2013-08-12

Practical Cost-Conscious Active Learning for Data Annotation in Annotator-Initiated Environments

Robbie A. Haertel
Brigham Young University - Provo

Practical Cost-Conscious Active Learning for Data Annotation in Annotator-Initiated Environments

Robbie A. Haertel

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Eric Karl Ringger, Chair
Kevin Darrell Seppi
Christophe Gerard Giraud-Carrier
Michael D. Jones
Kent Eldon Seamons

Department of Computer Science

Brigham Young University

August 2013

ABSTRACT

Practical Cost-Conscious Active Learning for Data Annotation in
Annotator-Initiated Environments

Robbie A. Haertel
Department of Computer Science, BYU
Doctor of Philosophy

Many projects exist whose purpose is to augment raw data with annotations that increase the usefulness of the data. The number of these projects is rapidly growing and in the age of "big data" the amount of data to be annotated is likewise growing within each project. One common use of such data is in supervised machine learning, which requires labeled data to train a predictive model. Annotation is often a very expensive proposition, particularly for structured data. The purpose of this dissertation is to explore methods of reducing the cost of creating such data sets, including annotated text corpora.

We focus on active learning to address the annotation problem. Active learning employs models trained using machine learning to identify instances in the data that are most informative and least costly. We introduce novel techniques for adapting vanilla active learning to situations wherein data instances are of varying benefit and cost, annotators request work "on-demand," and there are multiple, fallible annotators of differing levels of accuracy and cost. In order to account for data instances of varying cost, we build a model of cost from real annotation data based on a user study. We also introduce a novel cost-conscious active learning algorithm which we call return-on-investment, that selects instances for annotation that contain the most benefit per unit cost. To address the issue of annotators that request instances "on-demand," we develop a parallel, "no-wait" framework that performs computation while the annotator is annotating. As a result, annotators need not wait for the computer to determine the best instance for them to annotate—a common problem with existing approaches. Finally, we introduce a Bayesian model designed to simultaneously infer ground truth annotations from noisy annotations, infer each individual annotators accuracy, and predict its own accuracy on unseen data, without the use of a held-out set. We extend ROI-based active learning and our annotation framework to handle multiple annotators using this model. As a whole, our work shows that the techniques introduced in this dissertation reduce the cost of annotation in scenarios that are more true-to-life than previous research.

Keywords: active learning, cost-sensitive learning, machine learning, return-on-investment, Bayesian models, parallel active learning, natural language processing, part-of-speech tagging

ACKNOWLEDGMENTS

Nanos gigantum humeris insidentes, "Dwarves standing on the shoulders of giants,"[1] is an old metaphor typically used to refer to the fact that new research is always built upon a much larger body of existing research. While this is certainly the case, another interpretation exists. Namely, a scientist is unable to perform his research without the enormous assistance, aid, and support of many others. The following are a sampling of some of the giants that carried me while I was working towards my degree; my apologies in advance for any of those that I have not mentioned by name—know that I am appreciative of all those who have assisted in any way.

First and foremost, I would like to thank Dr. Eric Ringger, my advisor. I am ever grateful that, in his first year at Brigham Young University, he took a chance on a student of Linguistics interested in natural language processing by inviting me to do a doctorate under his tutelage. He has procured funding for me and has taught me more than I could have imagined through his classes, our research, and other interactions. More importantly, he has taught me how to perform research so that I may continue to learn and discover new things. He has as always been very supportive, inspiring, positive, uplifting, and, most of all, patient, even when I am sure I did not make these things easy. He has also deftly handled the administrative aspects of my degree.

I am also very grateful for Dr. Kevin Seppi, my second committee member. He has gone well beyond the call of duty of a second committee member and in my mind he is really a co-advisor. I have learned volumes from his classes and our interactions. Unlike the stereotypical professor, Dr. Seppi was always in the trenches: like Dr. Ringger, he was always either present physically or available by phone and email late into the evenings of paper deadlines. I am thankful for his support and encouragement.

Likewise, my full Ph.D. committee has been very supportive. They supported me in my decisions to take time off for internships and also helped ensure I finished my dissertation after leaving my studies to work full time.

[1] Translation taken from Wikipedia [97], which contains an interesting discussion about the history and use of the phrase.

My other co-authors have been incredibly helpful and insightful, including, but not limited to: Dr. James Carroll's, Paul Felt, George Busby, Peter McClanahan, Marc Carmen and Dr. Deryle Lonsdale. Although we have not (yet) been co-authors, I am also very grateful for meaningful and stimulating discussions with Dr. Daniel Walker IV.

I very much enjoyed discussions I had with my colleagues at conferences and workshops that undoubtedly shaped my views of active learning. In particular, I would like to thank Dr. Burr Settles for his kind feedback of a draft of Chapter 10 and for sharing his thoughts and ideas about all things active learning. I also had several fruitful conversations with Dr. Katrin Tomanek on the subject of cost-conscious active learning. In addition, I am thankful for her assistance as co-organizer of the Active Learning Workshop for NLP, 2010. Others who influenced me through our conversations and interactions include Dr. Michael Bloodgood and Dr. Kevin Small.

Completion of my degree would not have been possible without the aid of financial support and other resources. The Computer Science department has been very generous in this regard, providing funding for my research for all but one year. For that year, I am grateful to Microsoft who kindly provided me with a Mentor Grant. Brigham Young University also provides computing resources free of charge via the Fulton Supercomputing Lab, without which, none of this research would have been completed. Of course, I am also extremely grateful to Google for the experience gained on both of my paid internships; Robert Gardner, Max Lin, and Gideon Mann were fabulous hosts who helped me reach my potential and complete successful internships. During this last year while working on my dissertation while a full-time employee of Google, management has been very accommodating in allowing me to finish; it truly was a priority for those I work with and those above me to finish. While here at Google, I have completed some parts of the dissertation using company provided equipment.

Last, but certainly not least, I would like to thank my family. I am most grateful for the support and sacrifice proffered me by the love of my life, my beautiful and dear wife of nearly eleven years, Meri. She has made incredible sacrifices and has willingly taken upon herself extra burdens at home to allow me time to finish my dissertation. This dissertation simply would not have

been possible without her, for which I will be eternally grateful. This degree is as much hers as it is mine and I am so blessed to be married to her and to walk the journey of life by her side and with her help.

While a graduate student, all four of my children, Jared, Alex, Nathan, and my little princess Caroline, have been born. They, too, have been very patient with me through this process. Finally, I would like to thank my parents. They have provided continuous love and support for their, "eternal student."

Table of Contents

Chapter 1

Introduction

We live in a truly exciting time in which photos uploaded to the Internet are automatically tagged with the names of the people in them, cell phones facilitate communication between people that speak different languages, and cars drive themselves."Smart apps" are increasingly becoming a part of everyday life and these apps are driven by machine learning. Frequently, the models that give modern technology its smarts require large amounts of data that have been directly or indirectly annotated. Not surprisingly, efforts to manually annotate data can be time-consuming and costly. The ultimate goal of this dissertation is to produce novel machine learning techniques that will reduce the cost of producing annotated corpora in practice.

To give an idea of the cost required to annotate corpora in today's technological landscape, consider the Comprehensive Corpus of Classical Syriac, being undertaken by the Neal A. Maxwell Institute for Religious Scholarship—well-known for their publication of the digitized and morphologically annotated edition of the Dead Sea Scrolls—in conjunction with the Oriental Institute at Oxford. Augmenting the Syriac corpus with detailed morphological annotations requires trained experts, of which there are relatively few, and manual annotation of all 50 million tokens would be prohibitively costly. As a point of reference, The Way International Foundation spent 15 years annotating the New Testament portion of the Peshitta (100,000 words) with morphological annotations, similar to those the Maxwell Institute will employ [48]. Similarly, it has taken 2 years for two Syriac scholars to label approximately one-fourth of the Old Testament. The Maxwell Institute plans to use computational assistance in order to increase the amount of data it can annotate within its budget (or, if possible, reduce the cost of annotating all of the data).

The Syriac project is exemplary of the many projects which have sought or seek to compile an annotated corpus/dataset as cheaply as possible. Some other examples include: Palmer et al. [64] and Baldridge and Palmer [7], who are interested in language documentation (collecting, analyzing, and translating texts from dying languages for the purpose of preserving them); Tomanek et al. [91], who are interested in named entity recognition; Marcus et al. [56], who were interested in building a large corpus of syntactically parsed and part-of-speech tagged English texts; and so forth. In fact, governments and large companies are increasingly adopting machine learning techniques to accomplish complex tasks including speech recognition, face recognition, object recognition, trip planning, driver-less cars, etc. Many of the algorithms used in these techniques rely on annotated (i.e., labeled) data. Therefore, techniques that reduce the cost of obtaining annotated data will have the economic impact of facilitating a diversity of new projects, improving existing projects, and allowing for more complex applications.

Specifically, this dissertation focuses on the use of active learning, a technique for incrementally annotating data that intelligently selects the next "best" instance for annotation according to some specified criterion. We introduce novel techniques for adapting active learning to situations where:

- data instances to be annotated are of varying benefit and cost
- annotators request work "on-demand" (annotator-initiated) rather than the computer sending work to the annotators on its time schedule (learner-initiated).
- there are multiple, fallible annotators of differing levels of accuracy and cost

Although these scenarios are common across multiple disciplines (e.g., machine vision, bioinformatics, etc.) and a broad range of tasks, this dissertation focuses on natural language processing. Specifically, we focus our empirical results on the part-of-speech tagging task (described later), although our theoretical considerations are more generally applicable. The techniques contained in this dissertation reduce the total cost of annotating instances under these realistic settings.

In the next chapter, we provide a gentle introduction to the field of cost-conscious active learning, including defining pertinent terms. We do so in the context of a survey of existing literature, with a particular focus on cost-conscious active learning techniques. Since this dissertation is an ensemble of papers that were either previously published or that will be submitted for publication, Chapter 3 provides an overview that ties each of the chapters together as a part of the bigger picture. The subsequent chapters comprise the individual papers, which are followed by conclusions in Chapter 11.

Chapter 2

A Survey of Practical, Cost-Conscious Active Learning

A recent survey found that 80% of respondents that were collecting annotated data for NLP tasks did not use active learning (AL); 60% of them would not consider using AL in future projects [90]. One of the single biggest reasons (20.5% of participants) for not using AL was that participants were not convinced that AL would actually reduce the cost of annotation for their project. We deduce from this survey that a major impediment to the widespread adoption of active learning for annotating corpora is a disconnect between research and practice. After all, some theory has shown active learning to be capable of exponentially decreasing the labeling effort [29]. Indeed, active learning research has largely ignored a number of practical issues, not the least of which is the actual costs of annotation, as previously noted. In this chapter, we outline these aspects while presenting representative work for each aspect. The current survey is not intended to cover all aspects of active learning in general; for such a survey the reader is referred to Settles [79] and Olsson [62], the latter being focused on active learning for natural language processing.

2.1 Supervised Machine Learning

We begin with a brief overview of supervised machine learning, principally to define pertinent terms as a means of introducing active learning.

Learning is often defined as improvement in performance with experience. In traditional classroom settings, students are commonly evaluated using standardized tests at the beginning of the school year and again at the end. After a year's worth of lessons, homework, exams, etc., most students' scores improve between the pre-test and the post-test. Similarly, *machine learning* is the

ability of a computer to improve its performance on a task when given data (which can be seen as "experience"). Not unlike the classroom setting, a "task" consists of mapping an instance from an input (or problem) space $\mathscr{X}$ to an answer from the output (or solution) space $\mathscr{Y}$; the method of evaluation of the performance of a computer on a task is usually task-dependent; accuracy, precision, and recall are common metrics. Besides the Syriac example from the previous chapter, consider the following additional examples:

Spam Detection. Flag unwanted email messages as spam or not-spam; $\mathscr{X}$ is the set of all possible messages and $\mathscr{Y} = \{\text{spam}, \text{not spam}\}$.

Part-of-Speech (POS) Tagging. Not unlike grade-school grammar class, this problem consists of identifying the part-of-speech (i.e., noun, verb, preposition, etc.) of each word in a corpus. These annotations can aid the study of corpora, as well as improve performance of other tasks such as machine translation or parsing. In this task, $\mathscr{X}$ could be the set of all English words and $\mathscr{Y}$ is the set of all part-of-speech tags. More commonly, $\mathscr{X}$ is the infinite set of all English sentences and $\mathscr{Y}$ would be the infinite set of tag sequences.

Named Entity Recognition. This task consists of finding all the proper nouns in a sentence. Named entity recognition is used to identify gene names in scientific papers or the names of persons of interest in surveillance applications. $\mathscr{X}$ is the set of all possible sentences and $\mathscr{Y}$ is the set of all possible proper nouns.

Machine Translation. Translating a sentence in the source language to the target language; $\mathscr{X}$ is the infinite set of all possible sentences in the source language and $\mathscr{Y}$ is the infinite set of all possible sentences in the target language.

Parsing. Similar to the dreaded "sentence diagrams" from English class, this task consists of building a tree-like structure that groups words into phrases and phrases into higher-level phrases; $\mathscr{X}$ is the infinite set of all English sentences and $\mathscr{Y}$ is the infinite set of all trees.

Facial Recognition. Match faces to people for authentication or photo tagging; $\mathscr{X}$ is the set of all possible images and $\mathscr{Y}$ is the (finite) set of all recognizable people in a database.

When provided with instances and their solutions (e.g., email messages that have been flagged as spam or not-spam), computers can learn to map instances from the input space to the output space (e.g., can mark email messages as spam). This process is called *supervised learning*, which we define more formally in section 2.2.

As previously mentioned, a major obstacle to corpus creation is the cost associated with obtaining annotations from the output space for the corresponding samples from the input space, which usually requires an expert human annotator. To illustrate, consider the full Penn Treebank (i.e., not just the Wall Street Journal section). Obtaining the unannotated sentences in today's digital world would be trivial. In contrast, it reportedly took three years for four annotators with graduate-level experience in linguistics working 15 hours a week to provide tags for the 4.5 million words of text [56]. Contrast this effort to the amount of time it would take to scrape 4.5 million words from the web. The techniques that we discuss are particularly applicable when obtaining annotations is costly relative to the cost of obtaining instances.

The cost-reduction strategy that we focus on for the purposes of this dissertation is active learning (aka selective sampling and optimal experimental design). Let us return to the analogy of traditional pedagogy.[1] The examples that a teacher typically selects to illustrate concepts are rarely random but rather carefully selected to be particularly informative (without being overly time-consuming to explain); homework is rarely randomly selected for the same reason. This way, the teacher needs to present fewer examples on the board and can assign less homework while achieving similar performance from the students. This process is similar in spirit to active learning: rather than randomly selecting instances to train models, we can instead carefully select examples that are most beneficial and least costly. If we are successful, we will achieve similar levels of performance to randomly selecting instances, while reducing the cost of obtaining those annotations; or, equivalently, we will achieve higher levels of performance at the same cost. In fact, active learning has long been known to reduce the cost of training machine learning models, in theory: under certain not-necessarily reasonable conditions, it can be shown that active learning exponentially

[1]The analogy here, though informative, is technically deficient in the sense that the students do not select the examples.

reduces the number of labels required to train a model compared to standard supervised learning [20, 21, 29].

