This approach has been shown to work well in empirical studies, but can be computationally expensive if both the feature space and set of labelings are very large.

7.3.4 Variance Reduction and Fisher Information Ratio

Cohn et al. (1996) propose one of the first statistical analyses of active learning, demonstrating how to synthesize queries that minimize the learner's future error by minimizing its variance. They describe a query strategy for regression learning problems, in which the output label is a real-valued number (rather than from discrete set of class labels). They take advantage of the result by Geman et al. (1992) showing that a learner's expected future generalization error can be decomposed in the following way:

$$\begin{aligned} E_T\left[(o-y)^2|x\right] &= E\left[(y-E[y|x])^2\right] \\ &\quad + \left(E_{\mathcal{L}}[o]-E[y|x]\right)^2 \\ &\quad + E_{\mathcal{L}}\left[(o-E_{\mathcal{L}}[o])^2\right], \end{aligned}$$

where $E_{\mathcal{L}}[\cdot]$ is an expectation over some labeled set $\mathcal{L}$ of a given size, $E[\cdot]$ is an expectation over the conditional density $P(y|x)$, and E_T is an expectation over both. Here also $o = g(x;\theta)$ is shorthand for the model's predicted output for a given instance x, while y indicates the true label.

The first term on the right-hand side of this equation is the *noise*, i.e., the variance of the true label y given only x, which does not depend on the model or training data. Such noise may result from stochastic effects of the method used to obtain the true labels, for example, or because the feature representation is inadequate. The second term is the *bias*, which represents the error due to the model class itself, e.g., if a linear model is used to learn a function that is only approximately linear. This component of the overall error is invariant given a fixed model class. The third term is the model's *variance*, which is the remaining component of the learner's mean squared error with respect to the true regression function. Minimizing the variance, then, is guaranteed to minimize the future generalization error of the model (since the learner itself can do nothing about the noise or bias components).

Cohn et al. (1996) then use the estimated distribution of the model's output to estimate $\tilde{\sigma}_o^2$, the expected variance of the learner after querying some new instance $\tilde{x}$. They show that this can be done in closed-form for neural networks, Gaussian mixture models, and locally-weighted linear regression models. In particular, for neural networks the output variance is approximated by (MacKay, 1992):

$$\sigma_o^2 \approx S(\mathcal{L};\theta)\left(\frac{\partial o}{\partial \theta}\right)^{\mathsf{T}}\left(\frac{\partial^2}{\partial \theta^2}S(\mathcal{L};\theta)\right)^{-1}\left(\frac{\partial o}{\partial \theta}\right), \tag{7.1}$$

where $S(\mathcal{L};\theta) = \frac{1}{L}\sum_{l=1}^{L}(o^{(l)}-y^{(l)})^2$ is the mean squared error of the current model θ on the training set $\mathcal{L}$. In the equation above, the second and last terms are computed using the gradient of the model's predicted output with respect to model parameters θ. The middle term is the inverse of a covariance matrix representing a second-order expansion around the objective function S with respect to θ. A closed-form expression for $\tilde{\sigma}_o^2$ can then be derived, given the assumptions that $\frac{\partial o}{\partial \theta}$ is locally linear (true for most network configurations) and that variance is Gaussian and constant for all x; further details are given by Cohn (1994). Since the equation is a smooth function and differentiable with respect to any query $\tilde{x}$ in the input space, gradient methods can be used to

search for the best possible query that minimizes future variance, and therefore future error. This approach is derived from statistical theories of optimal experimental design (Federov, 1972).

However, the approach of Cohn et al. (1996) applies only to regression tasks, and synthesizes new queries de novo. For many learning problems like text classification, this technique cannot be used. More recently, though, Zhang and Oles (2000) have proposed an analogous approach for selecting optimal queries in a pool-based setting for discriminative classifiers based on *Fisher information*. Formally, Fisher information $\mathcal{I}(\theta)$ is the variance of the *score*, which is the partial derivative of the log-likelihood function with respect to model parameters θ (Schervish, 1995). Fisher information is given by:

$$\mathcal{I}(\theta) = -\int_x P(x) \int_y P(y|x;\theta) \frac{\partial^2}{\partial \theta^2} \log P(y|x;\theta), \tag{7.2}$$

and can be interpreted as the overall uncertainty about an input distribution $P(x)$ with respect to the estimated model parameters. For a model with multiple parameters, Fisher information takes the form of a covariance matrix. The optimal instance to query, then, is the one which optimizes what I call the Fisher information ratio (FIR):

$$\phi^{FIR}(x) = -\text{tr}(\mathcal{I}_x(\theta)^{-1}\mathcal{I}_{\mathcal{U}}(\theta)), \tag{7.3}$$

where $\mathcal{I}_x(\theta)$ is the Fisher information matrix for an unlabeled query candidate $x \in \mathcal{U}$, and $\mathcal{I}_{\mathcal{U}}(\theta)$ is the matrix integrated over the entire unlabeled pool $\mathcal{U}$. The trace function $\text{tr}(\cdot)$ is the sum of the terms along the principal diagonal of a matrix, thus ϕ^{FIR} provides us with a ratio given by the inner product of $\mathcal{I}_x(\theta)$'s inverse matrix and $\mathcal{I}_{\mathcal{U}}(\theta)$. The minus sign in front is simply to ensure that ϕ^{FIR} acts as a maximizer.

The key idea behind the Fisher information ratio is that $\mathcal{I}_x(\theta)$ will tell us not only how uncertain the model is about query instance x (e.g., the magnitude of the matrix diagonal), but it also tells us which model parameters are most responsible for this uncertainty, as it is encoded in the matrix. Likewise, $\mathcal{I}_{\mathcal{U}}(\theta)$ can tell us the same information about the entire unlabeled pool. By minimizing the ratio in Equation 7.3, the learner will query the instance whose model variance is most similar to the overall input distribution approximated by $\mathcal{U}$. A more formal explanation as to why this is the optimal approach stems from the Cramér-Rao lower-bound on asymptotic efficiency, as explained by Zhang and Oles (2000). They apply this method to text classification using binary logistic regression. Hoi et al. (2006a) extend this approach to active text classification in the batch-mode setting (see Section 7.7) in which a set of queries $\mathcal{Q}$ is selected at once in an attempt to minimize the ratio between $\mathcal{I}_{\mathcal{Q}}(\theta)$ and $\mathcal{I}_{\mathcal{U}}(\theta)$. Settles and Craven (2008) have also generalized the Fisher information ratio approach to probabilistic sequence models such as CRFs.

The query strategies of variance reduction (Cohn et al., 1996) and Fisher information ratio (Zhang and Oles, 2000), while designed for different tasks and active learning scenarios, are grouped together here because they can be viewed as strategies under a more general *variance minimization* framework. Both are grounded in statistics, and both select the optimal query to reduce model variance given the assumptions. There are some practical disadvantages to these methods, however, in terms of computational complexity. In both strategies, estimating the variance requires

inverting a $K \times K$ matrix for each new example, where K is the number of parameters in the model θ, resulting in a time complexity of $O(UK^3)$, where U is the size of the query pool $\mathcal{U}$. This quickly becomes intractable for large K, which is a common occurrence in, say natural language tasks. For variance estimation with neural networks, Paass and Kindermann (1995) propose a sampling approach based on Markov chains to address this problem. For inverting the Fisher information matrix, Hoi et al. (2006a) use principal component analysis to reduce the dimensionality of the parameter space. Alternatively, Settles and Craven (2008) approximate the matrix with its diagonal vector, which can be inverted in only $O(K)$ time. However, these methods are still empirically much slower than simpler query strategies like uncertainty sampling.

7.3.5 Estimated Error Reduction

Query strategies that attempt to minimize generalization error directly have also been considered in the literature. The algorithms in the previous section minimize error indirectly by reducing model variance, however, this cannot be done in closed form for all model classes. We can instead estimate the expected future error that would result if we labeled some new instance x and added it to $\mathcal{L}$, and select the instance that minimizes that expectation. The idea is similar in spirit to the EGL strategy (Section 7.3.3), but differs in that we want to query for minimal expected future error, as opposed to maximal expected model change.

Roy and McCallum (2001) first proposed the estimated error reduction framework for text classification using naïve Bayes. Zhu et al. (2003) combined this framework with a semi-supervised learning approach (Section 7.10.1), resulting in a drastic improvement over random or uncertainty sampling. Guo and Greiner (2007) employ an "optimistic" variant that also biases the expectation toward the most likely label[1]. This strategy has the dual advantage of being near-optimal and not dependent on the model class. All that is required is an appropriate loss function and a way to estimate posterior label probabilities. For example, strategies in this framework have been successfully used with a variety of models including naïve Bayes (Roy and McCallum, 2001), Gaussian random fields (Zhu et al., 2003), logistic regression (Guo and Greiner, 2007), and support vector machines (Moskovitch et al., 2007).

Unfortunately, estimated error reduction may also be the most prohibitively expensive query selection framework. Not only does it require estimating the expected future error over $\mathcal{U}$ for each query, but a new model must be incrementally re-trained for each possible query labeling, which in turn iterates over the entire pool. This leads to a dramatic increase in computational cost. For some model classes such as Gaussian random fields (Zhu et al., 2003), the incremental training procedure is efficient and exact, making this approach fairly practical. For a many other model classes, this is not the case. For example, a binary logistic regression model would require $O(ULG)$ time complexity simply to choose the next query, where U is the size of the unlabeled pool $\mathcal{U}$, L is the size of the current training set $\mathcal{L}$, and G is the number of gradient computations required by the by optimization procedure until convergence. A classification task with three or more labels using

[1] Guo and Greiner refer to their strategy as maximizing "mutual information." However, their formulation is, in fact, equivalent to minimizing the expected future log-loss.

a MaxEnt model would require $O(M^2ULG)$ time complexity, where M is the number of class labels. For a sequence labeling task using CRFs, the complexity explodes to $O(TM^{T+2}ULG)$, where T is the length of an input sequence. Because of this, the applications of the estimated error reduction framework have mostly only considered simple binary classification tasks. Moreover, because the approach is often still impractical, some researchers have resorted to subsampling the pool $\mathcal{U}$ when selecting queries (Roy and McCallum, 2001) or using only approximate training techniques (Guo and Greiner, 2007).

7.3.6 Density-Weighting Methods

The information density (ID) framework presented by Settles and Craven (2008), and further analyzed in Chapter 4 of this thesis, is a density-weighting technique. The main idea is that informative instances should not only be those which are uncertain, but also those which are "representative" of the input distribution (i.e., inhabit dense regions of the input space). The formulation considered in this thesis was first published in Settles and Craven (2008), however it is not the first strategy to consider density and representativeness in the literature. McCallum and Nigam (1998b) also developed a density-weighted QBC approach for text classification with naïve Bayes, which is a special case of information density. Fujii et al. (1998) also explored a query strategy for nearest-neighbor methods that selects queries that are (i) unlike the labeled instances already in $\mathcal{L}$, and (ii) most similar to the unlabeled instances in $\mathcal{U}$. Nguyen and Smeulders (2004) have also proposed a density-based approach that first clusters instances and tries to avoid querying outliers by propagating label information to instances in the same cluster. Similarly, Xu et al. (2007) use clustering to construct sets of queries for batch-mode active learning (Section 7.7) with SVMs. Reported results in all these approaches are superior to methods that do not consider density or representativeness measures. Furthermore, Settles and Craven (2008) show that if densities can be pre-computed and estimated efficiently, the time required to select the next query is essentially no different than the base informativeness measure (e.g., uncertainty sampling).

7.4 Empirical Analysis

An important question is: does active learning actually work? Most empirical results in the literature suggest yes (e.g., Cohn et al., 1994; Thompson et al., 1999; Tong and Koller, 2000; Tur et al., 2005; Settles and Craven, 2008), however, there are caveats. First, consider that a training set built in cooperation with an active learner is inherently tied to the learning model that was used to generate it (i.e., the model selecting the queries). Therefore, the labeled instances are not drawn i.i.d. from the underlying data distribution. If one were to change models—as we often do in machine learning when the state of the art advances—this training set may no longer be as useful to the new model class. Baldridge and Osborne (2004) study these effects for several natural language parsers and suggest some techniques for better model generalization. Schein and Ungar (2007) also show that active learning can sometimes require more labeled instances than "passive" supervised learning for the *same* model, in their case logistic regression. Guo and Schuurmans

(2008) demonstrate that general active learning strategies, when employed in a batch-mode setting (Section 7.7) are also often much worse than random i.i.d. sampling. Anecdotally, however, active learning does reduce the number of labeled instances required to achieve a given level of accuracy in most reported cases (though this may be due to the publication bias).