2.2 Formal Definitions

We now define traditional active learning formally, by first contrasting it with a formal definition of passive learning (standard supervised machine learning). For these definitions, we limit ourselves to the case were there is a single, infallible annotator (i.e., "oracle"), which assumption will be further addressed as a part of this dissertation. As a matter of notation, we use parentheses $()$ to indicate tuples and angled brackets $\langle\rangle$ to indicate a sequence. The symbol $\sim$ signifies that a variable is drawn from the specified distribution. For instance, $x \sim X$ means that x is drawn from the distribution of X. Lastly, we use $\oplus$ to represent the append operation for sequences.

2.2.1 Supervised Machine Learning

Formally, we define a machine learning *model* as a function $f : \mathscr{X} \mapsto \mathscr{Y}$. Let $\mathscr{F}_{\mathscr{X},\mathscr{Y}}$ be the set of all such functions for $\mathscr{X}$ and $\mathscr{Y}$. We further let X and Y be random variables with ranges $\mathscr{X}$ and $\mathscr{Y}$, respectively. Then a *training set* is a sample from the joint distribution of X and Y (this distribution is task and domain dependent). Let $\mathscr{D}_{\mathscr{X},\mathscr{Y}}$ be the set of all possible training sets for $\mathscr{X}$ and $\mathscr{Y}$. Thus, a *supervised learner* for input space $\mathscr{X}$ and output space $\mathscr{Y}$ is a function $g : \mathscr{D}_{\mathscr{X},\mathscr{Y}} \mapsto \mathscr{F}_{\mathscr{X},\mathscr{Y}}$; let $\mathscr{G}_{\mathscr{X},\mathscr{Y}}$ be the set of all such functions for $\mathscr{X}$ and $\mathscr{Y}$.

The example that will be used throughout this dissertation is the English POS tagging task. The training set that we use is the Wall Street Journal (WSJ) section of the Penn Treebank (PTB) [56], which contains approximately 1 million words of English with their corresponding part-of-speech tags across approximately 40,000 sentences. We let $\mathscr{X}$ be the set of all possible English sentences and $\mathscr{Y}$ be the set of all possible tag sequences. Therefore, the PTB is a member of $\mathscr{D}_{\mathscr{X},\mathscr{Y}}$ and contains about 40,000 samples from $p(X,Y)$. The model that we use is a Maximum Entropy Markov Model (MEMM) [71];[2] it takes as input an English sentence and outputs the tags

[2]The features we use are based on Toutanova and Manning [92].

for that sentence with between 96% and 97% accuracy. The learning algorithm we use minimizes the conditional likelihood of the training data to produce an MEMM.

The annotated (labeled) training set required for supervised machine learning is usually assumed to already exist. For the purposes of this dissertation, we are concerned with how such corpora are constructed; Algorithm 1 describes how corpora are assumed to be built. Although the algorithm is presented from the perspective of building a corpus, this is also the approach used to construct the traditional learning curves that are pervasive in the machine learning literature. This approach is called passive learning and it contrasts to active learning (described in the next subsection).

Algorithmically, obtaining annotations using passive learning proceeds as follows (see Algorithm 1). First, we obtain a sample $x \sim X$; so doing incurs a measurable *selection cost*. An annotation is obtained by drawing y from $Y|x$; the cost of this operation is called the *annotation cost*.[3] The provided annotation has some measurable benefit (e.g., increase in model accuracy on a held-out set) as indicated on line 9. We then append the pair $\langle x, y \rangle$ to the sequence L. This process continues until some stopping criterion is met (in practice, usually when a budget is exhausted). At the end of the process, we are left with a sequence of samples L drawn from the joint distribution $p(X,Y)$, $L \in \mathscr{D}_{\mathscr{X},\mathscr{Y}}$; c indicates the total cost of obtaining the annotated data, and b represents the benefit of the data. At this point, a supervised learning function $g \in \mathscr{G}_{\mathscr{X},\mathscr{Y}}$ can be employed to induce a model $f \in \mathscr{F}_{\mathscr{X},\mathscr{Y}}$ from L. It should be noted that this protocol is equivalent to standard supervised learning. This process is passive in the sense that the machine is not used to select which instances to annotate.

2.2.2 Active Learning

In contrast, rather than drawing instances randomly from $p(X)$ at each step, active learning seeks those instances that are ideally most informative and least costly. The idea is that by intelligently

[3] Equivalently, x and y may be drawn jointly from $p(X,Y)$, in which case SelectionCost and AnnotationCost are conflated. We further note that other schemes not directly applicable to this dissertation are available, e.g., *concept learning*, in which instances are drawn from $p(X|Y)$.

Output: An annotated corpus $L \in \mathscr{D}_{\mathscr{X},\mathscr{Y}}$ the total cost c, and benefit b, of obtaining L

$L = \langle\rangle$
$b = 0$
$c = 0$
while *not done* **do**
    $x \sim X$
    $c = c + \texttt{SelectionCost}(x)$
    $y \sim Y|x$
    $c = c + \texttt{AnnotationCost}(x, y|L)$
    $b = b + \texttt{Benefit}(x, y|L)$
    $L = L \oplus (x, y)$
end
return L, c, b

Algorithm 1: Passive Learning

selecting the "best" instances, the same amount of benefit can be derived from the solicited annotations for less total cost as compared to passive learning. For the purposes of this work, we assume that all instances are pre-annotated by the computer such that the task of the oracle is to correct any errors rather than to annotate from scratch. Formally, active learning can be defined as shown in Algorithm 2. First, a learning function $g(\cdot)$ is used to infer a model $f(\cdot)$ for pre-annotation using the annotated data obtained so far, $L \in \mathscr{D}_{\mathscr{X},\mathscr{Y}}$. So doing incurs a cost that we call the *training cost*. In addition, we need a scoring function that provides a measure of "best," balancing the cost and benefit associated with obtaining annotations for any given instance (higher is better). Formally, a scoring function is defined as $\phi : \mathscr{X} \times \mathscr{D}_{\mathscr{X},\mathscr{Y}} \mapsto \mathbb{R}$, where the second argument represents a sequence of already annotated items. For a given scoring function, we seek the next instance $x^* = \arg\max_{x \in \mathscr{X}} \phi(x|L)$. Unfortunately, this discrete optimization problem is NP-complete for many types of tasks and scoring functions. Instead, optimization seeks the instance x from a set of unique instances (usually samples from X), known as the pool[4] P, that maximizes $\phi(x|L)$. This approach is particularly appropriate for the annotation task since we are generally only interested in annotating the data which we have already collected. The time taken to find x^* is known as the *selection cost*. After pre-annotating x^* (and incurring a usually negligible *pre-annotation cost*) the

[4] *Pool-based active learning* is the only scenario we consider in this dissertation since it best matches the goal of obtaining annotations for unannotated data. Other scenarios are discussed by Carroll [13] and Settles [79].

Input: Learning function $g(\cdot)$ and initial "seed" data $L_0 \in \mathscr{D}_{\mathscr{X},\mathscr{Y}}$
Output: An annotated corpus $L \in \mathscr{D}_{\mathscr{X},\mathscr{Y}}$; the total cost c, and benefit b, of obtaining the additional data

1 $L = L_0$
2 $c = 0$
3 **while** *not done* **do**
4 $\quad f = g(L)$
5 $\quad c = c + \texttt{TrainingCost}(f)$
6 $\quad x^* = \arg\max_{x \in P} \phi(x|L)$
7 $\quad c = c + \texttt{SelectionCost}(x^*)$
8 $\quad \text{preannotation} = f(x^*)$
9 $\quad c = c + \texttt{PreAnnotationCost}(x^*)$
10 $\quad y \sim Y|x^*$
11 $\quad c = c + \texttt{AnnotationCost}(x^*, y, \text{preannotation} \,|L)$
12 $\quad b = b + \texttt{Benefit}(x^*, y|L)$
13 $\quad L = L \oplus (x^*, y)$
14 **end**
15 **return** $L, cost$

Algorithm 2: Active Learning with Pre-Annotation

oracle is then consulted to provide an annotation $y \sim Y|x^*$ and the pair (x^*, y) is added to L. This new information allows the scoring function to update its belief about which instances are most beneficial; it also presumably improves the accuracy of the pre-annotation model. The process is repeated until some criterion is met, frequently, when the budget is exhausted.

A few aspects of Algorithm 2 depart from traditional active learning. While we have chosen to show all of the costs associated with annotation (i.e., lines 5, 7, 9, and 11), these costs have traditionally been ignored, although there has been recent awareness of the annotation cost (e.g., Donmez and Carbonell [24], Settles et al. [81], Tomanek and Hahn [89]). Those already familiar with active learning will also notice that in our formulation: (1) P is a set rather than a multiset and (2) instances are not removed from P after being added to L. If the set of samples includes duplicates (a possibility for highly probable x), $\phi(x|L)$ need only be evaluated once per unique x. Furthermore, although learning the distribution $Y|x$ potentially requires that we obtain multiple annotations for each x, the number of annotations desired for each unique x does not necessarily depend on the frequency with which the x occurs in the sample, but should instead be determined by the scoring function, i.e., should be related to the benefit of obtaining another

annotation relative to the cost. However, some scoring functions may be deficient in this regard and the traditional approach may in fact be beneficial in these cases.

2.3 Transductive vs. Inductive Active Learning

Perhaps the most fundamental issue concerning active learning is the end to which it is employed. There are at least two non-mutually-exclusive objectives of active learning. The traditional use of active learning has been as a learning protocol (as implied by the name), in which case, the objective is to obtain a model of maximum quality on a fixed budget (or to otherwise balance the quality of the model with the cost of achieving that quality). We call this strategy *inductive active learning* after Carroll [13]; some early examples include Angluin [3], Cohn et al. [16], Dagan and Engelson [19], Lewis and Gale [51].

Since active learning obtains annotations for previously unannotated instances, an additional output of active learning is a set of annotated instances (c.f. Algorithm 2). Thus, another type of active learning is focused less on obtaining a model and more on obtaining as many high quality annotations as makes financial sense [73, 91]. In this sense, the model (and active learning) are means to an end, rather than the end itself. The measure of the usefulness of the final annotated corpus is project-dependent, but is usually related to the accuracy of the annotations and the quantity obtained; the quality of a model trained on the annotations is often used as a surrogate to measure the quality of the corpus.

Importantly, we can obtain many more annotations if we use a model trained on the annotated portion of the data to automatically annotate the rest of the corpus. The goal, of course, is to achieve the highest possible quality of annotations for the entire corpus, both on the portion annotated by human experts as well as the portion annotated by machine. If we choose instances carefully, we can let the machine tag the instances with which it would agree with the humans, and let the humans annotate the instances with which the machine needs help. We call this strategy *transductive active*

learning after Carroll [13].[5] Palmer et al. [64] and Baldridge and Palmer [7] explicitly mention this strategy as their goal in their language documentation efforts for Uspanteko. Transductive active learning can be achieved, though not necessarily optimally, by employing inductive active learning to achieve a high quality model and using this model to annotate the rest of the corpus. We will take this approach throughout most of this dissertation, given that, not surprisingly, Palmer et al. [64] found a correlation between model accuracy and corpus accuracy.

2.4 Evaluation

Each instance that is annotated results in some benefit and incurs some cost, as depicted in Algorithms 1 and 2. The exact determination of cost and benefit are typically project-dependent and sometimes difficult to measure. Nevertheless, the performance of an AL algorithm is ultimately determined by both the true cost and benefit, and hence, any attempt to characterize the performance of the algorithm should measure both aspects as closely to reality as possible.

Once a suitable metric is determined for both cost and benefit, there is still a need to combine them in a way that properly characterizes the cost/benefit trade-off. When cost and benefit are in the same units, the *net benefit* (i.e., benefit less cost) can be used. However, benefit and cost are rarely in the same units; instead, all previous work of which we are aware evaluates active learning methods using cost/benefit curves, or derivations thereof. Cost/benefit curves are generalizations of learning curves that parametrically plot the benefit achieved vs. cost incurred as a function of each annotation obtained. Cost/benefit curves are useful in showing the performance of scoring algorithms across an entire range of costs, rather than showing just the final values. This range is particularly useful when extrapolating information from the curve in the face of uncertainty about how much money will be spent on a project that uses the same techniques.

The remainder of this section discusses issues related to the measurement of benefit and cost, including a discussion on dealing with realistic cost in simulation.

[5] Transductive active learning does not necessarily imply that a transductive model is used to annotate instances. It simply refers to the goal of directly annotating the unseen data.

2.4.1 Benefit

Clearly, the benefit of active learning is intrinsically tied to its purpose. The benefit of inductive active learning is a model of the highest possible quality, and therefore the performance of the resultant model on held-out data is a useful measure. The benefit of transductive active learning could appropriately be measured by the accuracy of the labels on the entire data set (i.e., the portion labeled by machine as well as that labeled by human annotators). However, this metric ignores other difficult to measure benefits, such as the ability of other learning algorithms to successfully learn from the same data [6], and the utility of the resultant corpus to other disciplines.

2.4.2 Cost

Similarly, the cost of an AL project depends on many variable and fixed costs [61]. Recall that there are four costs associated with active learning: training costs, selection costs, pre-annotation costs, and annotation costs (see Algorithm 2). To the best of our knowledge, all previous work assumes that all but annotation costs are zero for evaluation purposes. This assumption is problematic since it encourages algorithms of increasing complexity without consideration for the delays they inflict. In practice, the other costs (training, selection, and pre-annotation) are either direct costs (i.e., they cause an annotator to wait for the computer in annotator-initiated active learning), or constitute opportunity costs (e.g., prolong the annotation process to the point that we obtain fewer total annotations). Clearly, these should be accounted for in some form. In fact, Tomanek et al. [91] explicitly list "fast selection time cycles" (i.e., low training, selection, and pre-annotation costs) as requisite for practical active learning. Some work, most notably Roy and McCallum [75], Settles and Craven [80], and Anderson and Moore [2], have discussed the scoring and training costs of their approaches, but chose not to include them in their actual evaluation.

Often times, how the annotators will be paid is one of the more significant factors of cost, in practice. The annotation cost depends both on which instance is being annotated as well as who is to annotate it [5], the nature of the annotation user interface [14, 32], and fatigue and learning curve effects [26]. The usual assumption for evaluating AL is that each instance is equally costly, which is

unrealistic, particularly for tasks involving labeling sequences or other structured data. For instance, all else being equal, the cost to POS-tag a one word sentence is vastly different than that of an 100-word sentence. Not surprisingly, Settles et al. [81] empirically showed that on some real-world annotation tasks, the time to annotate instances was in fact approximately constant, whereas for other tasks it was not. For the purposes of this dissertation, we are primarily interested in the latter. Becker et al. [8] and Haertel et al. [34] show how the relative performance of different algorithms is determined in part by how accurately cost is measured. Thus, evaluating results on the basis of constant cost can be very misleading when the actual cost is variable per instance.