7.5 Theoretical Analysis

A theoretical case for why and when active learning should work remains somewhat elusive, although there have been some recent advances. In particular, it would be nice to have some sort of bound on the number of queries required to learn a sufficiently accurate model for a given task, and theoretical guarantees that this number is less than in the passive supervised setting. Consider the following toy learning task to illustrate the potential of active learning. Suppose instances are points lying on a one-dimensional line, and our model class is a simple binary thresholding function g parameterized by θ:

$$g(x;\theta) = \begin{cases} 1 & \text{if } x > \theta, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

According to the *probably approximately correct* (PAC) learning model (Valiant, 1984), if the underlying data distribution can be perfectly classified by some hypothesis θ, then it is enough to draw $O(1/\epsilon)$ random labeled instances, where ϵ is the maximum desired error rate. Now consider an active learning setting, in which we can acquire the same number of *unlabeled* instances from this distribution for free. If we arrange these points on the real line, their (unknown) labels are a sequence of zeros followed by ones, and our goal is to quickly discover the location at which the transition occurs. By conducting a simple binary search through these unlabeled instances, a classifier with error less than ϵ can be achieved with a mere $O(\log 1/\epsilon)$ queries—since all other labels can be inferred—resulting in an exponential reduction in the number of labeled instances required. Of course, this is a one-dimensional, perfectly separable, noiseless, binary toy learning task. Generalizing this phenomenon to more interesting and realistic problem settings is the focus of much theoretical work in active learning.

Unfortunately, little more is known. There have been some fairly strong theoretical results for the membership query scenario, in which the learner is allowed to create query instances de novo and acquire their labels (Angluin, 1988, 2001). However, such instances can be difficult for humans to annotate (Baum and Lang, 1992) and may result in querying outliers, because they are not created according to an underlying natural distribution. Since a great many applications for active learning assume that unlabeled data (drawn from some natural distribution) are available, these results also have limited practical impact.

The main theoretical result to date in the stream-based and pool-based scenarios seems to be an analysis of the query-by-committee (QBC) algorithm by Freund et al. (1997). They show that, under certain assumptions, it is possible to achieve generalization error ϵ after seeing $O(d/\epsilon)$ unlabeled instances, where d is the *Vapnik-Chervonenkis (VC) dimension* (Vapnik and Chervonenkis, 1971) of the model space, and requesting only $O(d \log 1/\epsilon)$ labels. This, like the toy example

above, is an exponential improvement over the typical $O(d/\epsilon)$ sample complexity of the supervised setting. This result is tempered somewhat by the computational complexity of the QBC algorithm in practice, although Gilad-Bachrach et al. (2006) suggest some improvements by limiting the version space via kernel functions.

Dasgupta et al. (2005) propose a variant of the perceptron update rule which can achieve the same sample complexity bounds as reported for QBC, but for a single linear classifier. In earlier work, Dasgupta (2004) also provided a variety of theoretical upper and lower bounds for active learning in more general pool-based settings. In particular, if using linear classifiers the sample complexity can explode to $O(1/\epsilon)$ in the worst case, which offers no improvement over standard supervised learning, but is also no worse. However, Balcan et al. (2008) also show that, under an asymptotic setting, active learning is always better than supervised learning in the limit.

Most of these results have used theoretical frameworks similar to the standard PAC model, and necessarily assume that the learner knows the correct concept class in advance. Put another way, they assume that *some* model in our hypothesis class can perfectly classify the instances, and that the data are also noise-free. To address these limitations, there has been some more recent theoretical work in *agnostic active learning* (Balcan et al., 2006), which only requires that the unlabeled instances be drawn i.i.d. from a fixed distribution, and even noisy distributions are allowed. Hanneke (2007) extends this work by providing upper bounds on query complexity for the agnostic setting, and Dasgupta et al. (2008) propose a somewhat more efficient query selection algorithm. Cesa-Bianchi et al. (2005) have also shown that active learning is possible in the "regret" framework, also known as online adversarial learning.

However, most positive theoretical results to date have been based on intractable algorithms, or methods otherwise too prohibitively complex and particular to be used in practice. The few analyses performed on efficient algorithms have been with respect to uniform or near-uniform input distributions (Balcan et al., 2006; Dasgupta et al., 2005), or severely restricted hypothesis spaces. Furthermore, these studies have largely only been for simple (often binary) classification problems, with few implications for more complex models (e.g., that label structured instances like sequences and trees), which are central to many large-scale information extraction and management tasks addressed by the machine learning community today.

7.6 Structured Outputs

Active learning for classification tasks has been widely studied (Cohn et al., 1994; Zhang and Oles, 2000; Guo and Greiner, 2007). However, several important learning problems involve predicting structured outputs on instances, such as sequences and trees. Settles and Craven (2008) present and evaluate a large number of active learning algorithms for sequence labeling tasks using probabilistic sequence models like CRFs. Most of these algorithms can be generalized for use with other probabilistic sequence models, such as HMMs (Dagan and Engelson, 1995; Scheffer et al., 2001) and probabilistic context-free grammars (Baldridge and Osborne, 2004; Hwa, 2004). Thompson et al. (1999) also propose sampling strategies for structured tasks like semantic parsing and information extraction using inductive logic programming methods.

7.7 Batch-Mode Active Learning

In most of the active learning research, queries are selected in *serial*, i.e., one at a time. However, sometimes the training time required to induce a model is slow or expensive, as with large ensemble methods and many structured prediction tasks (see Section 7.6). Consider also that sometimes a distributed, parallel labeling environment may be available, e.g., multiple annotators working on different machines at the same time on a network. In both of these cases, selecting queries in serial may be inefficient. By contrast, *batch-mode* active learning allows the learner to query instances in groups, which is better suited to parallel labeling environments or models with slow training procedures.

Myopically querying the "N-best" queries according to a given instance-level query strategy often does not work well, since it fails to consider the overlap in information content among the "best" instances. To address this, a few batch-mode active learning algorithms have been proposed. Brinker (2003) considers an approach for SVMs that explicitly incorporates diversity among instances in the batch. Xu et al. (2007) propose a similar approach for SVM active learning, which also incorporates a density measure. Specifically, they query cluster centroids for instances that lie close to the decision boundary. Hoi et al. (2006a,b) extend the Fisher information framework (Section 7.3.4) to the batch-mode setting for binary logistic regression. Most of these approaches use greedy heuristics to ensure that instances in the batch are both diverse and informative, although Hoi et al. (2006b) exploit the properties of submodular functions to find near-optimal batches. Alternatively, Guo and Schuurmans (2008) treat batch construction for logistic regression as a discriminative optimization problem, and attempt to construct the most informative batch directly. For the most part, these approaches show improvements over random batch sampling, which in turn is generally better than simple "N-best" batch construction.

7.8 Active Learning With Costs

In some learning problems, the cost of acquiring labeled data can vary from one instance to the next. If our goal in active learning is to minimize the overall cost of training an accurate model, then reducing the number of labeled instances does not necessarily guarantee a reduction in overall labeling cost. One proposed approach for reducing annotation effort in active learning involves using the current trained model to assist in the labeling of query instances by pre-labeling them in structured learning tasks like parsing (Baldridge and Osborne, 2004) or information extraction (Culotta and McCallum, 2005). However, such methods do not actually represent or reason about labeling costs. Instead, they attempt to reduce cost indirectly by minimizing the number of annotation actions required for a query that has already been selected.

Another group of cost-sensitive active learning approaches explicitly accounts for varying label costs in active learning. Kapoor et al. (2007) propose one approach that takes into account both labeling costs and estimated misclassification costs. In this setting, each candidate query is evaluated by summing the labeling cost for the instance and the expected future misclassification costs that would be incurred if the instance were added to the training set. Instead of using real costs, however, their experiments make the simplifying assumption that the cost of labeling a

voicemail message is a linear function of its length (e.g., ten cents per second). King et al. (2004) use a similar active learning approach in an attempt to reduce actual labeling costs. They describe a "robot scientist" which can execute a series of autonomous biological experiments to discover metabolic pathways, with the objective of minimizing the cost of materials used (i.e., the cost of an experiment plus the expected total cost of future experiments until the correct hypothesis is found).

In both of the settings above, however, the cost of annotating an instance is assumed to be fixed and known to the learner before querying. Settles et al. (2008a) propose a novel approach to cost-sensitive active learning in settings where annotation costs are variable and *not* known, e.g., when the labeling cost is elapsed annotation time. They learn a regression cost-model alongside the active task-model which tries to predict the real (unknown) annotation cost based on a few simple "meta features" on the instances. Experiments show that such cost-models can learn to predict annotation times accurately, however further work is warranted to determine how such approximate, predicted labeling costs can be utilized effectively.

7.9 Alternative Query Types

To date, most work in active learning has assumed that a query unit is of the same type as the target concept to be learned. For example, if the task is to assign class labels to documents, the learner must query a document and obtains its label. What other forms might a query take?

Settles et al. (2008b) explore alternative query scenarios in the context of *multiple-instance active learning*. Here, the learner is sometimes allowed to query for labels at a finer granularity than the target concept, e.g., querying passages rather than entire documents, or segmented image regions rather than entire images. These experiments focus on active learning with the MI logistic regression model. Vijayanarasimhan and Grauman (2009) have extended this idea to SVMs for an image retrieval task, and also explore an approach that interleaves queries at these two granularities.

Raghavan et al. (2006) have explored a related idea called *tandem learning*, in which the learner is allowed to query for the labels of *features* as well as entire instances. They report not only that interleaving document- and word-level queries are very effective for a text classification problem, but also that words (features) are often much easier for human annotators to label in user studies.

7.10 Related Research Areas

Research in active learning is driven by two key ideas: (i) the learner should be allowed to ask questions, and (ii) unlabeled data are often readily available or easily obtained. There are a few related research areas with rich literature as well.

7.10.1 Semi-Supervised Learning

Active learning and *semi-supervised learning* (for a good introduction, see Zhu, 2005b) both traffic in making the most out of unlabeled data. As a result, there are a few conceptual overlaps between the two areas that are worth considering. For example, a very basic semi-supervised

technique is self-training (Yarowsky, 1995), in which the learner is first trained with a small amount of labeled data, and then used to classify the unlabeled data. Typically the *most* confident unlabeled instances, together with their predicted labels, are added to the training set, and the process repeats. A complementary technique in active learning is uncertainty sampling (see Section 7.3.1), where the instances about which the model is *least* confident are selected for querying.

Similarly, multi-view learning (de Sa, 1994) and co-training (Blum and Mitchell, 1998) use ensemble methods for semi-supervised learning. Initially, separate models are trained with the labeled data (usually using separate, conditionally independent feature sets), which then classify the unlabeled data, and "teach" the other models with a few unlabeled examples (with predicted labels) about which they are most confident. This helps to reduce the size of the version space, i.e., the models must agree on the unlabeled data as well as the labeled data. Query-by-committee (see Section 7.3.2) is an active learning compliment here, as the committee represents different parts of the version space, and is used to query the unlabeled instances about which they *dis*agree.

Through these illustrations, we begin to see that active learning and semi-supervised learning try to attack the same problem from different directions. Semi-supervised learning exploits what the learner thinks it already knows about the unlabeled data, and active learning attempts to explore the unknown aspects. It is therefore natural to think about combining the two. Some example formulations of semi-supervised active learning include McCallum and Nigam (1998b), Muslea et al. (2000), Zhu et al. (2003), Zhou et al. (2004b), and Tur et al. (2005).

7.10.2 Reinforcement Learning

In *reinforcement learning* (Sutton and Barto, 1998), the learner interacts with the world via "actions," and tries to find an optimal policy of behavior with respect to "rewards" it receives from the environment. For example, consider a machine that is learning how to play chess. In a supervised setting, one might provide the learner with board configurations from a database of chess games along with labels indicating which moves ultimately resulted in a win or loss. In a reinforcement setting, however, the machine actually plays the game against real or simulated opponents (Baxter et al., 2001). Each board configuration (state) allows for certain moves (actions), which result in rewards that are positive (e.g., capturing the opponent's queen) or negative (e.g., having its own queen taken). The learner aims to improve as it plays more games.

The relationship with active learning is that, in order to perform well, the learner must be proactive. It is easy to converge on a policy of actions that have worked well in the past but are sub-optimal or inflexible. In order to improve, a reinforcement learner must take risks and try out actions for which it is uncertain about the outcome, just as an active learner requests labels for instances it is uncertain how to label. This is often called the "exploration-exploitation" trade-off in the reinforcement learning literature. Furthermore, Mihalkova and Mooney (2006) consider an explicitly active reinforcement learning approach with aims to reduce the number of actions required to find an optimal policy.

7.10.3 Equivalence Query Learning

Another closely related research area is learning with *equivalence queries* (Angluin, 1988). Similar to membership query learning (see Section 7.2.1), here the learner is allowed to synthesize queries de novo. However, instead of generating *instances* to be labeled by the oracle, the learner instead generates a *hypothesis* of the target concept class, and the oracle either confirms or denies that the hypothesis is correct. If it is incorrect, the oracle should provide a counter-example, i.e., an instance that would be labeled differently by the true concept and the query hypothesis.

There seem to be few practical applications of equivalence query learning, because the oracle often does not know (or cannot provide) an exact description of the concept class for real-world problems. Otherwise, it would be sufficient to create an "expert system" by hand and machine learning is not required. However, it is an interesting intellectual exercise, and learning from combined membership and equivalence queries is in fact the basis of a popular inductive logic game titled Zendo[2].

7.10.4 Active Class Selection

Active learning assumes that instances are freely or inexpensively obtained, and it is the *labeling* process that incurs a cost. Imagine the opposite scenario, however, where a learner is allowed to query a known class label, and obtaining each *instance* incurs a cost. This fairly new problem setting is known as *active class selection*. Lomasky et al. (2007) propose several active class selection query algorithms for an "artificial nose" task, in which a machine learns to discriminate between different vapor types (the class labels) which must be chemically synthesized (to generate the instances). Some of their approaches show significant gains over uniform class sampling, the "passive" learning equivalent.

7.10.5 Active Feature Acquisition and Classification

In some learning domains, instances may have incomplete feature descriptions. For example, many data mining tasks in modern business are characterized by naturally incomplete customer data, due to reasons such as data ownership, client disclosure, or technological limitations. Consider a credit card company that wishes to model its most profitable customers; the company has access to data on client transactions using their own cards, but no data on transactions using cards from other companies. Here, the task of the model is to classify a customer using incomplete purchase information as the feature set. Similarly, consider a learning model used in medical diagnosis which has access to some patient symptom information, but not other symptoms that require complex or expensive procedures. Here, the task of the model is to suggest a diagnosis using incomplete symptom information as the feature set.

In these domains, *active feature acquisition* (Zheng and Padmanabhan, 2002; Melville et al., 2004) seeks to alleviate these problems by allowing the learner to request more complete feature information. The assumption is that some additional features can be obtained at a cost, such as

[2] http://www.wunderland.com/icehouse/Zendo/

leasing transaction records from other credit card companies, or running additional diagnostic procedures. The goal in active feature acquisition is to select the most informative features to obtain, rather than randomly or exhaustively acquiring all new features for all training instances. The difference between this learning setting and typical active learning is that these models request salient feature values rather than instance labels. Similarly, work in *active classification* (Greiner et al., 2002) considers the case in which features may be obtained during classification rather than training.

7.10.6 Model Parroting and Compression

Different machine learning algorithms possess different properties. In some cases, it is desirable to induce a model using one type of model class, and then "transfer" that model's knowledge to a model of a different class with another set of properties. For example, artificial neural networks have been shown to achieve better generalization accuracy than decision trees for many applications. However, decision trees represent symbolic hypotheses of the learned concept, and are therefore much more comprehensible to humans, who can inspect the logical rules and understand what the model has learned. Craven and Shavlik (1996) propose the TREPAN (Trees Parroting Networks) algorithm to extract highly accurate decision trees from trained artificial neural networks (or other incomprehensible model classes), providing comprehensible, symbolic interpretations. Buciluǎ et al. (2006) have adapted this idea to "compress" very large and computationally expensive model classes, such as complex ensembles, into smaller and more efficient model classes, such as neural networks.

These approaches can be thought of as active learning methods where the oracle is in fact another machine learning model (i.e., the one being parroted or compressed) rather than, say, a human annotator. In both cases, the "oracle model" can be trained using a small set of the available labeled data, and the "parrot model" is allowed to query the the oracle model for (i) the labels of any unlabeled data that is available, or (ii) synthesize new instances de novo. These two model parroting and compression approaches correspond to the pool-based and membership query scenarios for active learning, respectively.

Chapter 8

Additional Work in Biomedical Natural Language Processing

This chapter discusses other published original research that I have conducted during my graduate studies (Settles, 2004, 2005; Settles and Craven, 2005; Brow et al., 2006). While this work does not include active learning, it does involve machine learning and natural language processing for the biomedical literature using structured instance representations. Much of this other work helped motivate the active learning research previously described in this thesis.

8.1 Biomedical Named Entity Recognition

Recent efforts to organize the wealth of knowledge in biomedical literature have resulted in hundreds of databases and other resources (Bateman, 2008), providing scientists with access to structured biological information. However, with nearly half a million new research articles added to PubMed annually (Soteriades and Falagas, 2005), the sheer volume of publications and complexity of the knowledge to be extracted is beyond the means of most manual database curation efforts. As a result, many of these resources struggle to remain current. Automated information extraction, or at least automated assistance for such extraction tasks, seems a natural way to overcome these information management bottlenecks.

Named entity recognition (NER) is a subtask of information extraction, focused on finding mentions of various entities that belong to semantic classes of interest. In the biomedical domain, entities of interest are usually references to genes, proteins, cell types, and the like. Accurate NER systems are an important first step for many larger information management goals, such as automatic extraction of biologically relevant relationships (e.g., protein-protein interactions or sub-cellular location of gene products), biomedical document classification and retrieval, and ultimately the automatic maintenance of biomedical databases.

In order to facilitate and encourage research in the area of biomedical NER, several "bake-off" style competitions have been organized, in particular the NLPBA shared task (Kim et al., 2004) and the BioCreative challenges (Yeh et al., 2005; Smith et al., 2008). For these events, several research teams rapidly design, build, and submit results for machine learning systems using benchmark annotated text collections. Submissions are peer-reviewed, and the systems selected for publication showcase a variety of approaches to the problem, providing insight into what sorts of models and features are most effective.

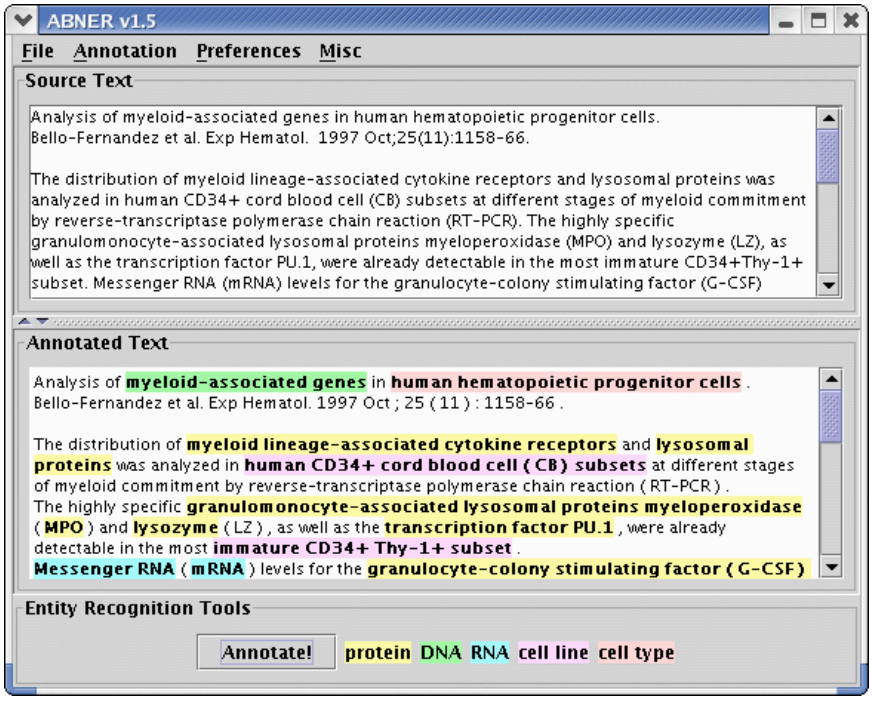


Figure 8.1: A screenshot of ABNER's graphical user interface.

8.1.1 The ABNER Software Tool

ABNER[1] stands for A Biomedical Named Entity Recognizer, which is a software tool that I developed for the named entity task in biomedical domains. I first released ABNER (Settles, 2005) in July 2004 as a demonstrational graphical user interface (GUI) for the system I developed as part of the NLPBA shared task challenge (Settles, 2004). In March 2005, a revised, open-source version of the software was released with some performance improvements and a new Java application programming interface (API). The goal is to encourage others to write custom interfaces to the core NER software, allowing it to be integrated into other, more sophisticated biomedical information management systems. ABNER also supports training new models on corpora labeled for different knowledge domains (e.g., specific model organisms, since gene naming conventions vary from species to species).

Figure 8.1 shows a screenshot of the intuitive GUI when ABNER is run as a stand-alone application. Text can be typed in manually or loaded from a file (top window), and then automatically tagged for multiple entities in real time (bottom window). Each entity type is highlighted with a unique color for easy visual reference, and tagged documents can be saved in a variety of annotated file formats. The application also has options for processing plain text documents on the file system in batch mode offline. ABNER has built-in functionality for tokenizing and segmenting sentences, which is fairly robust to line breaks and biomedical abbreviations (users can choose to bypass these features in favor of their own text preprocessing as well). The bundled ABNER

[1] http://pages.cs.wisc.edu/~bsettles/abner/

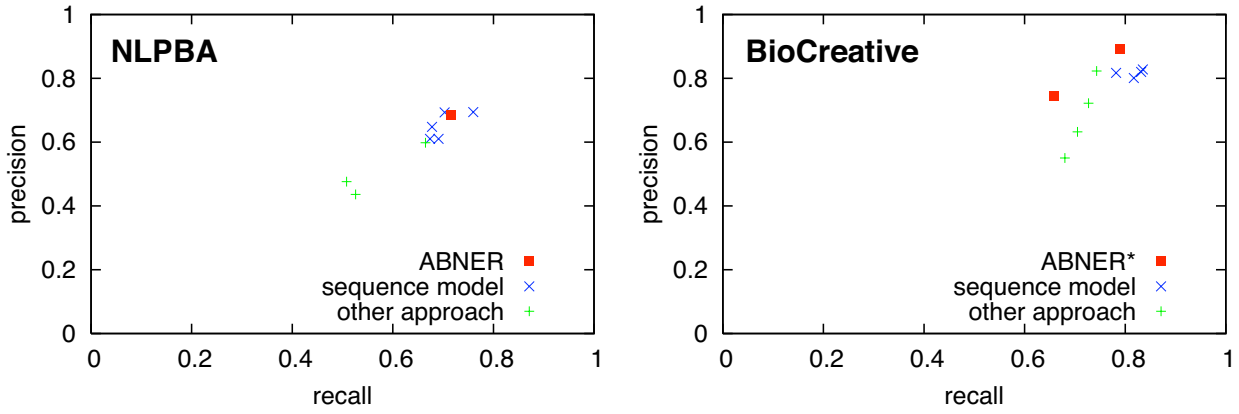


Figure 8.2: ABNER yields state-of-the-art results: (left) recall-precision points (over all entities) for ABNER vs. other published sequence-model and non-sequence approaches on the NLPBA corpus, (right) a similar comparison on the BioCreative corpus. A perfect system would be in the upper-right corner of the plot. For the BioCreative plot, other published systems are given credit for partial matches, whereas my evaluation of ABNER does not. The second, "better" point for ABNER is based on an extrapolation for better comparison with these systems.

application is implemented in Java and is therefore platform-independent, and has been tested on Linux, Solaris, Mac OS X, and Windows.