2.4.3 Cost in Simulation

Ideally, active learning would be evaluated using actual timings from real user studies, as in Baldridge and Osborne [6], Hachey et al. [33], Ngai and Yarowsky [61], Palmer et al. [64]. However, the number of experiments needed to compare competing approaches would normally make this type of evaluation cost-prohibitive [5]. One alternative is to collect timing on data collected one way (e.g. sequentially, randomly, or using a single active learning approach), and use it as a gold standard when evaluating other approaches; examples include Settles et al. [81], Tomanek and Hahn [89]. This technique implicitly assumes that the annotation time is independent of order (since different active learning algorithms will present instances in different orders). However, this assumption is not always acceptable. Suppose we employ pre-annotations in conjunction with active learning and update the model that produces the pre-annotations with the feedback from active learning (a form of correction propagation). In general, as the model performance improves over time, instances will presumably cost less since fewer changes need to be made (a fact confirmed by Felt et al. [28]). Thus, all else being equal, choosing to annotate an instance at the beginning of active learning will cost more than deferring that instance until the latter stages. Hence, it is not possible to reuse the timestamps. Arguably, the SeSAL approach of Tomanek and Hahn [88] is also subject to time-dependencies.

Another alternative involves simulating annotation time using a model of cost [5]. This approach has the advantage of being able to model order dependencies. The effectiveness of these simulations clearly depends on the quality of the cost model. To our knowledge, no previous work has taken this approach.

2.5 Scoring Functions

The success of an AL algorithm (as depicted on a cost/benefit curve) depends largely on its scoring function (see Algorithm 2). For this reason, the majority of active learning research has been focused on the development of new scoring functions for different problems, models, objectives, etc. Needless to say, the better a scoring function's outcome represents the actual benefit and cost of an instance, the better it will perform. In this section, we begin with a discussion of frameworks for scoring functions: one based on the *Expected Value of Sampling Information* (EVSI) [69] and the other based on Return-on-Investment (ROI); we also briefly mention other frameworks. Since these frameworks still depend on separately defined benefit and cost functions, we follow the exposition of the scoring frameworks with a discussion of the most commonly used benefit and cost functions.

The following definitions are pertinent to the remainder of this section. First, we assume a benefit function[6] $B : \mathscr{D}_{\mathscr{X},\mathscr{Y}} \times \mathscr{D}_{\mathscr{X},\mathscr{Y}} \mapsto \mathbb{R}$ that measures our overall satisfaction of a sequence L' of $\langle$instance, annotation$\rangle$ pairs, given some previously annotated data L; we denote such a function $B(L'|L)$. Likewise, we need a cost function $C : \mathscr{D}_{\mathscr{X},\mathscr{Y}} \times \mathscr{D}_{\mathscr{X},\mathscr{Y}} \mapsto \mathbb{R}^+$ that measures the cost of obtaining a set of annotations L', given a set of already obtained annotations L; we denote this function $C(L'|L)$.

2.5.1 EVSI Scoring Functions

In this section we present scoring functions based on the Expected Value of Information (EVSI) [69]. The *value of information*, i.e., the usefulness of obtaining additional annotated data L' is the

[6]The machine learning literature often speaks pessimistically in terms of loss functions and risk. We follow the more optimistic terms from decision theory and use the terms benefit functions and utility. Loss is an arbitrary constant minus the benefit.

difference between the benefit with the additional data and the benefit without it, i.e.,

$$V(L'|L) = B(L'|L) - B(\cdot|L).$$

Then the *net value* (of sampling information) of a sequence of annotations L' given previous annotations L, subject to budget D is:

$$N(L'|L) = \begin{cases} V(L'|L) - C(L'|L) & \text{if } C(L'|L) \leq D \\ -\infty & \text{otherwise} \end{cases}.$$

Since the value of any given decision depends on the effect the decision has on future decisions, the net value of any given instance is the maximum net value the decision can produce in the future (in expectation). Formally, we define the following recursive function, which represents the total expected net value of an instance; the parameter L' is used during recursion to collect the future sequence of annotations:

$$G(x, L'|L) = \max \begin{pmatrix} \mathbb{E}_{Y|x,L\oplus L'}\left[N(L'|L)\right], \\ \mathbb{E}_{Y|x,L\oplus L'}\left[\max_{x'} G(x', L' \oplus (x,y)|L)\right] \end{pmatrix}.$$

The first argument of the outermost max is the base case, representing the scenario in which the expected cost of further annotations outweighs the expected value of this information, i.e., the net value would be lower if we obtained more annotations. In this case, active learning would be expected to stop with the current sequence L' of annotations and would return the net value of this sequence. On the other hand, should more annotations result in a higher net value, the decision process continues, that is, finds the next instance that is expected to (eventually) maximize the expected net value; hence the second argument of the outermost max.

The fully recursive EVSI scoring function scores each instance using the maximum expected net value of that instance via the recursive function $G(\cdot)$:

$$\phi_{\text{REC-EVSI}}(x|L) = G(x, \cdot\,|L), \tag{2.1}$$

so long as the corresponding max is positive. Otherwise, it would be irrational to proceed with active learning. This condition is, in fact, the optimal *stopping criterion*.[7]

Unfortunately, this approach is intractable. In fact, the recursion is not necessarily guaranteed to terminate. Tractability can be obtained by limiting the depth of recursion to some constant r [13, 46], though optimality would only be guaranteed for the last r decisions. The greediest case involves no recursion, resulting in:

$$\phi_{\text{GREEDY-EVSI}}(x|L) = \mathbb{E}_{Y|x}\left[N((x,y)|L)\right]. \tag{2.2}$$

2.5.2 Return-on-Investment Scoring Function

The goal of every annotation project should be to obtain those annotations that maximize the net value of the annotated data. This optimality is guaranteed by equation 2.1, but *not* by the greedy approach in equation 2.2. Thus, other greedy approaches outside of the framework may perform equally well or better. We now turn to the financial world for an alternative approach. Similar to the goal of active learning, an investor's goal is to make the highest net profit possible during his lifetime. However, it is impossible to know all of the potential investment options that will be available in the course of his lifetime. Thus, investing—like the approach to active learning from equation 2.2—is a greedy operation (in both senses) in which the investor makes the best choices now in hopes that the gains from these decisions can be used to make unknown future decisions that will ultimately maximize net profit. One approach that investors use to attempt to greedily maximize their net profit (analogous to net value) is to make choices on the basis of return-on-investment,

[7] In the context of maximization, computing $B(\cdot|L)$ is unnecessary other than to evaluate the stopping criterion.

defined as the net profit over the cost. Using our established notation,

$$\begin{aligned}\phi_{\text{ROI}}(x|L) &= \frac{\mathbb{E}_{Y|x}[N((x,y)|L)]}{\mathbb{E}_{Y|x}[C((x,y)|L)]} \\ &= \frac{\mathbb{E}_{Y|x}[V((x,y)|L) - C((x,y)|L)]}{\mathbb{E}_{Y|x}[C((x,y)|L)]} \\ &= \frac{\mathbb{E}_{Y|x}[V((x,y)|L)]}{\mathbb{E}_{Y|x}[C((x,y)|L)]} - 1 \end{aligned} \tag{2.3}$$

so long as the return-on-investment is positive (no rational investor would invest in a loss; this is the same stopping criterion associated with equation 2.2).

All else being equal, if two investment options are equally costly, ROI prefers the investment that has the highest net profit. If two investment options would yield the same net profit, ROI gives preference to the one that costs less; we can use the extra money on other investments to similarly maximize our profits. By greedily maximizing the cost/benefit (profit) ratio, early returns can be reinvested more quickly, hopefully resulting in higher net profit for the same cost as alternative investing strategies. For essentially these same reasons, return-on-investment is a viable alternative to the greedy net EVSI framework (equation 2.2) in active learning. Another major advantage of return-on-investment is that the net value need not be explicitly computed and therefore cost and benefit need not be expressed in the same units.[8]

When each instance is presumed to be equally costly, then the two scoring functions ($\phi_{\text{GREEDY-EVSI}}$ and ϕ_{ROI}) are equivalent. However, as discussed in Section 2.4.2, structured learning problems have necessitated a means to deal with variable cost, and many of the early work in active learning for structured problems can be seen as instances of the ROI-framework. For instance, Engelson and Dagan [27] normalized their measure of uncertainty for POS-tagging by sentence length. If the cost to tag a sentence is a constant scalar multiple of the number of sentences, then length normalization is an instance of ROI. Other notable sequence labeling problems that do the same include Becker et al. [8], Hachey et al. [33], Settles and Craven [80], Tomanek et al. [91]. Tang et al. [86] and Hwa [42] use length normalization for parse trees, and Kapoor et al. [46] assumes that

[8] However, they must be converted to the same units in order to correctly compute the stopping criterion.

cost is a scalar multiple of the length of voicemails. However, Settles et al. [81] present evidence that the relationship between cost and length of instances is not simply linear.

The general framework for return-on-investment was independently proposed by Donmez and Carbonell [24] and Settles et al. [81]. Both essentially proposed using any of the heuristic benefit functions above. While the former showed positive experimental results using return-on-investment, the latter showed mixed results. Tomanek and Hahn [89] show that the benefit function must be carefully chosen. While the unmodified benefit function performed poorly, the modified function outperformed all other cost-conscious algorithms they tried (see next section) and was recommended by those authors.

2.5.3 Other Cost-Sensitive Scoring Functions

In addition to employing return-on-investment, Tomanek and Hahn [89] propose two novel approaches: constraining the sampling to a maximum cost per example and a linear combination of the utility rank and cost rank. Their results slightly favor the return-on-investment method when an appropriate utility function is used (in their case, they exponentiate a confidence score). The other methods require tuning a corpus-specific parameter, which may not be possible during active learning in practice.

2.6 Cost and Benefit Functions

As the cost-conscious scoring functions depend on cost and benefit functions, this section examines these functions more in-depth.

2.6.1 Decision Theory Benefit Functions

A decision-theoretic purist would insist on very specific benefit functions for inductive and transductive active learning. First, we define a utility function, $U : \mathscr{Y} \times \mathscr{Y} \mapsto \mathbb{R}$, which gives a measure of the satisfaction received when choosing annotation y (first argument) when the true label is $\hat{y}$ (second argument) for instance x. Since the true annotation is unknown, we consider its value in

expectation; the optimal annotation for instance x maximizes the expected utility. The corresponding decision function is:

$$\delta(x) = \arg\max_y \mathbb{E}_{\hat{Y}|x,L}[U(y,\hat{y})].$$

Induction seeks to maximize this quantity, on average, across all possible instances (i.e., maximize generalization). Formally, the benefit function is:

$$B_{\text{IND}}(L'|L) = \mathbb{E}_{\tilde{X}}\left[\max_y \mathbb{E}_{\tilde{Y}|\tilde{x},L\oplus L'}[U(y,\tilde{y})]\right]. \tag{2.4}$$

Roy and McCallum [75] advocate this approach, although they omit the cost component in their framework. Liang et al. [52] also essentially use this same approach in their measurement framework; however, in their actual experiments, they assume unit cost.

In practice, the actual distribution of X is not known. However, stochastic integration can be used to compute the expectation in equation 2.4 using a sequence of N samples, $x_1,\ldots,x_n \sim X$. This leads to the benefit function for transductive learning:

$$B_{\text{TRANS}}(L'|L) \approx \frac{1}{N}\sum_i^N \max_y \mathbb{E}_{Y_i|x_i,L\oplus L'}[U(y,y_i)]. \tag{2.5}$$

Roy and McCallum [75] employ this approach in their experiments (although they only compute the benefit over held-out, *unlabeled* examples).

There is a prevalent variant of the above benefit functions that, rather than considering maximum expected utility, instead considers the average expected utility:

$$B_{\text{AVG}}(L'|L) \approx \frac{1}{N}\sum_i^N \left[\mathbb{E}_{Y_i|x_i,L\oplus L'}\left[\mathbb{E}_{\tilde{Y}_i|x_i,L\oplus L'}[U(y_i,\tilde{y}_i)]\right]\right];$$

Anderson and Moore [2], Margineantu [58], etc. use this variant. If we assume that $p(Y_i = y_i|x_i,L\oplus L') = 1$ for all instances x_i that an oracle has annotated with y_i, then the outer expectation disappears for these instances; Kapoor et al. [46] use precisely this approach.

2.6.2 Heuristic Benefit Functions

The main drawback to the previous benefit functions is that they can be relatively expensive to compute, especially in the context of structured learning where the expectations are considerably more expensive to compute than in simple classification tasks. However, many heuristic benefit functions exist which typically avoid the expectations altogether. We claim that some of these heuristics can be derived from equation 2.2 by applying additional simplifying assumptions; others can probably be seen as nothing more than heuristics. We discuss several of the heuristics at a high level below; a more in-depth discussion of most of the approaches, including many representative citations, is provided by Settles [79].

One of the most influential heuristics is known as *uncertainty sampling* [51], or as we like to call it, *query-by-uncertainty* (QBU). Intuitively, if a model is already certain about an annotation, requesting an annotation is not likely to be helpful; conversely, the less sure the model is about what the annotation could be for an instance, the more likely it is that human input could help the model. There are many definitions of uncertainty, including least confidence [18], margin [76], entropy [42], etc. Settles [79] overviews other useful measures of uncertainty for sequence labeling. Formally, entropy-based uncertainty is defined as:

$$B_{\text{ENTROPY}}(x,y|L) = \mathbb{E}_{Y'|x,L}[-\log p(y'|x,L)]$$

Note the lack of dependency on y, which in turn eliminates the expectation from equation 2.2. This function inherently prefers instances that are probably wrong, which is similar to a utility function with higher weights when the annotations are not equal than when they are, e.g., $U(y,\hat{y}) = 1(y \neq \hat{y})$.

Another important benefit function is known as query-by-committee (QBC) [19, 82]. The idea is that if we have multiple diverse models, known as the committee, then instances for which the committee disagrees most about the predicted annotation could benefit most from human annotations. There are several approaches to creating a committee, including: training different types of models on the same data, sampling models of the same class from the posterior distribution

of model parameters given the annotated data [27], building models from the same class using bootstrap samples from the training data [1], and using models with different features extracted from the data [45]. Likewise, there are many approaches to quantifying uncertainty. Since QBC methods can be seen as uncertainty-based methods, where the distribution over which uncertainty is computed is the distribution of predictions from the committee of votes. Thus, the same uncertainty measures used for QBU may also be used for QBC. However, due to the relatively small size of the committees (typically 3-5 members), structured learning techniques approximate the uncertainty measures by considering the votes of each individual prediction in the structure. For example, Dagan and Engelson [19] use the sum of the entropy of the per-token vote distribution; others are detailed by Settles [79].

Many of these methods, especially QBU and QBC, favor instances that are outliers or noise, and hence are not always particularly useful. This bias can be corrected by weighting the original benefit by the density of the instance, thereby favoring instances that are common and therefore presumably more useful (at the risk of sampling instances that risk diversity). To be clear, the benefit functions in section 2.6.1 already account for density in the expectations; density-weighting is only needed to correct deficiencies in the heuristics. Since heuristics are often necessary in practice, Settles and Craven [80] and Settles [78] present a general framework for density-weighted methods; others who have used the method include Roy and McCallum [75] and McCallum and Nigam [59]. The general consensus is that, when properly tuned, the density-weighted methods outperform their non-weighted counterparts. However, density-weighted methods can be expensive, rely on an arbitrarily selected distance function, and have a parameter that must be tuned, thereby putting into question their practicality.

2.6.3 Cost

Early work can be seen as having implicitly assumed that each instance was equally costly, in which case, the constant cost can be dropped in the context of maximization.[9] Thus, instance selection

[9]Consequently, no conversion is necessary between units of benefit and cost, except for stopping criteria.

reduces to maximization of (expected) benefit. For many classification problems, this assumption may be acceptable, but structured learning problems pose new problems. Even previous work based on decision theory either omits the term in the theory (e.g. Roy and McCallum [75]) or assumes unit cost for the actual experiments, e.g., that annotating each sentence is equally costly [52], annotating each token is equally costly [27, 91], annotating each word type is equally costly [52], or that listening to voicemails takes the same amount of time [46]. Donmez and Carbonell [24] do use variable cost in one experiment, but it is a linear function of the proximity of the point to the decision boundary, rather than a real world cost.[10]

A better approach is to explicitly model cost. In some cases, it is possible to know costs ahead of time, such as the cost-of-goods needed to perform an experiment [47], or the hourly rate of different workers [24]. However, in many practical scenarios–especially with structured learning–the cost will not be known beforehand and must be learned during active learning (as with all other parameters relevant to active learning). Clearly, more work needs to be done to employ and model realistic costs in research in order to provide more credible results for practitioners.

2.7 Characteristics of Real Annotators

Nearly all previous work in active learning has assumed a single, infallible oracle with perfectly predictable time. In fact, Algorithm 2 implicitly makes these assumptions. Interesting work has been done to study the actual characteristics of real annotators. Although outside of the context of active learning, one comprehensive study of characteristics of annotators across multiple tasks was done by Settles et al. [81]. They found that annotation times vary substantially from one annotator to another; that there tends to be a learning curve effect, albeit very brief; that some annotators get distracted thus extending annotation time; and, perhaps most importantly, that annotation times can be modeled reasonably well. In a separate study, Hachey et al. [33] found that, despite providing a sizeable overall reduction in cost, active learning nevertheless tends to select instances that result in lower inter-annotator agreement and higher annotation times. This tendency was confirmed by

[10] Although such points will indeed be more costly, the costs are almost certainly non-linear.

Palmer et al. [64] on a language documentation task. They additionally found that an expert on the language benefited from active learning, while the non-expert did better with passive learning (and in fact, appears to have higher accuracy than the expert on randomly selected instances). However, the expert was typically faster. Another study by Arora et al. [5] confirmed that the proficiency of the annotator in the language of the task at hand affected their annotation speed.

These studies emphasize at least three important aspects of annotators in a practical active learning environment. First, most systems will employ multiple annotators. Second, all annotators provide noisy annotations–some more so than others. Third, annotators will not necessarily be equally costly (especially when accounting for opportunity costs and differences in accuracy). In addition to these aspects, Donmez and Carbonell [24] suggest that annotators may not always be willing to provide annotations for every instance with which they are presented.

2.8 Learner-Initiated vs. Annotator-Initiated Active Learning

There are two paradigms in which active learning can be used. Either the computer can ask an annotator for an annotation, which we call *learner-initiated active learning*, or the annotator can request an item from the computer, which we term *annotator-initiated active learning*. In the former, the computer does all of the processing necessary to choose the best item and the best person to annotate the item and then sends a request to the selected annotator, e.g., sending an email or text message requesting that the annotator log in to an annotation system to perform the single annotation (although requests could certainly be batched). If choosing the next item and next annotator is computationally expensive, long periods of time could pass before another annotation is requested. At the very least, the overall annotation effort would be prolonged. Note, also, that between the time the annotator is notified and the time that the annotator finishes annotating, the computer may not be able to do much. The result is tremendous opportunity costs of the type that are difficult to both measure and predict; hence they are ignored in research. Ignoring these costs is equivalent to assuming that annotators are instantly available.

Annotator-initiated active learning, on the other hand, more closely resembles actual annotation in practice. Typically, an annotator will log in to the system at her convenience and in accordance with her contract. After logging in, the system will present her with a continuous stream of instances to annotate until she logs out, or the system decides to stop providing instances. The difference between the two paradigms is analogous to push and pull chains or the observer and iterator design patterns. Finally, note that the distinction between the two paradigms disappears when there is a single annotator who is constantly and instantly available. Although this assumption is prevalent in active learning research, it is rarely true in practice. Unfortunately, all previous work is based on the learner-initiated paradigm.

Chapter 3

Roadmap

The preceding chapter establishes the need for active learning techniques in more realistic scenarios, and such techniques are the focus of this dissertation. Using the newly introduced terminology from the preceding chapter, we can precisely formulate our thesis statement as follows: *Corpus annotation via cost-conscious, transductive active learning can reduce the total cost of annotating instances of varying benefit and cost in an annotator-initiated environment. Furthermore, under these same conditions, cost reductions are also likely with multiple, fallible annotators of differing levels of accuracy and cost; we present methods for doing so.*

Each of the remaining chapters in this dissertation investigates one or more aspects of this thesis statement. Since the majority of the chapters in this dissertation were published independently in various academic venues,[1] this chapter relates these individual papers to the thesis statement using the framework established in the previous chapter. We do so by framing each chapter in its historic context.

Our initial foray continues a line of work targeting the use of active learning as a method for annotating corpora (as opposed to inductive active learning; e.g., Baldridge and Osborne [6], Hwa [42], Osborne and Baldridge [63], Tomanek et al. [91]). Full transductive active learning, however, takes things a step further: a machine could automatically annotate the remaining data to further augment the corpus. This use of active learning is the stated objective of Chapter 4, published in the *Proceedings of the Linguistic Annotation Workshop*, 2007 [73] (although we do not use the term "transductive" in that chapter), and was novel at the time. In that chapter, we mostly rely on the

[1] Each published paper is included verbatim, with the exception of some minor formatting changes and occasional spelling corrections. Errata and addenda have been added as appropriate to individual chapters.

traditional active learning techniques of the time to achieve the new goal, i.e., we use inductive active learning to learn a model that could be used to label the unannotated portion of the corpus, though we do not actually report the results of such automatic annotation. We continue this approach in all other chapters of this dissertation except for Chapter 10.

In our aforementioned work, we followed the precedent of measuring performance based on the number of instances that were selected for annotation. However, this investigation helped us realize that, in reality, "instances [are] of varying . . . cost," particularly in the case of structured learning problems such as our part-of-speech tagging task. Demonstrating that our techniques "reduce the total cost" of annotation requires the ability to account for this variable cost. To this end we conducted a user study to investigate the components of cost on our target task, part-of-speech tagging, which was published in the *Proceedings of the International Conference on Language Resources and Evaluation*, 2008 [74]. This article is found in Chapter 5. From the data obtained from this study, we built a model of the amount of time required to tag sentences. This model allows us to measure the cost of different algorithms while simulating active learning. Without the model, the sheer number of would-be human hours required to perform the experiments in subsequent work would have been prohibitively costly. But equally importantly, the model derived from real user study data enables us to demonstrate much more realistically that the total cost of annotation has been reduced in the experiments presented in the remainder of the dissertation.

Although we prefer to measure the cost of annotation in terms of time ("time is money"), cost is project-dependent. For instance, a project may instead choose to pay annotators for each sentence they annotate or each word they annotate. We were curious how various existing active learning algorithms performed under various costs—including, of course, time, as measured by our recently developed model of cost. Our empirical investigation is presented in Chapter 6, published in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers* (2008) [34]. Not surprisingly, we found that the relative performance of algorithms depends on the manner in which cost is measured; in the case of

measuring cost by the number of words that an annotator changes, no algorithm outperformed the random baseline.

The fact that performance of the algorithms depends on how cost is measured makes active learning a risky proposition: all else being equal, if a project's true costs differ from those measured in any experiments, the results may be invalid and in fact worse than random.[2] Clearly, then, algorithms that perform equally well regardless of how cost is measured are desirable.

This unfortunate reality brings the notion of "cost-conscious" algorithms to the forefront. Algorithms that explicitly model the cost being measured are considered cost-conscious. Chapter 6 evaluates an existing algorithm for active learning on sequence tasks which simply divides the result of a "standard" selection function by the length of the sequence. Since the length of the sequence is highly correlated with the time required to annotate the sequence [34], it performs well. More importantly, we recognized that this approach of normalizing an estimate of benefit (the output of an existing active learning scoring function), by an estimate of cost (sequence length), was an instance of a more general techniques for cost-conscious active learning: return-on-investment (ROI). Chapter 7, published in the *Proceedings of the NIPS Workshop on Cost-Sensitive Learning* (2013) [36], and Chapter 8, an unsubmitted manuscript, present theoretical justifications and empirical results for return-on-investment-based active learning. Note that ROI requires a cost and a benefit estimator, which, given an instance, can estimate the cost and benefit that the instance will provide. Thus, ROI adeptly handles the case of "instances of varying cost and benefit."

While working on ROI, we noticed that all prior research in the field of active learning is based on learner-initiation: the computer selects one or more instances and then requests an annotation from the human annotator. In this scenario, the computer must wait both for the annotator to begin working and for the annotator to finish the assigned work. Chapter 9 turns previous research on its head by examining the scenario in which annotators request instances from the computer; although we never use the term "annotator-initiated," the chapter makes clear that this scenario is the motivation for the study. While learner-initiated techniques are trivially converted to an

[2] In reality, without further information the true results are as likely to be better than the experiments show as they are to be worse.

annotator-initiated approach, the paradigm shift brings to light a new problem: annotators must wait for the computation to be performed. This "wait cost" may involve direct costs and/or opportunity costs. Chapter 9, published in the *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, 2010 [35], is the first work of which we are aware to account for this cost. The chapter focuses on a solution to eliminating wait cost, which involves, among other things, performing computation while the annotator is at work. This "parallelization" of effort is the focus of the title of the chapter, "Parallel Active Learning" (we also mention how the computation itself can be parallelized). We call this framework the "no-wait" framework.

The last remaining issue within the scope of this dissertation is that of "multiple, fallible annotators of differing levels of accuracy and cost." Chapter 8 extends the "no-wait" framework to allow for multiple annotators, and evaluates results using 20 identical oracles. Chapter 10 (unsubmitted manuscript) further extends that work for use with multiple, fallible annotators of differing levels of accuracy and cost. We present a Bayesian model that is designed to simultaneously infer ground truth annotations from noisy annotations, infer each individual annotator's accuracy, and predict its own accuracy on unseen data (without the use of a held-out set). We have also extended ROI-based active learning to the multiple annotator environment. The detailed design of these approaches is presented in Chapter 10, but experimentation remains for future work.

Most of our work has been done using maximum entropy Markov models—a class of sequence models. Using sequence models in active learning often involves the computation of expectations. If done naïvely, computing expected values would require time exponential in the length of the sequence. Instead, approximations are often used. In Appendix A, we present an approach to efficiently computing expectations exactly in sequence models using dynamic programming.

Taken as a whole, these chapters introduce cost-conscious active learning (i.e., the class of algorithms we call ROI). We show that these techniques can reduce the cost of annotation in realistic environments.

Chapter 4

Active Learning for Part-of-Speech Tagging: Accelerating Corpus Annotation[†]

Abstract

In the construction of a part-of-speech annotated corpus, we are constrained by a fixed budget. A fully annotated corpus is required, but we can afford to label only a subset. We train a Maximum Entropy Markov Model tagger from a labeled subset and automatically tag the remainder. This paper addresses the question of where to focus our manual tagging efforts in order to deliver an annotation of highest quality. In this context, we find that active learning is always helpful. We focus on Query by Uncertainty (QBU) and Query by Committee (QBC) and report on experiments with several baselines and new variations of QBC and QBU, inspired by weaknesses particular to their use in this application. Experiments on English prose and poetry test these approaches and evaluate their robustness. The results allow us to make recommendations for both types of text and raise questions that will lead to further inquiry.

4.1 Introduction

We are operating (as many do) on a fixed budget and need annotated text in the context of a larger project. We need a fully annotated corpus but can afford to annotate only a subset. To address our budgetary constraint, we train a model from a manually annotated subset of the corpus and automatically annotate the remainder. At issue is where to focus manual annotation efforts in order

[†]Eric Ringger, Marc Carmen, Robbie Haertel, Kevin Seppi, Deryle Lonsdale, Peter McClanahan, James Carroll, and Noel Ellison. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proceedings of the international Conference on Language Resources and Evaluations*, 2008.

to produce a complete annotation of highest possible quality. A follow-up question is whether these techniques work equally well on different types of text.

In particular, we require part-of-speech (POS) annotations. In this paper we employ a state-of-the-art tagger on both prose and poetry, and we examine multiple known and novel active learning (or sampling) techniques in order to determine which work best in this context. We show that the results obtained by a state-of-the-art tagger trained on a small portion of the data selected through active learning can approach the accuracy attained by human annotators and are on par with results from exhaustively trained automatic taggers.

In a study based on English language data presented here, we identify several active learning techniques and make several recommendations that we hope will be portable for application to other text types and to other languages. In section 4.2 we briefly review the state of the art approach to POS tagging. In section 4.3, we survey the approaches to active learning employed in this study, including variations on commonly known techniques. Section 4.4 introduces the experimental regime and presents results and their implications. Section 4.5 draws conclusions and identifies opportunities for follow-up research.

4.2 Part of Speech Tagging

Labeling natural language data with part-of-speech tags can be a complicated task, requiring much effort and expense, even for trained annotators. Several efforts, notably the Alembic workbench [23] and similar tools, have provided interfaces to aid annotators in the process.

Automatic POS tagging of text using probabilistic models is mostly a solved problem but requires supervised learning from substantial amounts of training data. Previous work demonstrates the suitability of Hidden Markov Models for POS tagging [9, 49]. More recent work has achieved state-of-the-art results with Maximum entropy conditional Markov models (MaxEnt CMMs, or MEMMs for short) [71, 92, 93]. Part of the success of MEMMs can be attributed to the absence of independence assumptions among predictive features and the resulting ease of feature engineering.

To the best of our knowledge, the present work is the first to present results using MEMMs in an active learning framework.