The basic ABNER distribution includes two built-in entity-tagging models trained on the **NLPBA** and **BioCreative** corpora. The first is a modified version of the GENIA corpus (Kim et al., 2003), containing five entity types (protein, DNA, RNA, cell-line and cell-type) labeled for 18,546 training sentences and 3,856 evaluation sentences. The latter corpus contains only one entity type that subsumes both genes and gene products labeled for 7,500 training sentences and 2,500 evaluation sentences. The underlying machine learning technology is based on a linear-chain CRF using the same feature set as the experiments in Chapters 3 and 4 on these corpora.

8.1.2 Experiments

Figure 8.2 presents recall-precision plots for the two built-in models compared against the results of other systems published as part of these task challenges, using the official training/evaluation splits of the data. ABNER ranked third in the official NLPBA evaluation in terms of F_1 measure. ABNER's accuracy is still roughly state-of-the-art today. To my knowledge, only two systems with published results have outperformed ABNER on the NLPBA corpus (Friedrich et al., 2006; Zhou and Su, 2004), and neither is freely available. Comparisons to published results on the BioCreative corpus are more difficult to interpret, as the ABNER results in Figure 8.2 reflect only perfectly accurate entity predictions (i.e., exact word-boundary matches), whereas the official BioCreative

evaluation gave some "partial credit" to incomplete entity extractions (Yeh et al., 2005). When adjusted for this, ABNER is competitive with the leading systems on this corpus as well, and is again the only freely available open-source system among them.

Third-party research also indicates that ABNER is among the most accurate publicly available NER tools for biomedical text. Kabiljo et al. (2007) performed a comparative analysis of three systems: ABNER (using the BioCreative model), GAPSCORE (Chang et al., 2004), and NLProt (Mika and Rost, 2004) on a new benchmark corpus called ProSpecTome, which is a subset of NLPBA re-annotated with more stringent labeling conventions. They found ABNER to be the most accurate on this new corpus by a significant margin. Lam et al. (2006) also conducted an informal comparison of ABNER (using the NLPBA model) to PowerBioNE (Zhou et al., 2004a) when deciding which to use as a component in their automated database maintenance system, and found ABNER to be consistently the best. Furthermore, most other systems are only available as web services or platform-specific compiled binaries, whereas ABNER is designed to be portable, flexible, and integrated into third-party biomedical NLP applications.

8.1.3 Beyond Named Entities

Accurate NER systems are an important first step for many larger information management goals. This subsection briefly discusses the impact ABNER has had in field the biomedical text processing by summarizing some of the published work by other researchers who use ABNER as a part of larger systems. These applications generally fall into three main categories: higher-level information extraction, text classification and information retrieval, and the automatic maintenance or curation of biological databases.

Higher-level information extraction

ABNER solves a basic subtask of more general information extraction needs, as it focuses only on finding entity mentions in text. Naturally, the next step is identifying the relationships among such entities directly from text. For example, in mining the biomedical literature this can mean extracting protein-protein interactions or identifying the sub-cellular localization of gene products.

Madkour et al. (2007) developed an extraction system for protein-protein interactions that employs ABNER in the protein identification phase. After proteins are annotated, the articles are mined using an unsupervised mutual reinforcement algorithm to rank textual patterns indicating an interaction relationship. They report an F_1 score of 0.55 on a corpus of MEDLINE abstracts, which appears to be near the current state-of-the-art for this formulation of the problem. To facilitate further progress in the area of extracting protein-protein interactions, a few variants of the task were proposed as part of the BioCreative2 challenge, and ABNER was also chosen as an NER component in at least four of the competing approaches (Abi-Haidar et al., 2007; Figueroa and Neumann, 2007; Gonzalez et al., 2007; Huang et al., 2007). BioCreative2 results are somewhat mixed, however, and substantially lower than those reported by Madkour et al.

Bethard et al. (2008) propose another interesting information extraction task that involves extracting semantic role arguments for protein transport predicates. Consider the following sentence:

"IRS-3 expression blocked glucose/IGF-1 induced IRS-2 translocation from the cytosol to the plasma membrane." They developed a system that attempts to automatically extract relational predicate records like TRANSLOCATION(*IRS-2*, *cytosol*, *plasma membrane*) from such passages of the biomedical literature. The extracted predicate name represents the type of protein transport, and the arguments correspond to the target protein and the sub-cellular source and destination locations of the transport action, respectively. The authors employ ABNER's protein predictions as part of the predicate extraction system, resulting in $F_1 = 0.792$ (compared to $F_1 = 0.841$ if protein mentions are already perfectly known).

Text classification and information retrieval

Most information retrieval systems aim to retrieve documents that are relevant to the user's particular information needs. Recently, however, interest has grown in developing systems that combine retrieval (particularly in the biomedical domain) with text classification and information extraction, attempting to answer user questions or put them in context, while providing supplementary information and linking to the original sources (Hersh et al., 2007).

Several researchers who work on these more sophisticated information retrieval systems have found that utilizing named entity predictions can improve their accuracy. For example, Tari et al. (2007) employ ABNER to process query topics in a "question and answer" style document retrieval system. The extracted entities are then matched against synonym lists in gene databases as part of a query-expansion step to improve recall. Another task, part of the Text Retrieval Conference (TREC) 2005 genomics track, involves filtering a set of documents for those which are appropriate for manual curation in four different biological databases. Several systems developed to solve this task (Li et al., 2007; Yang et al., 2006; Yu et al., 2006) use ABNER's entity predictions to enhance the feature set in this classification problem. Similarly, ABNER is used effectively by information retrieval systems designed to filter passages of text for mentions of protein-protein interactions (Abi-Haidar et al., 2007; Figueroa and Neumann, 2007; Huang et al., 2007).

Automatic maintenance of biological databases

Biological researchers often rely on specialized databases to maintain an in-depth repository of domain knowledge. For example, a database may only catalog information on a single, organism-specific genome, or functionally classified toxins and other chemicals. However, as indicated in the introduction, the rate of growth for new information to be mined from the primary literature or filtered from larger, general-purpose databases each year far eclipses the ability of curators to keep things up-to-date manually, even with a focused and specialized scope of interest.

Lam et al. (2006) present a novel system to address some of these issues, combining ABNER with a protein sequence motif extractor to automatically update special-interest databases. Entities are extracted from the textual fields of target database records (e.g., titles and abstracts or reference articles), and motifs are likewise extracted from the protein sequence fields (i.e., the actual amino acid sequences). The entity keywords and sequence motifs are then combined to generate queries for more general-purpose databases in the public domain, such as GenBank or SwissProt. The idea

is to filter the records from these broader interest databases and automatically extract the records that are relevant to the special-interest resources at hand. Their experiments in automatically maintaining a snake venom database achieve an $F_1 = 0.800$ using both ABNER keywords and sequence motifs (as opposed to $F_1 = 0.045$ and $F_1 = 0.410$, respectively, using either one in isolation).

Cakmak and Ozsoyoglu (2007) present another system that uses ABNER to extract gene mentions from the literature, and infer new function annotations from the Gene Ontology (GO Consortium, 2004) that may have been overlooked. The GO is a standardized vocabulary for molecular function of gene products used in most model organism genome databases. The resulting GO annotations can be appended to the extracted genes' database records automatically. They report F_1 scores of 0.66, 0.66, and 0.64 for the Biological Process, Molecular Function, and Cellular Component sub-ontologies, respectively.

8.1.4 Summary

The ABNER system, which I developed, is an efficient, accurate, cross-platform software tool for finding named entities in biomedical text. It has been demonstrated to perform at or near the state-of-the-art on multiple benchmark corpora, and remains one of the few high-accuracy NER systems available freely and under an open-source license at the time of this writing. It also ships with its own API, allowing users to re-train the underlying machine learning system for specific tasks, or to integrate it into larger, more sophisticated information management systems. ABNER has been used as a vital component in several such systems, including applications for higher-level information extraction, document classification and retrieval, and the automatic maintenance of biological databases.

However, the models ABNER ships with are trained with very large labeled corpora (e.g., 10,000+ sentences) that represent hundreds if not thousands of man-hours to annotate. Building accurate NER models for other knowledge domains, such as diseases, chemical compounds, or even gene products in other model organisms present significant annotation challenges. This is because such tasks often require building domain-specific training sets. As evidenced by Figure 8.2 (and a wealth of the related literature), approaches based on sequence models are state-of-the-art for these sorts of tasks. Thus, the need for understanding which active learning query strategies are most effective for such sequence models was the main motivation for the research in Chapter 3.

8.2 Document-Passage Relationships in Biomedical Text Classification

In some natural language processing tasks, such as text classification and information retrieval for certain knowledge domains, only a portion of the text is relevant to the task at hand. For example, most model organism databases such as the Mouse Genome Informatics (MGI) database (Eppig et al., 2005) employ teams of PhD-level biologists to read the scientific literature and then manually enter relevant information into the databases. It is our conjecture that curating genes for functional activity likely depends on a small set of passages rather than an entire document.

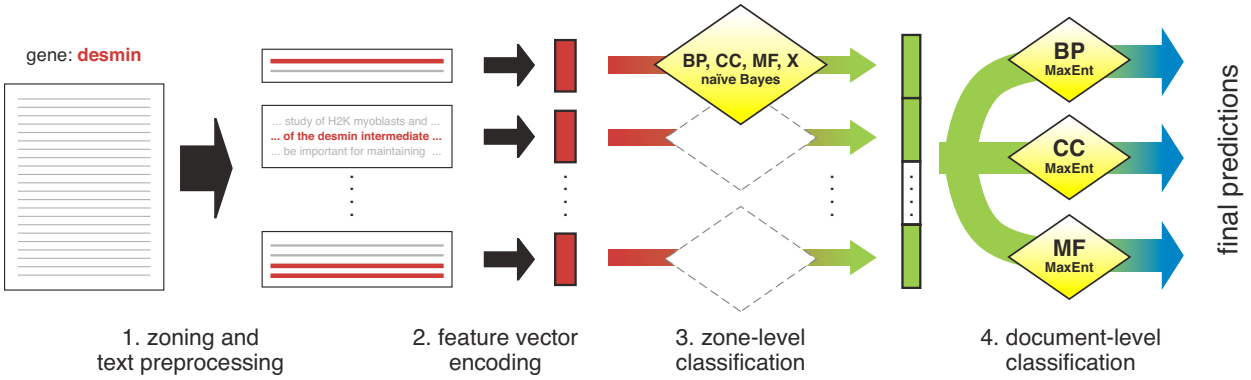


Figure 8.3: A two-tier system for the TREC 2004 classification task.

The Genomic Track portion of the Text Retrieval Conference (TREC) in 2004 and 2005 contained text classification tasks aimed at investigating methods that might help human curators filter the scientific literature and identify articles relevant to the curation process. In 2004, this task involved taking document-gene tuples $\langle d, g \rangle$ and annotating them with functional information from the Gene Ontology (GO Consortium, 2004). Specifically, systems had to determine which, if any, of the three GO subdomains—Biological Process (BP), Cellular Component (CC), and Molecular Function (MF)—are applicable to gene g according to the text in document d. Note that these labels are not mutually exclusive. In 2005, the classification task involved taking the full text of a scientific article and triaging them in to pools to be curated for each four categories: GO annotation (GO), the Alleles and Phenotypes category of the Mouse Genome Database (Allele), the Gene Expression Database (Expression), and the Mouse Tumor Biology Database (Tumor). Again, these categories are not mutually exclusive, thus an article may be classified into any number of categories from zero (should not be curated for any database) to four (should be curated for them all).

In both tasks, labels are provided at a document-level resolution. My approaches to these tasks have been methods that involve (i) basing classifications on selected passages from articles, and (ii) adjusting the classifier training process such that certain putatively relevant passages affect the learned model more than other passages. I consider baseline methods that make classifications by considering the entire text of each article, and that are trained by equally weighting all parts of all training articles.

8.2.1 TREC 2004 Experiments

For the TREC 2004 task (functional document-gene GO annotation), the classification approach consists of four main steps (Settles and Craven, 2005). First, each document d is partitioned into six sections: *title*, *abstract*, *introduction*, *methods*, *results*, and *discussion*. Sentences with mentions of gene g are identified (using regular expressions and MGI gene/protein synonym lists) within each section. Second, feature vectors for each section are constructed using only the

sentences where gene g is mentioned. Third, label predictions are made for each section independently by a multi-class multinomial naïve Bayes model (McCallum and Nigam, 1998a) trained for four labels: BP, CC, MF, and null. These posterior probabilities are then concatenated to create a secondary document-level feature vector with 24 features (4 labels $\times$ 6 document sections). Finally, predictions for each GO domain annotation are made by three binary MaxEnt classifier models (Berger et al., 1996): one for each domain. Features for the second step consist of words as well as syntactic rules (patterns) and "informative terms" that are statistically induced from the training texts. An illustration of the system is presented in Figure 8.3.

During the development of this system, I ran several experiments using four-fold cross-validation on the training set (correct labels for the evaluation data were not provided until after the conference). Official evaluation for the system is in terms of precision, recall, and F_1. First, I investigated the validity of using (localized) section information in classification. The left side of Figure 8.4 compares recall-precision curves for three systems: (i) a baseline naïve Bayes model trained on words from the entire document and ranked by posterior probability, (ii) a similar model divided into zones (each zone submitting a "vote" for classification weighted by its posterior probability), and (iii) the two-tier classification illustrated in Figure 8.3. From this we can see that moving from a full-text classification model to a simple voting system that incorporates local section information has a slight impact: sacrificing precision at low levels of recall, but maintaining higher precision at high levels of recall. However, moving on to the two-tier system significantly improves precision across the whole recall spectrum. The right side of Figure 8.4 compares our two-tier system (using various feature sets) with all other runs submitted for evaluation. Our system performed quite well and the best run achieved an F_1 score of 51.38 (the official metric of comparison), which ranked it second best among all participating systems.

The accuracy of our two-tier system, our first venture into incorporating localized decisions, is promising. The major goals of this study were to investigate the value of (i) section information, and (ii) advanced features not often used in such text classification tasks (specifically syntactic features and GO-specific informative terms; see Settles and Craven (2005) for more details). This early work suggests that both facets can offer substantial gains over our baseline approach for this problem.

8.2.2 TREC 2005 Experiments

For the TREC 2005 task[2]—which requires triaging documents for curation in the Allele, Expression, GO, or Tumor databases—we experimented with several different methods for making localized classification decisions (Brow et al., 2006). We investigated three main approaches: the first two augment the classification procedure to exploit document-passage relationships, while the third augments the training procedure.

In the first approach, we segment full-text scientific articles into individual paragraphs and classify them individually. The posterior probabilities of each paragraph belonging to the positive

[2]This subsection describes joint work with Thomas Brow.

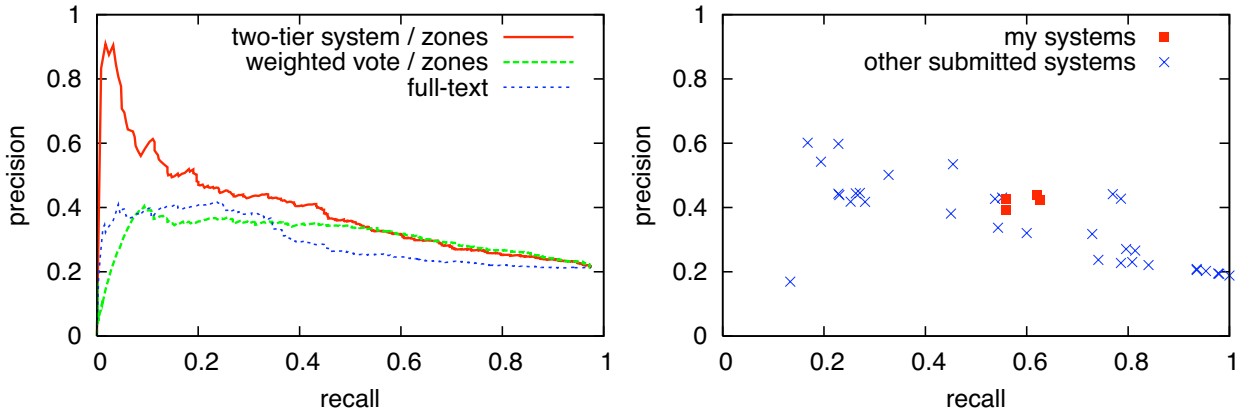


Figure 8.4: TREC 2004 results in recall-precision space: (left) curves comparing use of zone information in various ways, (right) official evaluation results for four variants of my system vs. all other submitted systems.

class are then aggregated into a document-level classification by taking the N-best most probable paragraphs and averaging them.

Representing the collection of paragraph-level probabilities by their mean discards information about the *distribution* of those probabilities. We hypothesize that the shape of the paragraph-probability distribution on individual documents can be recognized as belonging to positive or negative instances. To take advantage of this difference in distributions, we train a secondary statistical model to discriminate the two. The feature vector for this "metaclassifier" is generated by using an integer-valued feature to represent each bin in a discrete representation of the distribution (i.e., a histogram). The value of the feature is the count of paragraphs that have probabilities in the corresponding interval.

Another way of localizing the classifier is to focus the model's attention to important passages during the training process itself. One way of accomplishing this is by employing the *expectation-maximization* (EM) algorithm (Dempster et al., 1977), which is an approach to finding likelihood estimates for parameters in probabilistic settings with hidden variables. In our setting, the *hidden variables* represent the extent to which individual paragraphs should be treated as positive instances during training (i.e., an instance weight). We employ one hidden variable for each paragraph in positive documents. (We assume that all paragraphs in a negative document really are negative, and thus there are no hidden variables for these cases.) In the *E-step*, we use the current model to estimate the probability that each paragraph in a given document is positive (contains text relevant to the document being positive), and assign weights to paragraph instances equal to the model's output. Occasionally, there may be no paragraphs that appear positive for a given training document that is known to be positive. To test and correct for this, we sum the model's output for all paragraphs in a document. If the sum is less than some threshold δ, we re-normalize weights

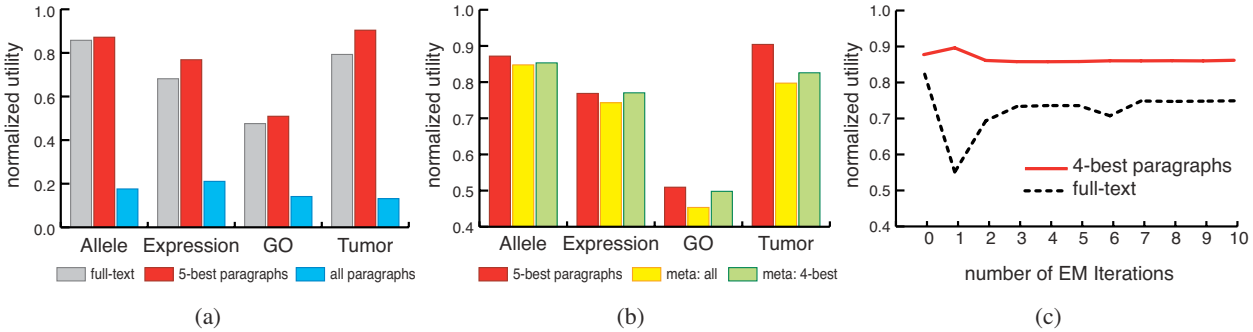


Figure 8.5: TREC 2005 results. (a) Results for models that make classifications with the paragraph-based approach. (b) Metaclassifier results compared to simply averaging the N-best. (c) The effects of ten EM iterations on utility for the Allele task (similar results for the other categories). Classifiers are trained using EM-weighted paragraphs and evaluated against both full-text and N-best paragraphs.

to sum to δ. The assumption here is that a positive document has at least δ paragraphs that are relevant to its class.

Experiments during system development employ MaxEnt classifiers (Berger et al., 1996) and use four-fold cross-validation on the training data. Features that describe paragraph/article text consist simply of words (or the "bag of words" representation). As in the official TREC evaluation, we measure classifier performance by computing *normalized utility* $= \eta(fp) + \varepsilon(tp)$, where fp and tp are the counts of false positives and true positives, respectively (see Section 2.3 for an explanation of these numbers). The coefficients η and ε are category-specific weights (or "relative utilities") defined to account for the varying number of positive instances across categories. These weights are defined $\eta = -1$ and $\varepsilon = \frac{an}{ap}$, where an and ap are the total counts of actual negative and positive instances, respectively.

Figure 8.5(a) presents the results for the paragraph-level classifier approach. Here, we see that the N-best classifier approaches provide better utility than the baseline models for all tasks. This result supports our hypothesis that only certain passages are relevant. The results for the all-paragraphs control (e.g., N-best with N equal to the total number of paragraphs) indicate that success of the N-best method is due to its focus on a small number of paragraphs, rather than some other aspect of its paragraph-based representation. Figure 8.5(b) compares the performance of the metaclassifier approach, both with and without paragraph selection, to the simple mean N-best paragraphs approach. The distribution metaclassifier does not result in higher utility scores, suggesting that the metaclassifier approach is susceptible to over-fitting. Figure 8.5(c) shows the classification utility realized by the N-best and full-text classification methods as a function of the number of EM training iterations (with normalization threshold $\delta = 2$). Utility generally drops immediately after EM re-weighting begins, and while subsequent iterations show gradual improvement, the models appear to converge before reaching even the initial model's utility. This result indicates that the EM algorithm, as used here, either is not effective at identifying the most

Table 8.1: MILR vs. the two best classifiers from our TREC 2005 experiments. Reported is the utility score of each approach. For each subtask, the best result is shown in bold.

Task	Allele	Expression	GO	Tumor
MILR	0.8462	**0.7725**	0.4181	**0.9103**
5-best	**0.8636**	0.7617	**0.5023**	0.8774
full-text	0.8323	0.6107	0.4407	0.7325

relevant paragraphs or that there is no benefit in doing so. We also note that the N-best method of evaluation outperforms the full-text method in this context as well.

Our first hypothesis was that we can achieve more accurate classifications by basing decisions on selected paragraphs in test articles. This was well supported by our experiments. The second hypothesis was that we could achieve more accurate classifications by employing a rich representation of predicted paragraph-level class probabilities. The third hypothesis was that we could learn more accurate models by having the training process put more emphasis on some paragraphs than others. Neither of these latter two hypotheses were supported by our experimental results.

Notice, however, that our assumptions in the EM-based training procedure are consistent with the multiple-instance (MI) problem setting presented in Section 2.2.2 and explored in detail in Chapter 5. As an alternative to EM training, I have subsequently run additional experiments on these four subtasks using MI logistic regression (MILR). Table 8.1 compares utility scores for MILR against the two best methods from our TREC 2005 paper: the 5-best paragraph model, and the baseline full-text model. Here we see that MILR is the best method for two of the four subtasks, and outperforms the full-text baseline on the Allele task. We can conclude from these results that, even though passage-focused learning was not beneficial using the EM approach described above, it can be useful when representing the problem in the MI setting.

8.2.3 Summary

Our experiments in both TREC 2004 and 2005 indicate that not all parts of a document are relevant to certain text classification tasks, and that exploiting passage-level information can yield more accurate models. Additionally, I have shown that the MI learning representation is well-suited to passage-level learning in TREC 2005 classification tasks. However, ambiguity still remains about *which* passages are most relevant to the task at hand during training. This is partially what motivated the work in MI active learning discussed in Chapter 5.

Chapter 9

Conclusion

In today's information-rich digital world, supervised machine learning approaches are used extensively to solve real-world challenges in organizing, extracting, and retrieving information from large data sources. However, these learning algorithms have a non-trivial limitation: they require labeled training data which is often difficult, time-consuming, or expensive to obtain. For many of these tasks, inputs take the form of structured instances which can make the labeling and learning tasks more complicated.

This thesis has focused on effective active learning approaches for such problem domains, in which instances take structured form and labeling costs may vary. The methods introduced in this thesis aim to reduce the overall cost of acquiring labeled data by allowing the learner to effectively choose the instances on which it is trained. In this final chapter, I summarize the specific contributions (and limitations) of this work, and discuss several open research directions aimed at better utilizing data and labeling resources for machine learning problems through active learning.

9.1 Summary of Contributions

This thesis has made several contributions to the state of the art in active learning. Specific contributions include:

- *An analysis of active learning with sequences.* One contribution of this thesis is a large-scale analysis of active learning strategies for sequence labeling tasks like information extraction. The empirical study presented in Chapter 3 expands the frontier of query selection strategies designed for sequence models into new and previously unexplored active learning frameworks. These results reveal which approaches are most effective and computationally feasible (for interactive settings), including the information density algorithm, which was introduced in this work. Additionally, my results show that query strategies should consider each sequence as a whole, rather than aggregating token-level information, and that informativeness measures should not be normalized for sequence length (as it has often been previously assumed).