An MEMM is a probabilistic model for sequence labeling. It is a Conditional Markov Model (CMM as illustrated in Figure 4.1) in which a Maximum Entropy (MaxEnt) classifier is employed to estimate the probability distribution $p(t_i|\underline{w},\underline{t}_{1\ldots i-1}) \approx p_{ME}(t_i|w_i,\underline{f_i},t_{i-1},t_{i-2})$ over possible labels for each element in the sequence—in our case, for each word w_i in a sentence $\underline{w}$. The MaxEnt model is trained from labeled data and has access to any predefined attributes (represented here by the collection $\underline{f_i}$) of the entire word sequence and to the labels of previous words ($\underline{t}_{1\ldots i-1}$). Our implementation employs an order-two Markov assumption so the classifier has access only to the two previous tags t_{i-1}, t_{i-2}. We refer to the features $(w_i,\underline{f_i},t_{i-1},t_{i-2})$ from which the classifier predicts the distribution over tags as "the local trigram context".

A Viterbi decoder is a dynamic programming algorithm that applies the MaxEnt classifier to score multiple competing tag-sequence hypotheses efficiently and to produce the best tag sequence, according to the model. We approximate Viterbi very closely using a fast beam search. Essentially, the decoding process involves sequential classification, conditioned on the (uncertain) decisions of the previous local trigram context classifications. The chosen tag sequence $\hat{\underline{t}}$ is the tag sequence maximizing the following quantity:

$$\begin{aligned}\hat{\underline{t}} &= \arg\max_{\underline{t}} P(\underline{t}|\underline{w}) \\ &= \arg\max_{\underline{t}} \prod_{i=1}^{n} p_{ME}(t_i|w_i,\underline{f_i},t_{i-1},t_{i-2}).\end{aligned}$$

The features used in this work are reasonably typical for modern MEMM feature-based POS tagging and consist of a combination of lexical, orthographic, contextual, and frequency-based information. In particular, for each word the following features are defined: the textual form of the word itself, the POS tags of the preceding two words, and the textual form of the following word. Following Toutanova and Manning [92] approximately, more information is defined for words that

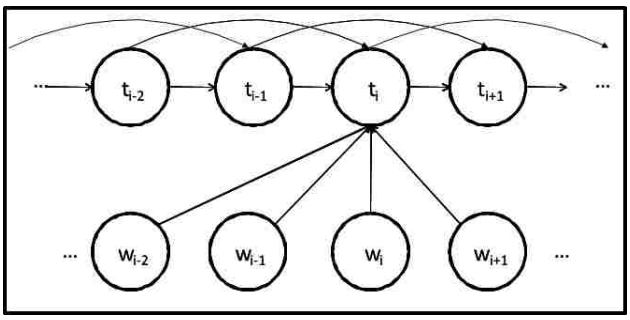


Figure 4.1: Simple Markov order 2 CMM, with focus on the *i*-th hidden label (or tag).

are considered rare (which we define here as words that occur fewer than fifteen times). We consider the tagger to be near-state-of-the-art in terms of tagging accuracy.

4.3 Active Learning

The objective of this research is to produce more high quality annotated data with less human annotator time and effort. Active learning is an approach to machine learning in which a model is trained with the selective help of an oracle. The oracle provides labels on a sufficient number of "tough" cases, as identified by the model. Easy cases are assumed to be understood by the model and to require no additional annotation by the oracle. Many variations have been proposed in the broader active learning and decision theory literature under many names, including "active sampling" and "optimal sampling."

In active learning for POS tagging, as in other applications, the oracle can be a human. For experimental purposes, a human oracle is simulated using pre-labeled data, where the labels are hidden until queried. To begin, the active learning process requires some small amount of training data to seed the model. The process proceeds by identifying the data in the given corpus that should be tagged first for maximal impact.

4.3.1 Active Learning in the Language Context

When considering the role of active learning, we were initially drawn to the work in active learning for classification. In a simple configuration, each instance (document, image, etc.) to be labeled

can be considered to be independent. However, for active learning for the POS tagging problem we considered the nature of human input as an oracle for the task. As an approximation, people read sentences as propositional atoms, gathering contextual cues from the sentence in order to assemble the meaning of the whole. Consequently, we thought it unreasonable to choose the word as the granularity for active learning. Instead, we begin with the assumption that a human will usually require much of the sentence or at least local context from the sentence in order to label a single word with its POS label. While focusing on a single word, the human may as well label the entire sentence or at least correct the labels assigned by the tagger for the sentence. Consequently, the sentence is the granularity of annotation for this work. (Future work will question this assumption and investigate tagging a word or a subsequence of words at a time.) This distinguishes our work from active learning for classification since labels are not drawn from a fixed set of labels. Rather, every sentence of length n can be labeled with a tag sequence drawn from a set of size T^n, where T is the size of the per-word tag set. Granted, many of the options have very low probability.

To underscore our choice of annotating at the granularity of a sentence, we also note that a maximum entropy classifier for isolated word tagging that leverages attributes of neighboring words–but is blind to all tags–will underperform an MEMM that includes the tags of neighboring words (usually on the left) among its features. Previous experiments demonstrate the usefulness of tags in context on the standard Wall Street Journal data from the Penn Treebank [57]. A MaxEnt isolated word tagger achieves 93.7% on words observed in the training set and 82.6% on words unseen in the training set. Toutanova and Manning [92] achieves 96.9% (on seen) and 86.9% (on unseen) with an MEMM. They surpassed their earlier work in 2003 with a "cyclic dependency network tagger," achieving 97.2%/89.05% (seen/unseen) [93]. The generally agreed upon upper bound is around 98%, due to label inconsistencies in the Treebank. The main point is that effective use of contextual features is necessary to achieve state of the art performance in POS tagging.

In active learning, we employ several sets of data that we refer to by the following names:

Initial Training the small set of data used to train the original model before active learning starts

Training data that has already been labeled by the oracle as of step i in the learning cycle

Unannotated data not yet labeled by the oracle as of step i

Test (specifically Development Test) labeled data used to measure the accuracy of the model at each stage of the active learning process. Labels on this set are held in reserve for comparison with the labels chosen by the model. It is the accuracy on this set that we report in our experimental results in Section 4.4.

Note that the Training set grows at the expense of the Unannotated set as active learning progresses.

Active Learning for POS Tagging consists of the following steps:

1. Train a model with Initial Training data
2. Apply model to Unannotated data
3. Compute potential informativeness of each sentence
4. Remove top *n* sentences with most potential informativeness from Unannotated data and give to oracle
5. Add *n* sentences annotated (or corrected) by the oracle to Training data
6. Retrain model with Training data
7. Return to step 2 until stopping condition is met.

There are several possible stopping conditions, including reaching a quality bar based on accuracy on the Test set, the rate of oracle error corrections in the given cycle, or even the cumulative number of oracle error corrections. In practice, the exhaustion of resources, such as time or money, may completely dominate all other desirable stopping conditions.

Several methods are available for determining which sentences will provide the most information. Expected Value of Sample Information (EVSI) [69] would be the optimal approach from a decision theoretic point of view, but it is computationally prohibitive and is not considered here. We also do not consider the related notion of query-by-model-improvement or other methods [2, 75]. While worth exploring, they do not fit in the context of this current work and should be

considered in future work. We focus here on the more widely used Query by Committee (QBC) and Query by Uncertainty (QBU), including our new adaptations of these.

Our implementation of maximum entropy training employs a convex optimization procedure known as LBFGS. Although this procedure is relatively fast, training a model (or models in the case of QBC) from scratch on the training data during every round of the active learning loop would prolong our experiments unnecessarily. Instead we start each optimization search with a parameter set consisting of the model parameters from the previous iteration of active learning (we call this "Fast MaxEnt"). In practice, this converges quickly and produces equivalent results.

4.3.2 Query by Committee

Query by Committee (QBC) was introduced by Seung et al. [82]. Freund et al. [29] provided a careful analysis of the approach. Engelson and Dagan [27] experimented with QBC using HMMs for POS tagging and found that selective sampling of sentences can significantly reduce the number of samples required to achieve desirable tag accuracies. Unlike the present work, Engelson & Dagan were restricted by computational resources to selection from small windows of the Unannotated set, not from the entire Unannotated set. Related work includes learning ensembles of POS taggers, as in the work of Brill and Wu [10], where an ensemble consisting of a unigram model, an N-gram model, a transformation-based model, and an MEMM for POS tagging achieves substantial results beyond the individual taggers. Their conclusion relevant to this paper is that different taggers commit complementary errors, a useful fact to exploit in active learning. QBC employs a committee of N models, in which each model votes on the correct tagging of a sentence. The potential informativeness of a sentence is measured by the total number of tag sequence disagreements (compared pair-wise) among the committee members. Possible variants of QBC involve the number of committee members, how the training data is split among the committee members, and whether the training data is sampled with or without replacement.

A potential problem with QBC in this application is that words occur with different frequencies in the corpus. Because of the potential for greater impact across the corpus, querying for the

			NN	*0 .85*
			VB	*0.13*
			...	
RB	**DT**	**JJS**	*CD*	*2.0E-7*
Perhaps	**the**	**biggest**	**hurdle**	…

Figure 4.2: Distribution over tags for the word "hurdle" in italics. The local trigram context is in boldface.

tag of a more frequent word may be more desirable than querying for the tag of a word that occurs less frequently, even if there is greater disagreement on the tags for the less frequent word. We attempted to compensate for this by weighting the number of disagreements by the corpus frequency of the word in the full data set (Training and Unannotated). Unfortunately, this resulted in worse performance; solving this problem is an interesting avenue for future work.

4.3.3 Query by Uncertainty

The idea behind active sampling based on uncertainty appears to originate with Thrun and Möller [87]. QBU has received significant attention in general. Early experiments involving QBU were conducted by Lewis and Gale [51] on text classification, where they demonstrated significant benefits of the approach. Lewis and Catlett [50] examined its application for non-probabilistic learners in conjunction with other probabilistic learners under the name "uncertainty sampling." Anderson and Moore [2] explored QBU using HMMs and concluded that it is sometimes advantageous. We are not aware of any published work on the application of QBU to POS tagging. In our implementation, QBU employs a single MEMM tagger. The MaxEnt model comprising the tagger can assess the probability distribution over tags for any word in its local trigram context, as illustrated in the example in Figure 4.2.

In Query by Uncertainty (QBU), the informativeness of a sample is assumed to be the uncertainty in the predicted distribution over tags for that sample, that is the entropy of $p_{ME}(t_i|w_i,\underline{f_i},t_{i-1},t_{i-2})$. To determine the potential informativeness of a word, we can measure

the entropy in that distribution. Since we are selecting sentences, we must extend our measure of uncertainty beyond the word.

4.3.4 Adaptations of QBU

There are several problems with the use of QBU in this context:

- Some words are more important; i.e., they contain more information perhaps because they occur more frequently.
- MaxEnt estimates per-word distributions over tags, not per-sentence distributions over tag sequences.
- Entropy computations are relatively costly.

We address the first issue in a new version of QBU which we call "Weighted Query by Uncertainty" (WQBU). In WQBU, per-word uncertainty is weighted by the word's corpus frequency.

To address the issue of estimating per-sentence uncertainty from distributions over tag *sequences*, we have considered several different approaches. The per-word (conditional) entropy is defined as follows:

$$H(T_i|w_i,\underline{f_i},t_{i-1},t_{i-2}) = -\sum_{t_i \in \text{Tagset}} p_{ME}(t_i|w_i,\underline{f_i},t_{i-1},t_{i-2}) \cdot \log p_{ME}(t_i|w_i,\underline{f_i},t_{i-1},t_{i-2})$$

where T_i is the random variable for the tag t_i on word w_i, and the features of the context in which w_i occurs are denoted, as before, by the collection $\underline{f_i}$ and the prior tags t_{i-1}, t_{i-2}. It is straightforward to calculate this entropy for each word in a sentence from the Unannotated set, if we assume that previous tags t_{i-1}, t_{i-2} are from the Viterbi (best) tag sequence (for the entire sentence) according to the model.

For an entire sentence, we estimate the tag-sequence entropy by summing over all possible tag sequences. However, computing this estimate exactly on a 25-word sentence, where each word can be labeled with one of 35 tags, would require $35^{25} = 3.99 \cdot 10^{38}$ steps. Instead, we approximate

the per-sentence tag sequence distribution entropy by summing per-word entropy:

$$\hat{H}(\underline{T}|\underline{w}) \approx -\sum_{w_i \in \underline{w}} H(T_i|w_i, \underline{f_i}, t_{i-1}, t_{i-2})$$

This is the approach we refer to as QBU in the experimental results section. We have experimented with a second approach that estimates the per-sentence entropy of the tag-sequence distribution by Monte Carlo decoding. Unfortunately, current active learning results involving this MC POS tagging decoder are negative on small Training set sizes, so we do not present them here. Another alternative approximation worth pursuing is computing the per-sentence entropy using the n-best POS tag sequences. Very recent work by Mann and McCallum [55] proposes an approach in which exact sequence entropy can be calculated efficiently. Further experimentation is required to compare our approximation to these alternatives.

An alternative approach that eliminates the overhead of entropy computations entirely is to estimate per-sentence uncertainty with $1 - P(\hat{t})$, where $\hat{t}$ is the Viterbi (best) tag sequence. We call this scheme QBUV. In essence, it selects a sample consisting of the sentences having the highest probability that the Viterbi sequence is wrong. To our knowledge, this is a novel approach to active learning.

4.4 Experimental Results

In this section, we examine the experimental setup, the prose and poetry data sets, and the results from using the various active learning algorithms on these corpora.

4.4.1 Setup

The experiments focus on the annotation scenario posed earlier, in which budgetary constraints afford only some number x of sentences to be annotated. The x-axis in each graph captures the number of sentences. For most of the experiments, the graphs present accuracies on the (Development)

Test set. Later in this section, we present results for an alternate metric, namely number of words corrected by the oracle.

In order to ascertain the usefulness of the active learning approaches explored here, the results are presented against a baseline in which sentences are selected randomly from the Unannotated set. We consider this baseline to represent the use of a state-of-the-art tagger trained on the same amount of data as the active learner. Due to randomization, the random baseline is actually distinct from experiment to experiment without any surprising deviations. Also, each result curve in each graph represents the average of three distinct runs.

Worth noting is that most of the graphs include active learning curves that are run to completion; namely, the rightmost extent of all curves represents the exhaustion of the Unannotated data. At this extreme point, active learning and random sample selection all have the same Training set. In the scenarios we are targeting, this far right side is not of interest. Points representing smaller amounts of annotated data are our primary interest.

In the experiments that follow, we address several natural questions that arise in the course of applying active learning. We also compare the variants of QBU and QBC. For QBC, committee members divide the training set (at each stage of the active learning process) evenly. All committee members and final models are MEMMs. Likewise, all variants of QBU employ MEMMs.

4.4.2 Data Sets

The experiments involve two data sets in search of conclusions that generalize over two very different kinds of English text. The first data set consists of English prose from the POS-tagged one-million-word Wall Street Journal text in the Penn Treebank (PTB) version 3. We use a random sample of the corpus constituting 25% of the traditional training set (sections 2-21). Initial Training data consists of 1% of this set. We employ section 24 as the Development Test set. Average sentence length is approximately 25 words.

Our second experimental set consists of English poetry from the British National Corpus (BNC) [31, 41, 70]. The text is also fully tagged with 91 parts of speech from a different tag set

	100	200	400	800	1600	3200	6400
QBU	76.26	**86.11**	**90.63**	**92.27**	93.67	94.65	95.42
QBUV	**76.65**	85.09	89.75	92.24	**93.72**	**94.96**	**95.60**
QBC	76.19	85.77	89.37	91.78	93.49	94.62	95.36
Base	76.57	82.13	86.68	90.12	92.49	94.02	95.19

Table 4.1: The best models (on PTB WSJ data) with various amounts of annotation (columns).

than the one used for the PTB. The BNC XML data was taken from the files B1C.xml, CBO.xml, and H8R.xml. This results in a set of 60,056 words and 8,917 sentences.

4.4.3 General Results

To begin, each step in the active learning process adds a batch of 100 sentences from the Unannotated set at a time. Figure 4.3 demonstrates (using QBU) that the size of a query batch is not significant in these experiments.

The primary question to address is whether active learning helps or not. Figure 4.4 demonstrates that QBU, QBUV, and QBC all outperform the random baseline in terms of total, per-word accuracy on the Test set, given the same amount of Training data. Figure 4.5 is a close-up version of Figure 4.4, placing emphasis on points up to 1000 annotated sentences. In these figures, QBU and QBUV vie for the best performing active learning algorithm. These results appear to give some useful advice captured in Table 4.1. The first column in the table contains the starting conditions. The remaining columns indicate that for between 800–1600 sentences of annotation, QBUV takes over from QBU as the best selection algorithm.

The next question to address is how much initial training data should be used; i.e., when should we start using active learning? The experiment in Figure 4.6 demonstrates (using QBU) that one should use as little data as possible for Initial Training Data. There is always a significant advantage to starting early. In the experiment documented in this figure, a batch query size of one was employed in order to make the point as clearly as possible. Larger batch query sizes produce a graph with similar trends as do experiments involving larger Unannotated sets and other active learners.

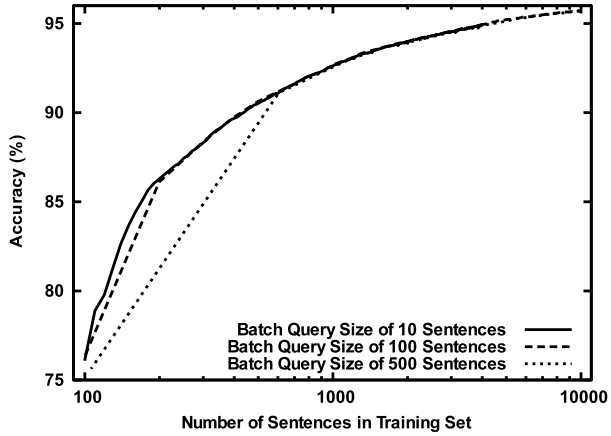


Figure 4.3: Varying the size of the query batch in active learning yields identical results after the first query batch.

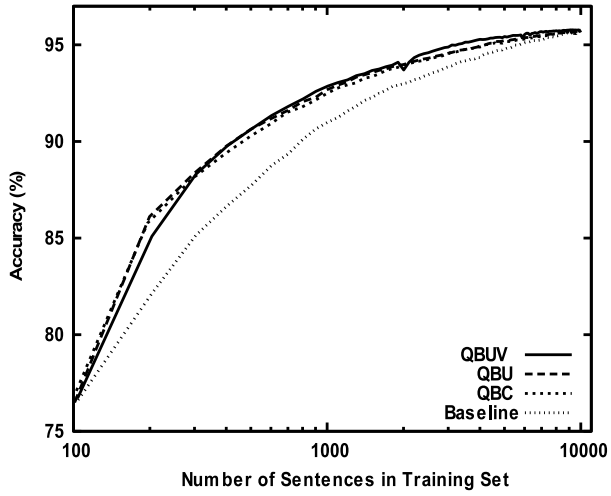


Figure 4.4: The best representatives of each type of active learner beat the baseline. QBU and QBUV trade off the top position over QBC and the Baseline.

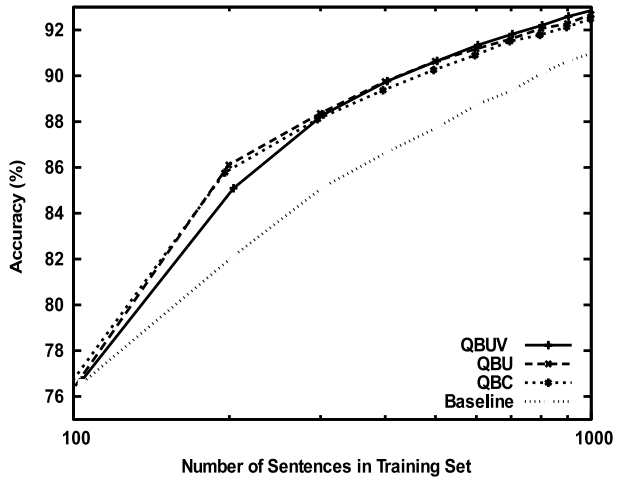


Figure 4.5: Close-up of the low end of the graph from Figure 4.4. QBUV and QBU are nearly tied for best performance. this figure, a batch query size of one was employed in order to make the point as clearly as possible. Larger batch query sizes produce a graph with similar trends as do experiments involving larger Unannotated sets and other active learners.

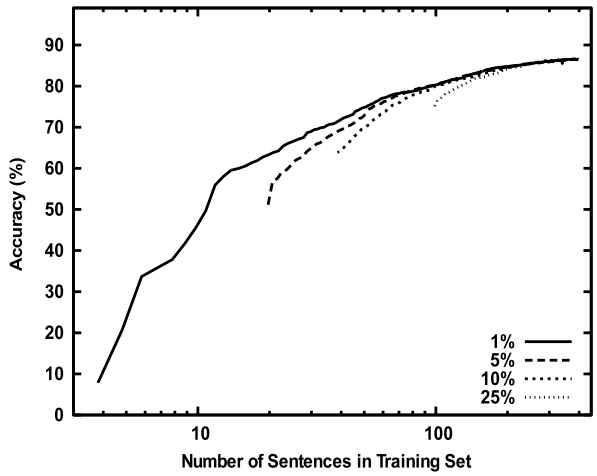


Figure 4.6: Start active learning as early as possible for a head start.

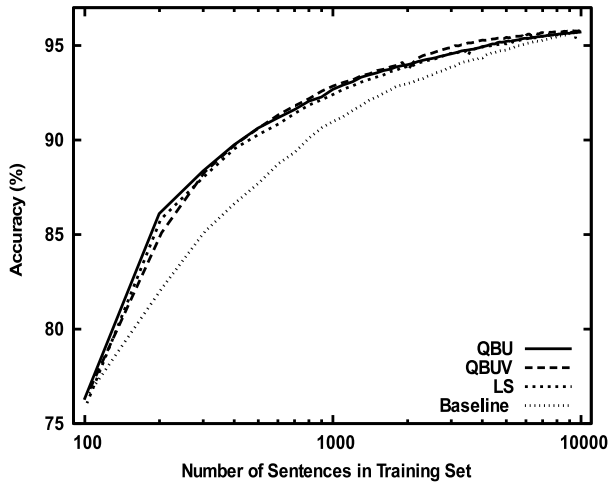


Figure 4.7: QBUV is superior to QBU overall, but QBU is better for very low counts. Both are superior to the random baseline and the Longest Sentence (LS) baseline.

4.4.4 QBC Results

An important question to address for QBC is what number of committee members produces the best results? There was no significant difference in results from the QBC experiments when using between 3 and 7 committee members. For brevity we omit the graph.

4.4.5 QBU Results

For Query by Uncertainty, the experiment in Figure 4.7 demonstrates that QBU is superior to QBUV for low counts, but that QBUV slightly overtakes QBU beyond approximately 300 sentences. In fact, all QBU variants, including the weighted version, surpassed the baseline. WQBU has been omitted from the graph, as it was inferior to straight-forward QBU.

4.4.6 Results on the BNC

Next we introduce results on poetry from the British National Corpus. Recall that the feature set employed by the MEMM tagger was optimized for performance on the Wall Street Journal. For the experiment presented in Figure 4.8, all data in the Training and Unannotated sets is from the BNC, but we employ the same feature set from the WSJ experiments. This result on the BNC data shows

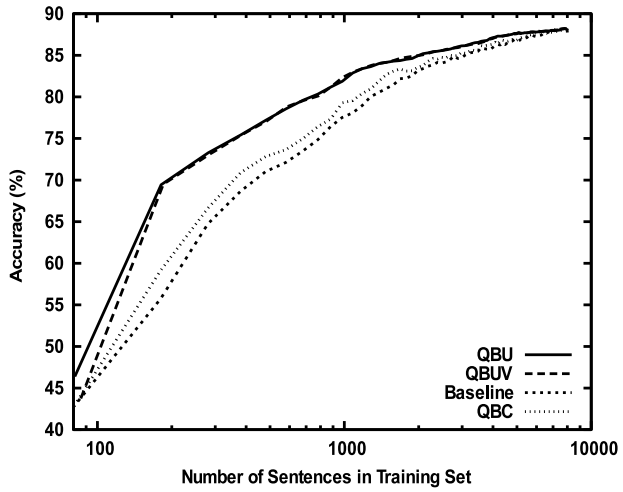


Figure 4.8: Active learning results on the BNC poetry data. Accuracy of QBUV, QBU, and QBC against the random baseline. QBU and QBUV are nearly indistinguishable.

first of all that tagging poetry with this tagger leaves a final shortfall of approximately 8% from the WSJ results. Nonetheless and more importantly, the active learning trends observed on the WSJ still hold. QBC is better than the baseline, and QBU and QBUV trade off for first place. Furthermore, for low numbers of sentences, it is overwhelmingly to one's advantage to employ active learning for annotation.

4.4.7 Another Perspective

Next, briefly consider a different metric on the vertical axis. In Figure 4.9, the metric is the total number of words changed (corrected) by the oracle. This quantity reflects the cumulative number of differences between the tagger's hypothesis on a sentence (at the point in time when the oracle is queried) and the oracle's answer (over the training set). It corresponds roughly to the amount of time that would be required for a human annotator to correct the tags suggested by the model. This figure reveals that QBUV makes significantly more changes than QBU, QBC, or LS (the Longest Sentence baseline). Hence, the superiority of QBU over QBUV, as measured by this metric, appears to outweigh the small wins provided by QBUV when measured by accuracy alone. That said, the random baseline makes the fewest changes of all. If this metric (and not some combination with accuracy) were our only consideration, then active learning would appear not to serve our needs.

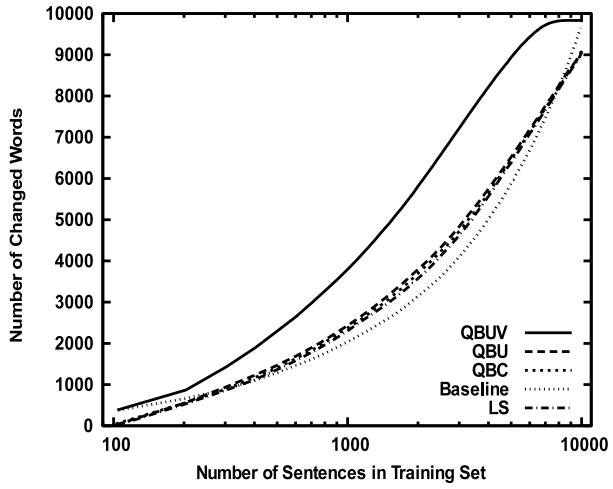


Figure 4.9: Cumulative number of corrections made by the oracle for several competitive active learning algorithms. QBU requires fewer corrections than QBUV.

This metric is also a measure of how well a particular query algorithm selects sentences that especially require assistance from the oracle. In this sense, QBUV appears most effective.

4.5 Conclusions

Active learning is a viable way to accelerate the efficiency of a human annotator and is most effective when done as early as possible. We have presented state-of-the-art tagging results using a fraction of the labeled data. QBUV is a cheap approach to performing active learning, only to be surpassed by QBU when labeling small numbers of sentences. We are in the midst of conducting a user study to assess the true costs of annotating a sentence at a time or a word at a time. We plan to incorporate these specific costs into a model of cost measured in time (or money) that will supplant the metrics reported here, namely accuracy and number of words corrected. As noted earlier, future work will also evaluate active learning at the granularity of a word or a subsequence of words, to be evaluated by the cost metric.

4.6 Errata

This chapter corrects two errors in the references section of the published version of this paper. First, the reference to Raiffa and Schlaifer [69] now has the correct year. Second, there were two citations to Roy and McCallum [75] with different titles; we have retained only the correct citation.

Chapter 5

Assessing the Costs of Machine-Assisted Corpus Annotation Through a User Study[†]

Abstract

Fixed, limited budgets often constrain the amount of expert annotation that can go into the construction of annotated corpora. Estimating the cost of annotation is the first step toward using annotation resources wisely. We present here a study of the cost of annotation. This study includes the participation of annotators at various skill levels and with varying backgrounds. Conducted over the web, the study consists of tests that simulate machine-assisted pre-annotation, requiring correction by the annotator rather than annotation from scratch. The study also includes tests representative of an annotation scenario involving Active Learning as it progresses from a naïve model to a knowledgeable model; in particular, annotators encounter pre-annotation of varying degrees of accuracy. The annotation interface lists tags considered likely by the annotation model in preference to other tags. We present the experimental parameters of the study and report both descriptive and inferential statistics on the results of the study. We conclude with a model for estimating the hourly cost of annotation for annotators of various skill levels. We also present models for two granularities of annotation: sentence at a time and word at a time.

[†]Eric Ringger, Marc Carmen, Robbie Haertel, Kevin Seppi, Deryle Londsale, Peter McClanahan, James Carroll, and Noel Ellison. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2008.

5.1 Introduction

In the construction of annotated corpora, we are constrained by fixed budgets for expert annotation. Although a fully annotated corpus is required, we can afford only to label a subset. Obtaining human annotations for linguistic data is labor-intensive and typically the costliest part of the acquisition of an annotated corpus. Hence, there is strong motivation to reduce annotation costs, but not at the expense of quality.

The current work focuses on part-of-speech tagging, although other annotation tasks can also benefit from the techniques discussed. In addition to a labeled corpus, we also aim to produce a probabilistic tagger that can accurately tag future texts. The annotation environment incorporates the probabilistic tagger in order to facilitate the annotation process: annotators are able to focus on correcting the tough cases missed by the automatic tagger while avoiding work on the cases tagged correctly in the pre-annotation. Future work will evaluate the comparative merits of annotation from scratch versus correction of pre-tagged text. In this work, a probabilistic part-of-speech tagger is incrementally trained from the labeled subset of a given corpus and employed to tag automatically the remainder of the corpus.