- *Multiple-instance active learning.* Another major contribution of this work is a novel approach to active learning in multiple-instance (MI) learning domains. The methods pioneered in this thesis allow MI learners to reduce the inherent ambiguity of the MI representation by learning from labels at mixed levels of granularity: both bag-level (coarse, but inexpensive) and instance-level (fine, but more expensive). The query algorithms introduced in Chapter 5 can be used to reduce overall annotation costs by actively querying the most informative instances from positive bags. These results have implications for several real-world learning applications, such as classifying and retrieving text documents and images. Future work in this area includes exploring alternate scenarios, such as interleaving queries at various levels of granularity.

- *A new approach to cost-sensitive active learning.* The study presented in Chapter 6 answers several important questions about the nature of real-world annotation costs that are unknown to the learner when selecting queries, a setting that has not been previously considered in the literature. Specifically, I show that (i) the labeling time required by human annotators is often not (approximately) constant from one instance to the next, (ii) that it can vary significantly from one annotator to another, and that (iii) it sometimes contains a stochastic component. I reason about the causes and implications of these findings, and demonstrate that in some cases we can learn to accurately predict future unknown annotation costs, even after only a few queries have been labeled. While the actual utility of these cost predictions is thus far inconclusive, this approach to active learning holds promise for reducing actual annotation costs in domains where they are not known. Developing robust cost-sensitive active learning approaches given approximate, predicted cost information is a key direction for future work.

- *Two new general active learning frameworks.* I have proposed two novel approaches to pool-based active learning: the information density algorithm, and the expected gradient length (EGL) framework. The information density algorithm improves upon a simple "base" strategy by selecting queries that are not only informative, but also representative of the input distribution. I present formulations for both sequence labeling (Chapters 3 and 4) and classification (Chapter 4) tasks. Extensive evaluation demonstrates that information density is a state-of-the-art strategy for both problem settings. Future work includes developing fast and accurate approximations to the density calculation, improved similarity measures for sequence tasks, and generalizations to batch-mode active learning. The EGL framework differs in that it attempts to select queries based on how much they will influence the model if we knew their instance labels. I show that this approach can be effective in sequence labeling (Chapter 3) and multiple-instance (Chapter 5) applications. However, for more complex problems this method may require approximations or be prohibitively slow in practice.

- *State-of-the-art approaches to biomedical natural language processing.* In Chapter 8, I presented ABNER, a state-of-the-art named entity recognition system I have developed and released as an open-source tool for the biomedical natural language processing community, which (at the time of this writing) has generated more than 4,000 downloads and over 150 citations to date. Many researchers in related areas of biomedical information extraction,

information retrieval, and automatic database curation use ABNER to improve and enhance their own machine learning systems. I also demonstrated that biomedical text classification systems can often do better by making decisions at, and learning from, the granularity of passages. This result may be further exploited by the multiple-instance active learning presented here.

9.2 Open Problems and Future Work

Throughout my work on active learning, I have encountered many practical challenges and interesting empirical results, which have inspired several ideas for novel ways of looking at active learning. This section introduces some of these ideas and problem settings, which I feel are fruitful directions for future work.

- *Multi-task active learning*. The vast majority of active learning research has assumed that there is only one learner trying to solve a single task. Consider the CKB system from Chapter 6, however, which actually comprises two learners solving two subtasks: entity recognition and relation extraction. For many real-world problems, the same data can be labeled multiple ways for different subtasks. In such cases, it is likely most economical to label a single instance for all subtasks simultaneously. Therefore, *multi-task active learning* algorithms should assume that a single query will be labeled for multiple tasks, and assess the "informativeness" of an instance with respect to all the learners involved. In the case of the CKB corpus, the two tasks are related (i.e., the relations take entities as arguments) but they need not be. Consider a database of film reviews, which we might use to build a system that (i) extracts the names of key actors and production crew, (ii) classifies the film by genre, and (iii) predicts a rating based on the text. Such a system would probably employ three independent learners: a sequence model for entity extraction, a classifier for genres, and a regression model to predict ratings. Effectively selecting queries that benefit all three of these learners is an open and potentially important direction for active learning.

- *Alternative query types*. Results from Chapter 5 show that it is possible (and indeed beneficial) for multiple-instance active learners to request labels at the instance granularity as well as bags, which is the typical unit of labeling. Recently, methods have been developed to combine unlabeled data with "domain knowledge" in the form of feature labels (Druck et al., 2008) or known label distributions (Mann and McCallum, 2007a). Combining such approaches with active learning holds much promise for reducing overall labeling costs involving human annotators. Consider an active text classifier that, instead of querying the annotator with documents to be labeled, queries instead with words. These words may even be "softly" labeled, e.g., GAME as a feature might indicate the baseball class 60% of the time and the hockey class 40% of the time, but autos and motorcycles none of the time. Similarly, the learner might ask the annotator if he or she knows roughly what the distribution of class labels in the overall corpus should be. As we develop new forms of supervision in machine learning, we also open the door to alternative forms of active learning.

- *Interactive learning*. Much of the work in this thesis has been focused on systems that learn interactively. For example, Chapter 3 provides a run-time analysis of query strategies to determine which methods are most practical when the oracle provides immediate feedback. The work presented in Chapter 6 attempts to predict the amount of time a specific annotator will require to label queries, with the goal of exploiting that knowledge in real-time. Another direction for *interactive learning* is to combine active query strategies with novel label acquisition methods such as "games with a purpose" (von Ahn and Dabbish, 2008), which aim to gather instance labels through Internet-enabled community games. A major limitation for many learning tasks today is the time and availability of domain experts. However, there is some recent evidence that aggregating redundant labelings from non-experts can be equally effective for machine learning (Snow et al., 2008). Thus, acquiring multiple, inexpensive, non-expert labels via online games for instances that have been actively selected by the learner is a promising new direction for active learning.

- *Active transfer learning*. It is often the case that large training sets are available for one learning task, but not for a different (but similar) task. For example, the BioCreative corpus from Chapters 3 and 4 is annotated for gene and protein mentions in the genomics literature involving human cells. To my knowledge, however, no such training sets exist for the rat or mouse genomics literature. *Transfer learning* (Torrey and Shavlik, 2009) is a subfield of machine learning that involves transferring knowledge from one domain to expedite learning in another, e.g., taking what is known about information extraction in the human genomics literature and applying it to rat or mouse knowledge domains. Combining this approach with active learning holds the potential for active, rapid adaptation of mature "source" learning problems to new "destination" problem domains.

9.3 Last Words

In this thesis, I have explored active learning in a variety of real-world problem domains characterized by complex structured instances and potentially varied annotation costs. This work has helped to answer outstanding questions about active learning in some of these applications, e.g., "what methods are best for sequence labeling tasks?" and "do annotation times vary among instances or annotators?" At the same time, this work has also introduced several new ways of thinking about active learning in other domains, e.g., mixed-granularity queries for multiple-instance learning, and predicting unknown future annotation costs for cost-sensitive active learning. It is my hope that the research findings I have presented here will serve as a foundation for future work in active learning applied to real-world learning problems.

Bibliography

N. Abe and H. Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1–9. Morgan Kaufmann, 1998.

A. Abi-Haidar, J. Kaur, A. Maguitman, P. Radivojac, A. Retchsteiner, K. Verspoor, Z. Wang, and L.M. Rocha. Uncovering protein-protein interactions in the bibliome. In *Proceedings of the BioCreative2 Workshop*, pages 247–255, 2007.

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, pages 561–568. MIT Press, 2003.

D. Angluin. Queries revisited. In *Proceedings of the International Conference on Algorithmic Learning Theory*, pages 12–31. Springer-Verlag, 2001.

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

M.F. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 65–72. ACM Press, 2006.

M.F. Balcan, S. Hanneke, and J. Wortman. The true sample complexity of active learning. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 45–56. Springer, 2008.

J. Baldridge and M. Osborne. Active learning and the total cost of annotation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9–16. ACL Press, 2004.

A. Bateman. Editorial. *Nucleic Acids Research*, 36(Database issue):D1, 2008.

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

J. Baxter, A. Tridgell, and L. Weaver. Reinforcement learning and chess. In J. Furnkranz and M. Kubat, editors, *Machines that Learn to Play Games*, pages 91–116. Nova Science Publishers, 2001.

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

S. Bethard, Z. Lu, J.H. Martin, and L. Hunter. Semantic role labeling for protein transport predicates. *BMC Bioinformatics*, 9:277, 2008.

F.R. Blattner, G. Plunkett, C.A. Bloch, N.T. Perna, V. Burland, M. Riley, J. Collado-Vides, J.D. Glasner, C.K. Rode, G.F. Mayhew, J. Gregor, N.W. Davis, H.A. Kirkpatrick, M.A. Goeden, D.J. Rose, B. Mau, and Y. Shao. The complete genome sequence of Escherichia coli K-12. *Science*, 277:1453–1474, 1997.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 92–100. Morgan Kaufmann, 1998.

C. Bonwell and J. Eison. *Active Learning: Creating Excitement in the Classroom*. Jossey-Bass, 1991.

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

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

T. Brow, B. Settles, and M. Craven. Classifying biomedical articles by making localized decisions. In *Proceedings of the Text Retrieval Conference (TREC)*, 2006.

C. Buciluǎ, R. Caruana, and A. Niculescu-Mizil. Model compression. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 535–541. ACM Press, 2006.

A. Cakmak and G. Ozsoyoglu. Annotating genes using textual patterns. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, volume 12, pages 221–232. World Scientific Press, 2007.

V.R. Carvalho and W. Cohen. Learning to extract signature and reply lines from email. In *Proceedings of the Conference on Email and Anti-Spam (CEAS)*, 2004.

N. Cesa-Bianchi, C. Gentile, A. Tironi, and L. Zaniboni. Worst-case analysis of selective sampling for linear-threshold algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, volume 17, pages 233–240. MIT Press, 2005.

J.T. Chang, H. Schütze, and R.B. Altman. GAPSCORE: finding gene and protein names one word at a time. *Bioinformatics*, 20(2):216–225, 2004.

O. Chapelle, P. Haffner, and V.N. Vapnik. Support vector machines for histogram-based image classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

D. Cohn. Neural network exploration using optimal experiment design. In *Advances in Neural Information Processing Systems (NIPS)*, volume 6, pages 679–686. Morgan Kaufmann, 1994.

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

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

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

M. Craven and J. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems (NIPS)*, volume 8, pages 24–30. MIT Press, 1996.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 509–516. AAAI Press, 1998.

A. Culotta and A. McCallum. Reducing labeling effort for stuctured prediction tasks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 746–751. AAAI Press, 2005.

I. Dagan and S. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 150–157. Morgan Kaufmann, 1995.

S. Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 337–344. MIT Press, 2004.

S. Dasgupta, A. Kalai, and C. Monteleoni. Analysis of perceptron-based active learning. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 249–263. Springer, 2005.

S. Dasgupta, D. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 353–360. MIT Press, 2008.

V.R. de Sa. Learning classification with unlabeled data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 6, pages 112–119. MIT Press, 1994.

A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38, 1977.

T. Dietterich, R. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997.

G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–602. ACM Press, 2008.

R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

J.T. Eppig, C.J. Bult, J.A. Kadin, J.E. Richardson, J.A. Blake, and the members of the Mouse Genome Database Group. The Mouse Genome Database (MGD): from genes to mice–a community resource for mouse biology. *Nucleic Acids Research*, 33:D471–D475, 2005. `http://www.informatics.jax.org`.

V. Federov. *Theory of Optimal Experiments*. Academic Press, 1972.

A. Figueroa and G. Neumann. Identifying protein-protein interactions in biomedical publications. In *Proceedings of the BioCreative2 Workshop*, pages 217–225, 2007.

Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

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

C.M. Friedrich, T. Revillion, M. Hofmann, and J. Fluck. Biomedical and chemical named entity recognition with conditional random fields: The advantage of dictionary features. In *Proceedings of the International Symposium on Semantic Mining in Biomedicine (SMBM)*, pages 85–89, 2006.