One important question that naturally arises in this setting of machine-assisted annotation is how to best focus the attention and expertise of the human annotator(s). One aspect of this question is: on which instances in the data should the annotators focus? This is the question addressed by active learning. Active Learning (AL) can be employed to reduce the costs of corpus annotation [27, 73, 91]. Our previous work [73] demonstrates that by applying active learning techniques, a state of the art tagging model can be trained on as little as one-half of the amount of data required by more traditional, less strategic annotation schemes to achieve the same levels of accuracy. With the assistance of AL, the role of the human oracle is either to label a datum of interest or simply to correct the label choice of an automatic labeler. AL directs an annotator's attention to those data which are likely to be maximally informative according to a given tagging model. In AL, the learner leverages newly provided annotations to select more informative sentences and to provide more accurate annotations in future iterations. Ideally, this process yields accurate labels with less human

annotation. Several heuristic AL methods have been investigated for determining which data will provide the most information and hopefully the best accuracy. Perhaps the best known are Query by Committee (QBC) [82] and Uncertainty Sampling (or Query by Uncertainty, QBU) [87].

A second aspect of the focus question is: at which granularity should the annotators direct their efforts? This paper focuses on this particular aspect and describes a user study designed specifically to assess the true cost of labeling a whole sentence or just a word at a time. Annotation cost is project-dependent. For instance, annotators may be paid by the hour or for the number of annotations they produce (measured in words or sentences). With few exceptions, much of the previous work on AL has largely ignored the question of cost estimation. One exception is Ngai and Yarowsky [61] who compare the cost of manual rule-writing with annotation using AL for noun phrase chunking.

In our previous work [73], we assumed that the unit of annotation was a sentence. The assumption was based on *a priori* consideration of the nature of human input as an oracle for the POS tagging task. We reasoned that people gather contextual cues from a sentence in order to assemble the meaning of the whole. Consequently, we began with the assumption that a human annotator will usually require significant context from the sentence in order to label a single word with its POS label. We also reasoned that while focusing on a single word, the human may as well label (or correct the labels on) the entire sentence. The user study reported here questions the sentential assumption and tests its effectiveness against the word-at-a-time alternative.

In the following sections, we describe the experimental design of the study, the methods for data selection, the implementation of the web-based study itself, the user pool, and a statistical summary of the data produced by the study. The results allow us to make recommendations for the granularity of annotation and provide guidance for a model of the true hourly cost of annotation.

5.2 Experimental Design

5.2.1 Conditions

To address the question about the granularity of annotation, we construct an experiment designed to assess the cost of two alternatives: we ask users of a web-browser based interface to correct parts of speech on entire sentences and, separately, to correct the part of speech of individual words. For data annotation, time is money, so our measure of cost is the time required to annotate.

The reason for correction instead of de novo annotation is that in a machine-assisted annotation framework, we have access to labels from the statistical tagger. In this work we assume that machine assistance is always of some value, thus we do not test the case where words are tagged without assistance from the statistical tagger. Testing this assumption is the subject of future work.

For the second condition, correcting the tag on a single word, tags on neighboring words are hidden to avoid time wasted due to the distractions offered by tags on those neighboring words. We wish to avoid the potential cognitive load incurred on an annotator by the puzzlement related to seeing incorrect tags on words that cannot be corrected.

5.2.2 Control Variables

Our experiment includes two control variables. The first control variable is the accuracy of the tagger producing the tags to be corrected by the annotator. This variable allows us to determine the impact of tagger accuracy at various stages during the annotation process. In particular, for this first control variable, we employ statistical taggers of known error rates (created using a development corpus with known tags). Our study included tests using probabilistic taggers with accuracies: 50%, 75%, and 95%. This progression of increasing accuracy is typical in the process of active learning; hence, the data sheds some light on the time required to annotate in such circumstances. For tagging, we employed a probabilistic tagger, namely a Maximum Entropy Conditional Markov Model tagger

[71, 92, 93]. Such taggers are referred to alternatively in the literature as MaxEnt CMMs, MEMMs, or simply "MaxEnt" taggers.

The second control variable is the sentence length. Sentence length is discretized into three ranges: 1–15 words, 16–21 words, and 22–29 words. By selecting data for the study belonging to these three ranges, we are able to assess the impact of sentence length on the final cost.

5.2.3 Session Size

For the two conditions (entire sentences and word-at-a-time), three values of the first controlled variable (tagger accuracy), and three values of the second (discretized sentence length), we have eighteen different types of cases. Furthermore, we reasoned that each user should provide input on at least two examples of each of the eighteen case types. Consequently, in addition to practice and control sentences (4 each), each user responds to 36 cases, for a total of 44. As for the order of presentation, we decided to alternate between cases for each condition, beginning with a whole sentence annotation case, followed by a single word annotation case. The order of presentation was otherwise randomized without respect to the two control variables.

5.2.4 Data Selection

The study employs English prose from the Wall Street Journal portion of the Penn Treebank. This data set is well known, and the quality and shortcomings of its annotations are well understood. Intuitively, data annotated by a more accurate model will require that fewer tags be corrected, thus requiring less annotation time. Similarly, longer sentences will be more costly than shorter sentences in general. As noted above, we determined that annotators could tag 36 cases, not including practice and control cases, in a reasonable amount of time. We estimated that we would be able to obtain data from a minimum of 25–30 annotators. We thus decided to produce 14 non-overlapping sets of 36 unique cases. From these, we produced 28 total "templates": each of the 14 sets are used in one template to be annotated word-at-a-time and in a second template as sentence-at-a-time. Consequently, the same cases are guaranteed to be annotated using both methods. Once more than

Table 5.1: Accuracy figures for the three models used to select sentences.

Model	Tag Accuracy	Unknown Word Tag Accuracy	Sentence Tag Accuracy
Tag50	54.3%	47.1%	0.4%
Tag75	75.1%	65.6%	1.5%
Tag95	95.3%	86.9%	36.9%

28 users had participated in the study, templates were re-used. We control for model accuracy by training three MaxEnt POS taggers of differing quality in an iterative fashion as follows. After the available training data (sections 2–21 of the Penn Treebank) was randomly ordered, a small batch of sentences was added to the (initially empty) annotated set, the model was retrained, and its accuracy on a held out set (section 24 of the PTB) was computed.

This process was repeated until the desired accuracy was achieved and the resulting model was saved. This was done for each of three desired accuracy levels: 50% (Tag50), 75% (Tag75), and 95% (Tag95); the actual accuracies obtained are shown in Table 5.1.

In order to control for length, we fit a log-normal distribution to the lengths of the sentences in the training data. The 25th, 50th, and 75th percentiles were used to create three distinct length buckets of [1, 15], [16, 21], and [22, 30] words, respectively. We excluded sentences having more than 30 words. We considered it better to present to the annotators a larger number of sentences in the time allotted to annotators than to have annotations on extremely long sentences.

The sentences in these length buckets are further divided into six equal parts in order to control for the other factors: two buckets (one for sentence at a time and one for word at a time annotation) for each of the three trained models (tag50, tag75 and tag90). The model with the appropriate accuracy was used to sort the sentences within each of the 18 buckets using the model corresponding to the bucket, and the first 28 sentences in each bucket were set aside. To ensure that a single template did not consist of all of the hardest sentences or words, we independently shuffled the 28 remaining sentences in each of the buckets. A template was created by taking two sentences from each of the 18 buckets; half of them are presented the same way they were chosen (i.e., to be annotated as a full sentence if chosen using the sentence-at-a-time algorithm), and the other half were presented using the word-at-a-time method.

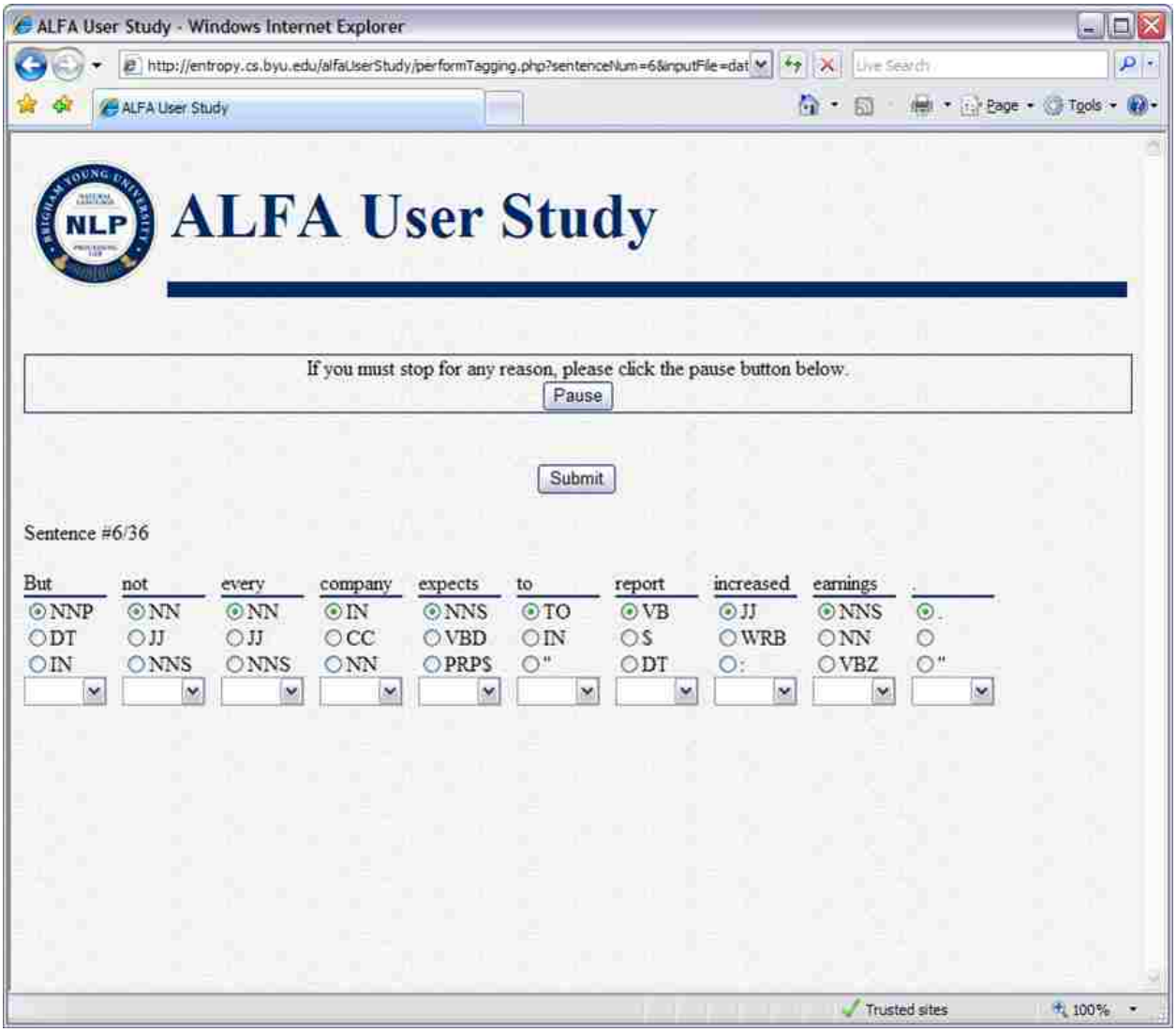


Figure 5.1: Web interface for the correction of tags for an entire sentence.

5.2.5 User Interface

The web study was implemented as a PHP application, and all session templates and results were encoded in XML. On the matter of how to present the cases to the user, we settled on the use of radio buttons for the top three tags with a drop-down for the other options. The top three tags are the three most likely correct tags as determined by the probabilistic tagger. The order of the tags in the drop-down list is static so that users do not have to "hunt" for the location of the desired tag. Figures 5.1 and 5.2 illustrate with screenshots of the interface for the annotation of sentences and words, respectively.

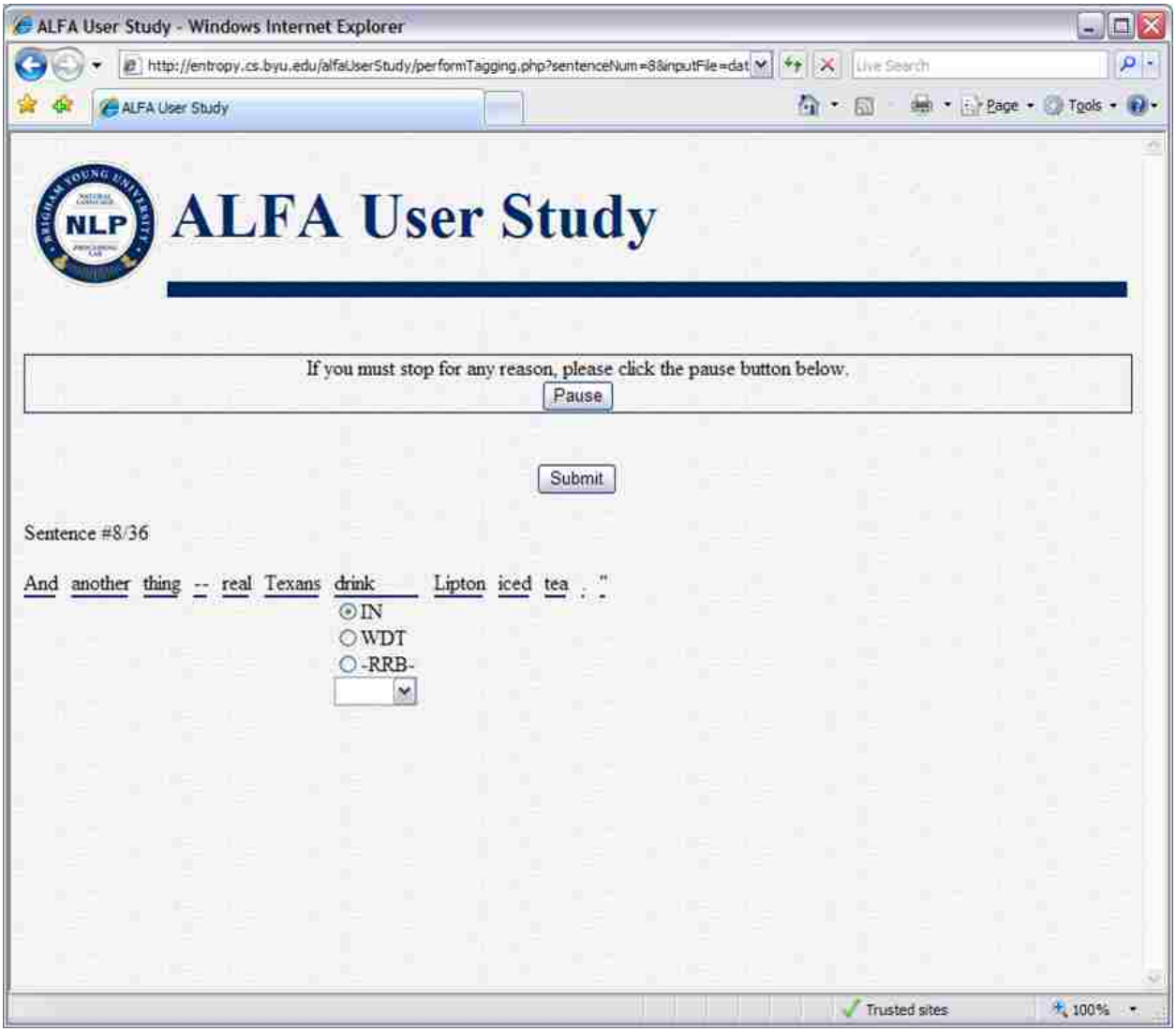


Figure 5.2: Web interface for the correction of tags for an individual word in sentential context.