A. Fujii, T. Tokunaga, K. Inui, and H. Tanaka. Selective sampling for example-based word sense disambiguation. *Computational Linguistics*, 24(4):573–597, 1998.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

R. Gilad-Bachrach, A. Navot, and N. Tishby. Query by committee made real. In *Advances in Neural Information Processing Systems (NIPS)*, volume 18, pages 443–450. MIT Press, 2006.

The GO Consortium. The Gene Ontology (GO) database and informatics resource. *Nucleic Acids Research*, 32:D258–D261, 2004. `http://www.geneontology.org`.

T. Golub, D. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. Mesirov, H. Coller, M. Loh, J. Downing, M. Caligiuri, C. Bloomfield, and E. Lander. Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537, 1999.

G. Gonzalez, L. Tari, A. Gitter, R. Leaman, S. Nikkila, R. Wendt, A. Zeigler, and C. Baral. Integrating knowledge extracted from biomedical literature: Normalization and evidence statements for interactions. In *Proceedings of the BioCreative2 Workshop*, pages 227–235, 2007.

R. Greiner, A. Grove, and D. Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139:137–174, 2002.

Y. Guo and R. Greiner. Optimistic active learning using mutual information. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 823–829. AAAI Press, 2007.

Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems (NIPS)*, number 20, pages 593–600. MIT Press, Cambridge, MA, 2008.

S. Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 353–360. ACM Press, 2007.

A. Hauptmann, W. Lin, R. Yan, J. Yang, and M.Y. Chen. Extreme video retrieval: joint maximization of human and computer performance. In *Proceedings of the ACM Workshop on Multimedia Image Retrieval*, pages 385–394. ACM Press, 2006.

D. Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4(1):7–40, 1994.

W. Hersh, A.M. Cohen, P. Roberts, and H.K. Rekapalli. Trec 2006 genomics track overview. In *Proceedings of the Text Retrieval Conference (TREC)*, 2007.

W.R. Hersh, C. Buckley, T.J. Leone, and D.H. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201. ACM Press, 1994.

S.C.H. Hoi, R. Jin, and M.R. Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the International Conference on the World Wide Web*, pages 633–642. ACM Press, 2006a.

S.C.H. Hoi, R. Jin, J. Zhu, and M.R. Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 417–424. ACM Press, 2006b.

D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2137–2144. IEEE Press, 2006.

A. Huang, S. Ding, H. Wang, and X. Zhu. Mining physical protein-protein interactions from literature. In *Proceedings of the BioCreative2 Workshop*, 2007.

J. Hull. A database for handwriting recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994.

R. Hwa. Sample selection for statistical parsing. *Computational Linguistics*, 30(3):73–77, 2004.

R Kabiljo, D Stoycheva, and AJ Shepard. ProSpecTome: A new tagged corpus for protein named entity recognition. In *Proceedings of the ISMB BioLINK*, pages 24–27. Oxford University Press, 2007.

A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning,. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 877–882. AAAI Press, 2007.

J. Kim, T. Ohta, Y. Teteisi, and J. Tsujii. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl. 1):i180–i182, 2003.

J. Kim, T. Ohta, Y. Tsuruoka, Y. Tateisi, and N. Collier. Introduction to the bio-entity recognition task at JNLPBA. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 70–75, 2004.

S. Kim, Y. Song, K. Kim, J.W. Cha, and G.G. Lee. MMR-based active machine learning for bio named entity recognition. In *Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 69–72. ACL Press, 2006.

R.D. King, K.E. Whelan, F.M. Jones, P.G. Reiser, C.H. Bryant, S.H. Muggleton, D.B. Kell, and S.G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–52, 2004.

V. Krishnamurthy. Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Transactions on Signal Processing*, 50(6):1382–1397, 2002.

S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 282–289. Morgan Kaufmann, 2001.

K. Lam, J.L.Y. Koh, B. Veeravalli, and V. Brusic. Incremental maintenance of biological databases using association rule mining. In S. Istrail, P. Pevzner, and M. Waterman, editors, *Pattern Recognition in Bioinformatics*, pages 140–150. Springer, 2006.

K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 331–339. Morgan Kaufmann, 1995.

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

L. Lee. Measures of distributional similarity. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 25–32. ACL Press, 1999.

D. Lewis and J. Catlett. Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 148–156. Morgan Kaufmann, 1994.

D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. ACM/Springer, 1994.

D. Lewis and M. Ringuette. A comparison of two learning algorithms for text categorization. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, pages 81–93, 1994.

T. Li, M. Ogihara, and Q. Li. A comparative study on content-based music genre classification. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 282–289. ACM Press, 2003.

Y. Li, H. Lin, and Z. Yang. Two approaches for biomedical text classification. In *Proceedings of the International Conference Bioinformatics and Biomedical Engineering (ICBBE)*, pages 310–313. IEEE Press, 2007.

M. Light, X.Y. Qiu, and P. Srinivasan. The language of bioscience: Facts, speculations, and statements in between. In *Proceedings of the ISMB BioLINK*, pages 17–24. ACM Press, 2004.

M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for nearest neighbor classifiers. *Machine Learning*, 54(2):125–152, 2004.

D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization methods. *Mathematical Programming*, 45:503–528, 1989.

Y. Liu. Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of Chemical Information and Computer Sciences*, 44:1936–1941, 2004.

R. Lomasky, C.E. Brodley, M. Aernecke, D. Walt, and M. Friedl. Active class selection. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 640–647. Springer, 2007.

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

A. Madkour, K. Darwish, H. Hassan, A. Hassan, and O. Emam. BioNoculars: Extracting protein-protein interatctions from biomedical text. In *BioNLP 2007: Biological, translational, and clinical language processing*, pages 89–96. ACM Press, 2007.

O. Mangasarian, W.N. Street, and W. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, 1995.

G. Mann and A. McCallum. Simple, robust, scalable semi-supervized learning via expectation regularization. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 593–600. ACM Press, 2007a.

G. Mann and A. McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of the North American Association for Computational Linguistics (NAACL)*, pages 109–112. ACL Press, 2007b.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

D. Margineantu. Active cost-sensitive learning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1622–1623. AAAI Press, 2005.

O. Maron and T. Lozano-Perez. A framework for multiple-instance learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 10, pages 570–576. MIT Press, 1998.

A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the AAAI Workshop on Learning for Text Categorization*, pages 41–48, 1998a.

A. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367. Morgan Kaufmann, 1998b.

P. Melville and R. Mooney. Diverse ensembles for active learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 584–591. Morgan Kaufmann, 2004.

P. Melville, M. Saar-Tsechansky, F. Provost, and R. Mooney. Active feature-value acquisition for classifier induction. In *Proceedings of the IEEE Conference on Data Mining (ICDM)*, pages 483–486. IEEE Press, 2004.

L. Mihalkova and R. Mooney. Using active relocation to aid reinforcement learning. In *Proccedings of the Florida Artificial Intelligence Research Society (FLAIRS)*, pages 580–585. AAAI Press, 2006.

S. Mika and B. Rost. Protein names precisely peeled off free text. *Bioinformatics*, 20(suppl 1): I241–I247, 2004.

T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

R. Moskovitch, N. Nissim, D. Stopel, C. Feher, R. Englert, and Y. Elovici. Improving the detection of unknown computer worms activity using active learning. In *Proceedings of the German Conference on AI*, pages 489–493. Springer, 2007.

I. Muslea, S. Minton, and C.A. Knoblock. Selective sampling with redundant views. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 621–626. AAAI Press, 2000.

H.T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 79–86. ACM Press, 2004.

J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 1999.

M. Nyffenegger, J.C. Chappelier, and E. Gaussier. Revisiting Fisher kernels for document similarities. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 727–734. Springer, 2006.

G. Paass and J. Kindermann. Bayesian query construction for neural network models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 7, pages 443–450. MIT Press, 1995.

F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 329–336. ACL Press, 2004.

F. Provost, T. Fawcett, and R. Kohavi. Building the case against accuracy estimation for comparing induction algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 445–453. Morgan Kaufmann, 1998.

L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

H. Raghavan, O. Madani, and R. Jones. Active learning with feedback on both features and instances. *Journal of Machine Learning Research*, 7:1655–1686, 2006.

R. Rahmani and S.A. Goldman. MISSL: Multiple-instance semi-supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 705–712. ACM Press, 2006.

L.A. Ramshaw and M.P. Marcus. Text chunking using transformation-based learning. In *Proceedings of the ACL Workshop on Very Large Corpora*, 1995.

S. Ray and M. Craven. Supervised versus multiple instance learning: An empirical comparison. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 697–704. ACM Press, 2005.

N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 441–448. Morgan Kaufmann, 2001.

E.F.T.K. Sang and F. DeMeulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, pages 142–147, 2003.

T. Scheffer, C. Decomain, and S. Wrobel. Active hidden Markov models for information extraction. In *Proceedings of the International Conference on Advances in Intelligent Data Analysis (CAIDA)*, pages 309–318. Springer-Verlag, 2001.

A.I. Schein and L.H. Ungar. Active learning for logistic regression: An evaluation. *Machine Learning*, 68(3):235–265, 2007.

M. Schena, D. Shalong, R. Davis, and P.O. Brown. Quantitaive monitoring of gene expression patterns with a complimentary DNA microarray. *Science*, 270:467–470, 1995.

M.J. Schervish. *Theory of Statistics*. Springer, 1995.

R. Schwartz and Y.-L. Chow. The N-best algorithm: an efficient and exact procedure for finding the N most likely sentence hypotheses. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 81–83. IEEE Press, 1990.

B. Settles. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192, 2005.

B. Settles. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 104–107, 2004.

B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1069–1078. ACL Press, 2008.

B. Settles and M. Craven. Exploiting zone information, syntactic features, and informative terms in gene ontology annotation from biomedical documents. In *Proceedings of the Text Retrieval Conference (TREC)*, 2005.

B. Settles, M. Craven, and L. Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS Workshop on Cost-Sensitive Learning*, pages 1–10, 2008a.

B. Settles, M. Craven, and S. Ray. Multiple-instance active learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 20, pages 1289–1296. MIT Press, 2008b.

H.S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology and the North American Association for Computational Linguistics (HLT-NAACL)*, pages 213–220. ACL Press, 2003.

C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27: 379–423,623–656, 1948.

L. Smith, L.K. Tanabe, R.J. Ando, C.J. Kuo, I.F. Chung, C.N. Hsu, Y.S. Lin, R. Klinger, C.M. Friedrich, K. Ganchev, M. Torii, H. Liu, B. Haddow, C.A. Struble, R.J. Povinelli, A. Vlachos, W.A. Baumgartner Jr, L. Hunter, B. Carpenter, R.T. Tsai, H.J. Dai, F. Liu, Y. Chen, C. Sun, S. Katrenko, P. Adriaans, C. Blaschke, R. Torres, M. Neves, P. Nakov, A. Divoli, M.M. López, J. Mata, and W.J. Wilbur. Overview of BioCreative II gene mention recognition. *Genome Biology*, 9(Suppl 2):S2, 2008.

A.J. Smola and B. Schölkopf. A tutorial on support vector regression. Technical Report NC2-TR-1998-030, NueroCOLT2 Technical Report Series, 1998.

R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast—but is it good? In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263. ACM Press, 2008.

E.S. Soteriades and M.E. Falagas. Comparison of amount of biomedical research originating from th euriopean union and the united states. *British Medical Journal*, 331:192–194, 2005.

C. Sutton and A. McCallum. An introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006.

R. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Q. Tao, S.D. Scott, and N.V. Vinodchandran. SVM-based generalized multiple-instance learning via approximate box counting. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 779–806. Morgan Kaufmann, 2004.

L. Tari, G. Gonzalez, R. Leaman, S. Nikkila, R. Wendt, and C. Baral. ASU at TREC 2006 genomics track. In *Proceedings of the Text Retrieval Conference (TREC)*, 2007.

C.A. Thompson, M.E. Califf, and R.J. Mooney. Active learning for natural language parsing and information extraction. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 406–414. Morgan Kaufmann, 1999.

S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *Proceedings of the ACM International Conference on Multimedia*, pages 107–118. ACM Press, 2001.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 999–1006. Morgan Kaufmann, 2000.

L. Torrey and J. Shavlik. Transfer learning. In E. Soria, J. Martin, R. Magdalena, M. Martinez, and A. Serrano, editors, *Handbook of Research on Machine Learning Applications*. To appear, 2009.

G. Tur, D. Hakkani-Tür, and R.E. Schapire. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication*, 45(2):171–186, 2005.

C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M.N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T.M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.W. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

V.N. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, 16:264–280, 1971.

S. Vijayanarasimhan and K. Grauman. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21. MIT Press, 2009.

A. Vlachos. Evaluating and combining biomedical named entity recognition systems. In *BioNLP 2007: Biological, translational, and clinical language processing*, pages 199–206, 2007.

L. von Ahn and L. Dabbish. General techniques for designing games with a purpose. *Communications of the ACM*, 51(8):58–67, 2008.

D.G. Wang, J.B. Fan, C.J. Siao, A. Berno, P. Young, R. Sapolsky, G. Ghandour, N. Perkins, E. Winchester, J. Spencer, L. Kruglyak, L. Stein, L. Hsie, T. Topaloglou, E. Hubbell, E. Robinson, M. Mittmann, M.S. Morris, N. Shen, D. Kilburn, J. Rioux, C. Nusbaum, S. Rozen, T.J. Hudson, R. Lipshutz, M. Chee, and E.S. Lander. Large-scale identification, mapping, and genotyping of single-nucleotide polymorphisms in the human genome. *Science*, 280(5366):1077–1082, 1998.

Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 246–257. Springer-Verlag, 2007.

R. Yan, J. Yang, and A. Hauptmann. Automatically labeling video data using multi-class active learning. In *Proceedings of the International Conference on Computer Vision*, pages 516–523. IEEE Press, 2003.

Z. Yang, H. Lin, Y. Li, B Liu, and Y. Lu. TREC 2005 genomics track experiments at DUTAI. In *Proceedings of the Text Retrieval Conference (TREC)*, 2006.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 189–196. ACL Press, 1995.

A. Yeh, A. Morgan, M. Colosimo, and L. Hirschman. Biocreative task 1a: gene mention finding evaluation. *BMC Bioinformatics*, 6(Suppl 1):S2, 2005.

H. Yu. SVM selective sampling for ranking with application to data retrieval. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 354–363. ACM Press, 2005.

L. Yu, S.T. Ahmed, G. Gonzalez, B. Logsdon, M. Nakamura, S. Nikkila, K. Shah, L. Tari, R. Wendt, A. Ziegler, and C Baral. Genomic information retrieval through selective extraction and tagging by the ASU-BoiAI group. In *Proceedings of the Text Retrieval Conference (TREC)*, 2006.

C. Zhang and T. Chen. An active learning framework for content based information retrieval. *IEEE Transactions on Multimedia*, 4(2):260–268, 2002.

T. Zhang and F.J. Oles. A probability analysis on the value of unlabeled data for classification problems. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1191–1198. Morgan Kaufmann, 2000.

Z. Zheng and B. Padmanabhan. On active learning for data acquisition. In *Proceedings of the IEEE Conference on Data Mining (ICDM)*, pages 562–569. IEEE Press, 2002.

G. Zhou and J. Su. Exploring deep knowledge resources in biomedical name recognition. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications (NLPBA)*, pages 96–99, 2004.

G.D. Zhou, J. Zhang, J. Su, D. Shen, and C.L. Tan. Recognizing names in biomedical texts: A machine learning approach. *Bioinformatics*, 20(7):1178–1190, 2004a.

Z.H. Zhou, K.J. Chen, and Y. Jiang. Exploiting unlabeled data in content-based image retrieval. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 425–435. Springer, 2004b.

X. Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005a.

X. Zhu. Semi-supervised learning literature survey. Computer Sciences Technical Report 1530, University of Wisconsin–Madison, 2005b.

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised 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.

APPENDIX
Implementation Notes

This appendix provides additional details on the implementation of the learning algorithms discussed in this thesis. Logistic regression, maximum entropy (MaxEnt) models, conditional random fields (CRFs), and MI logistic regression (MILR) are all machine learning models in the exponential family. In all experiments, they are trained by maximizing log-likelihood $\ell(\mathcal{L};\theta)$ via gradient optimization, using the quasi-Newton method L-BFGS (Liu and Nocedal, 1989). Additionally, I use $\sigma = 1$ in the regularization term during training for all models.

CRFs, MaxEnt Models, and Naïve Bayes

The implementations for CRFs (Chapters 3, 4, 6, and 8), MaxEnt models (Chapters 4, 6, and 8), and naïve Bayes (Chapter 8) used for experiments in this thesis come from a modified version of MALLET[1] (Machine Learning for Language Toolkit). This is an open-source library of Java code for various natural language tasks, which I have customized by implementing the necessary methods for active learning strategies and passage-level classification and learning functions.

Logistic Regression and MILR

The implementations of simple logistic regression (Chapter 2) and MILR (Chapter 5) discussed in this thesis use AMIL[2] (Active Multiple-Instance Library). This is an open-source library of Java code that I have written and released for multiple-instance learning, and MI active learning in particular. This library was used for the Diverse Density (DD) results reported in Table A.1 as well. For simple binary logistic regression experiments, MILR was employed with data in a multiple-instance representation, but using only one instance per bag.

Although the logarithm in the log-likelihood objective function ℓ alleviates most floating-point machine precision errors, the instance-level or bag-level output probabilities for MILR and other Diverse Density models can still occasionally be zero or one. This means that we are sometimes faced with computing the log or inverse of zero. Following Maron and Lozano-Perez (1998), my implementation smooths $0 \equiv 10^{-7}$ in such cases.

[1] http://mallet.cs.umass.edu/

[2] http://pages.cs.wisc.edu/~bsettles/amil/

MILR Gradient Calculations

Here I discuss how to compute the gradient $\nabla \ell$ for training MILR, and for use with the "expected gradient length" family of query selection strategies. As stated in Section 5.3.1, the training gradient $\nabla \ell(\mathcal{L}; \theta) = [\frac{\partial \ell}{\partial \theta_0}, \ldots, \frac{\partial \ell}{\partial \theta_K}]$ is a vector whose components are the partial derivatives of the log-likelihood objective function ℓ from Equation 2.5 with respect to each model parameter θ_k. Using the chain rule, each partial derivative can be expressed as:

$$\frac{\partial \ell}{\partial \theta_k} = \sum_{l=1}^{L} \frac{\partial \ell}{\partial o^{(l)}} \sum_{n=1}^{N} \left(\frac{\partial o^{(l)}}{\partial o_n} \frac{\partial o_n}{\partial \theta_k} \right) - \frac{\theta_k}{\sigma^2},$$

where $o^{(l)}$ is the output probability for the bag $\mathcal{X}^{(l)}$ under the current model, N is the number of instances in that bag, and o_n is the respective output probability for each instance $x_n \in \mathcal{X}^{(l)}$. Hereafter, I drop the explicit (l) superscript to simplify the notation. The first term $\frac{\partial \ell}{\partial o}$ is the derivative of the composite log-likelihood (2.5), which yields:

$$\frac{\partial \ell}{\partial o} = \frac{o - y}{o^2 - o} = \begin{cases} o^{-1} & \text{if } y = 1 \text{ (the bag is positive), and} \\ (o-1)^{-1} & \text{if } y = 0 \text{ (the bag is negative).} \end{cases}$$

The next term $\frac{\partial o}{\partial o_n}$ is the derivative of the softmax combining function (2.4). This is also used as the relevance-weighting term in the MIU query strategy (5.2), computed as follows:

$$\frac{\partial o}{\partial o_n} = \frac{(1 + \alpha\, o_n - \alpha\, o) \exp(\alpha\, o_n)}{\sum_{i=1}^{N} \exp(\alpha\, o_i)}.$$

The final term $\frac{\partial o_n}{\partial \theta_k}$ is the derivative of the instance-level logistic function (2.3), given by:

$$\frac{\partial o_n}{\partial \theta_k} = o_n(1 - o_n) f_k(x_n).$$

Comparison of MI Learning Algorithms

Table A.1 presents a detailed empirical comparison of three MI learning algorithms on all of the data sets described in Secion 5.4. MILR$_\ell$ is the formulation of MI logistic regression used in the experiments from Section 5.5, optimizing the regularized log-likelihood objective function (whose gradient calculations are given above). MILR$_S$ is the squared-loss formulation previously considered in the literature (Settles et al., 2008b; Ray and Craven, 2005). DD refers to the original Diverse Density algorithm formulation, which uses a Gaussian instance model with a noisy-or combining function (Maron and Lozano-Perez, 1998). The results are averaged over 20 independent runs, each using 20 bags (ten positive, ten negative) for training and the rest for evaluation. Because MILR$_S$ and DD are more prone to becoming stuck in local optima, I report results for the model that best fits the training data after ten random restarts. Note that MILR$_\ell$ is at an implicit disadvantage, because its results are using only a single model (no restarts), yet it performs best on over half the tasks and requires significantly less time to train.

Table A.1: Comparison of three MI learning algorithms: MILR_ℓ (MI logistic regression trained using a regularized log-likelihood objective function), MILR_S (trained using squared loss), and DD (Diverse Density). AUROC scores are shown for each algorithm for each task, with the best algorithm indicated in bold.

TASK	MILR_ℓ	MILR_S	DD	TASK	MILR_ℓ	MILR_S	DD
zero	**0.976**	0.476	0.651	musk1	**0.646**	0.536	0.624
one	**0.730**	0.492	0.630	musk2	**0.619**	0.474	0.563
two	**0.904**	0.497	0.639				
three	**0.762**	0.496	0.635	trx	0.618	0.581	**0.667**
four	**0.890**	0.486	0.646				
five	**0.811**	0.487	0.632	tst1	0.922	**0.941**	0.693
six	**0.897**	0.485	0.637	tst2	0.652	**0.728**	0.612
seven	**0.790**	0.488	0.641	tst3	0.755	**0.826**	0.564
eight	0.551	0.489	**0.632**	tst4	0.771	**0.851**	0.638
nine	**0.676**	0.488	0.627	tst7	0.750	**0.796**	0.661
				tst9	0.626	**0.664**	0.605
ajaxorange	**0.653**	0.559	0.573	tst10	0.786	**0.820**	0.645
apple	**0.416**	0.370	0.413				
banana	**0.534**	0.419	0.447	alt.atheism	0.537	**0.551**	0.522
bluescrunge	**0.616**	0.350	0.321	comp.graphics	0.556	0.560	**0.582**
candlewithholder	0.664	**0.692**	0.486	comp.os.ms-windows	0.576	**0.644**	0.493
cardboardbox	0.452	0.409	**0.494**	comp.sys.ibm.pc.hardware	0.576	**0.636**	0.467
checkeredscarf	0.799	**0.835**	0.668	comp.sys.mac.hardware	0.482	0.535	**0.567**
cokecan	**0.799**	0.742	0.555	comp.windows.x	0.562	**0.590**	0.535
dataminingbook	**0.504**	0.430	0.485	misc.forsale	0.554	**0.650**	0.544
dirtyrunningshoe	0.724	**0.773**	0.594	rec.autos	0.546	**0.691**	0.565
dirtyworkgloves	**0.617**	0.488	0.476	rec.motorcycles	**0.554**	0.496	0.529
fabricsoftenerbox	**0.680**	0.482	0.588	rec.sport.baseball	0.553	**0.567**	0.542
feltflowerrug	**0.787**	0.783	0.586	rec.sport.hockey	**0.634**	**0.634**	0.459
glazedwoodpot	**0.454**	0.420	0.452	sci.crypt	**0.671**	0.560	0.532
goldmedal	**0.690**	0.481	0.479	sci.electronics	0.421	**0.516**	0.436
greenteabox	**0.762**	0.523	0.629	sci.med	0.531	**0.568**	**0.568**
juliespot	**0.647**	0.457	0.522	sci.space	**0.592**	0.579	0.547
largespoon	0.391	**0.396**	0.383	soc.religion.christian	0.686	**0.688**	0.474
rapbook	0.438	0.433	**0.491**	talk.politics.guns	0.544	**0.626**	0.512
smileyfacedoll	**0.737**	0.620	0.458	talk.politics.mideast	**0.723**	0.657	0.494
spritecan	**0.758**	0.715	0.526	talk.politics.misc	0.568	**0.594**	0.503
stripednotebook	0.493	0.446	**0.501**	talk.religion.misc	**0.647**	0.601	0.574
translucentbowl	**0.749**	0.492	0.475				
wd40can	**0.773**	0.684	0.550				
woodrollingpin	**0.444**	0.418	0.377				

Index