5.2.6 Subjects

The majority of the subjects for this study were undergraduate linguistics students in their third week of an undergraduate syntax course. They had received a general review of phrase types and word classes. The students received no special training in part of speech tagging, the tag set employed in the Penn Treebank, or even linguistic experimentation. The assignment was optional, and the equivalent value of a small homework assignment was awarded upon completion of the study. They were also rewarded for enrolling up to two additional people in the study. Ten of the 47 students were non-native speakers of English, and four of the 47 had participated in an earlier round of the study. When subjects finished the study they were given a brief survey in which they were asked several questions related to their ability and their performance in the study, as described in greater detail below.

5.3 Descriptive Statistics

We present the results of this study in two parts: first we introduce the data gathered and present descriptive statistics. Second, we model the data and draw conclusions using inferential statistics. The web-based annotation tool described in the preceding sections gathered the following data during the annotation study:

Length The number of tokens in the sentence; when annotating a single word it is the length of the sentence in which the word appears

Time The time in seconds that the subject spent on the current case

Subject Accuracy The percentage of tokens correctly tagged by the subject. When annotating a single word this is either 0% or 100%

Location Index of the current case in the session

Tagger Accuracy The percentage of words correctly tagged by the automatic tagger in the sentence. When annotating a single word this is either 0% or 100%

Table 5.2: Statistics for Sentence-at-a-time and word-at-a-time annotation.

Label	Sentence					Word				
	Mean	Std Dev	Median	Min	Max	Mean	Std Dev	Median	Min	Max
Length	19.86	5.74	20	4	30	19.85	5.77	20	4	30
Time (Seconds)	137.02	141.69	97.69	0.02	1448.67	19.4	21.92	14.09	0.14	299.23
Subject Accuracy	78.37	16.18	80.95	5	100	69.92	45.88	100	0	100
Location	22.91	12.66	23	1	50	22.42	12.93	22	2	49
Tagger Accuracy	63.68	22.86	65	5	100	32.21	46.75	0	0	100
Number Needing Correction	7.21	5.07	7	0	23	7.35	5.04	7	0	23
Percent Done	51.7	28.54	52.27	2	100	50.63	29.17	50	4	98
Conditional Entropy	30.51	16.81	31.09	3.74	75.23	30.64	16.83	31.09	3.74	75.23
From Tagger	73.74	16.68	75	50	95	73.62	16.66	75	50	95
Native English Speaker	0.79	0.41	1	0	1	0.79	0.41	1	0	1
Previously Participated in Study	0.09	0.29	0	0	1	0.09	0.28	0	0	1
Self Evaluation Tagging Proficiency	2.11	0.72	2	1	3	2.1	0.72	2	1	3
Self Evaluation of Performance in Study	2.77	0.83	3	1	5	2.76	0.83	3	1	5
Time per Tag	6.98	7.01	5.09	0	68.98	19.4	21.92	14.09	0.14	299.23

Number Needing Correction The number of words in the case needing correction

Percent Done Percentage of the cases assigned to the current subject already encountered

Conditional Entropy For whole sentence annotation, an estimate of the total tag sequence entropy given the words in the current sentence. For single word annotation, the entropy of the tag distribution for the current word.

From Tagger The accuracy (50, 75, or 95) of the tagger providing the candidate tags

Native English Speaker A 0/1 indicator of whether the subject is a native English speaker

Previously Participated in Study A 0/1 indicator of whether the subject was part of a previous (similar) tagging exercise

Self Evaluation Tagging Proficiency A 1/2/3/4/5 indicator of the subject self-evaluation of tagging proficiency

Self Evaluation of Performance in Study A 1/2/3/4/5 indicator of the subject self-evaluation of tagging accuracy in this study

Descriptive statistics on these attributes are shown in Table 5.2. The final row in the table measures the time required per tag. For the word-at-a-time case, this is just the value of the Time variable above. For the sentence-at-a-time case, the time per tag is the ratio of the time per sentence by the length of the sentence.

5.4 Hourly Cost Models

We are interested primarily in linear models predictive of the time required for annotation/correction tasks. Based on the data from the annotation user study, we derive such models. We will refer to these models as "hourly cost models." First we focus only on the data annotated/corrected one sentence at a time. There were 1046 annotated sentences in the data from the user study. We discarded extreme outliers having time less than or equal to five seconds or greater than or equal to 1000 seconds. Such outliers are probably best explained by the failure of a subject to use the "pause" button in the web interface or by negligent speeding through the study. This left 906 sentences. The hourly cost model computed on that data by means of linear regression is as follows:

$$h = \frac{3.795 \cdot l + 5.387 \cdot c + 12.57}{3600}$$

where h is the time in hours spent on the sentence, l is the number of tokens in the sentence, and c is the number of words in the sentence needing correction. The model uses only a small subset of the raw statistics available in the annotation study for two reasons: first, some variables (e.g., proficiency assessment) are not included because we explicitly wish to assume that tagging will be conducted by a mix of people with tagging skills similar to the mix of skills tested in the user study. Second, some variables fail to have a statistically meaningful effect on the resultant model. We employed the Bayesian Information Criterion (as implemented in the LEAPS package in R) to assess which of the variables listed in Section 5.3 should be included in the model. For this model, the Residual Standard Error (RSE) is 89.5, and the adjusted correlation (R^2) is 0.181.

The model has an intuitive interpretation: the annotator considers each word and decides whether or not it needs to be corrected (3.795 seconds per word); only words needing correction are changed (5.387 seconds per correction). Additionally, there is 12.57 seconds of overhead per sentence. In contrast to the model presented in Ngai and Yarowsky [61] which predicts monetary cost given time spent, this model estimates time spent from characteristics of a sentence. Many of

the costs employed in other work [42, 63] can be seen as estimating only some portion of the hourly cost. These distinctions make our work a novel contribution.

This model reflects the abilities of the annotators in the study may not be representative of expert annotators hired for other annotation work. A better model, linear or otherwise, based on data from other annotators could be employed, but the methodology employed in this study could be applied directly.

We also found the following relationships. Interestingly, participation by non-native speakers of English did not appear to affect accuracy but does affect completion time. As noted above, each subject was asked to rate his proficiency in tagging; self evaluation is statistically significant in relationship to the subject's accuracy. Also, a subject's self evaluation and correction accuracy are correlated. Whether or not a subject had participated in a similar previous experiment had no statistically significant impact on subject accuracy or on time to completion. "Conditional entropy" of the tag sequence distribution given the words has a negative effect on the subject's accuracy: as entropy increases, subject accuracy decreases.

We have also entertained other questions regarding the sentence-at-a-time data. If we consider only the self-rated experts in the data set, in other words if we examine only the data from subjects whose proficiency rating is 3, then we are limited to 300 sentences in the data. We apply the same linear regression techniques, and the resulting hourly model is:

$$h = \frac{4.261 \cdot l + 4.683 \cdot c - 5.579}{3600}$$

As before, l = length, and c = the number of tokens needing correction. For this model, the RSE is 76.57, and the adjusted R^2 is 0.2345.

If we consider only the beginners (self rating is 2 or lower), then we are limited to 606 sentences in the data. The resulting model is:

$$h = \frac{3.441 \cdot l + 3.441 \cdot c + 20.752}{3600}$$

For this model, the RSE is 94.98, and the adjusted R^2 is 0.1622.

Next, we analyze the sentences having high annotation accuracy. The number of sentences having an annotation accuracy of at least 95% (annotations come from the user study subjects) is 111. The linear model on this subset of the data is:

$$h = \frac{0.711 \cdot l + 5.174 \cdot c + 47.876}{3600}$$

For this model, the RSE is 76.64, and the adjusted R^2 is 0.1109.

For the word-at-a-time data, there were 1035 cases of annotated words. We discarded extreme outliers having time less than or equal to one second or greater than or equal to 200 seconds. This left 915 sentences. The hourly cost model computed on the word-at-a-time data by means of linear regression is as follows:

$$h = \frac{14.193 + 5.670 \cdot b}{3600}$$

where b is a binary indicator reflecting whether or not the word in question actually needed correction. As before, this model was preferred by the Bayesian Information Criterion. For this model, the RSE was 15.76, and the adjusted R^2 is 0.0256. The next best model according to BIC included the length of the sentence in which the word occurred, as in the sentence-at-a-time model. Intuitively, length may play a small role, for instance, affecting the time to scan for and find the word to be tagged; also, in longer sentences larger context may be needed by the subject to disambiguate more distant co-reference.

5.5 Future Work

Based on this analysis, we have simple linear models with which to predict the time required to annotate data using each presentation technique, whether sentence-at-a-time or word-at-a-time. Our future work focuses on the application of these results in the context of active learning. Our previous work demonstrates that by applying active learning techniques, a state of the art tagging model can be trained on as little as one-half of the amount of data required by more traditional

machine-assisted annotation schemes to achieve the same levels of accuracy. These results assumed that cost was measured in terms of the number of sentences annotated. We will also evaluate the cost of annotation with the new models and assess overall cost reductions in terms of time and therefore money. It is ultimately expenditures of money that are limited by our project budgets. We also plan to apply the models of annotation cost derived here in the current work to select the presentation style for annotation. By adaptively presenting cases one word-at-a-time or one sentence-at-a-time, we expect to be able to further minimize annotation time and, therefore, cost.

5.6 Addendum

The careful reader will have noticed relatively low R^2 values for the various linear models (as noted by the RSE). We note that intra-annotator variance tends to be high for non-trivial annotation tasks (c.f. [81]). This is especially the case in this study which includes non-native English speakers and, more generally, a wide array of skills. Concerning the per-word model, there is high variance across words, also evidenced by the RSE. Finally, we note the R^2 values for the sentence-at-a-time models are on par with that reported by Settles et al. [81] for his CKB task for the model pooled across all annotators (though individual annotators are much more predictable).

Chapter 6

Assessing the Costs of Sampling Methods in Active Learning for Annotation[†]

Abstract

Traditional Active Learning (AL) techniques assume that the annotation of each datum costs the same. This is not the case when annotating sequences; some sequences will take longer than others. We show that the AL technique which performs best depends on how cost is measured. Applying an hourly cost model based on the results of an annotation user study, we approximate the amount of time necessary to annotate a given sentence. This model allows us to evaluate the effectiveness of AL sampling methods in terms of time spent in annotation. We achieve a 77% reduction in hours from a random baseline to achieve 96.5% tag accuracy on the Penn Treebank. More significantly, we make the case for measuring cost in assessing AL methods.

6.1 Introduction

Obtaining human annotations for linguistic data is labor intensive and typically the costliest part of the acquisition of an annotated corpus. Hence, there is strong motivation to reduce annotation costs, but not at the expense of quality. Active learning (AL) can be employed to reduce the costs of corpus annotation [27, 73, 91]. With the assistance of AL, the role of the human oracle is either to label a datum or simply to correct the label from an automatic labeler. For the present work, we assume that correction is less costly than annotation from scratch; testing this assumption is

[†]Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. Assessing the costs of sampling methods in active learning for annotation. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, OH, USA, June 2008.

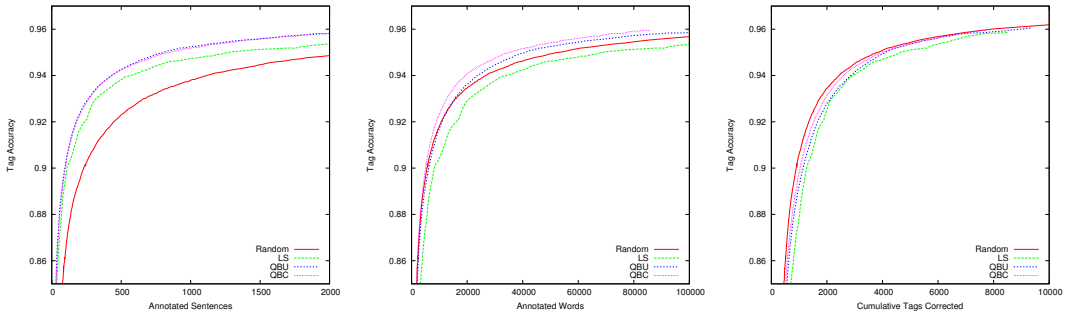
the subject of future work. In AL, the learner leverages newly provided annotations to select more informative sentences which in turn can be used by the automatic labeler to provide more accurate annotations in future iterations. Ideally, this process yields accurate labels with less human effort.

Annotation cost is project dependent. For instance, annotators may be paid for the number of annotations they produce or by the hour. In the context of parse tree annotation, Hwa [43] estimates cost using the number of constituents needing labeling and Osborne and Baldridge [63] use a measure related to the number of possible parses. With few exceptions, previous work on AL has largely ignored the question of actual labeling *time*. One exception is [61] (discussed later) which compares the cost of manual rule writing with AL-based annotation for noun phrase chunking. In contrast, we focus on the performance of AL algorithms using different estimates of cost (including time) for part of speech (POS) tagging, although the results are applicable to AL for sequential labeling in general. We make the case for measuring cost in assessing AL methods by showing that the choice of a cost function significantly affects the choice of AL algorithm.

6.2 Benefit and Cost in Active Learning

Every annotation task begins with a set of un-annotated items $\mathscr{U}$. The ordered set $\mathscr{A} \subseteq \mathscr{U}$ consists of all annotated data after annotation is complete or after available financial resources (or time) have been exhausted. We expand the goal of AL to produce the annotated set $\hat{\mathscr{A}}$ such that the benefit gained is maximized and cost is minimized.

In the case of POS tagging, tag accuracy is usually used as the measure of benefit. Several heuristic AL methods have been investigated for determining which data will provide the most information and hopefully the best accuracy. Perhaps the best known are Query by Committee (QBC) [82] and uncertainty sampling (or Query by Uncertainty, QBU) [87]. Unfortunately, AL algorithms such as these ignore the cost term of the maximization problem and thus assume a uniform cost of annotating each item. In this case, the ordering of annotated data $\mathscr{A}$ will depend entirely on the algorithm's estimate of the expected benefit.



(a) (b) (c)

Figure 6.1: QBU, LS, QBC, and the random baseline plotted in terms of accuracy versus various cost functions: (a) number of sentences annotated; (b) number of words annotated; and (c) number of tags corrected.

However, for AL in POS tagging, the cost term may not be uniform. If annotators are required to change only those automatically generated tags that are incorrect, and depending on how annotators are paid, the cost of tagging one sentence can depend greatly on what is known from sentences already annotated. Thus, in POS tagging both the benefit (increase in accuracy) and cost of annotating a sentence depend not only on properties of the sentence but also on the order in which the items are annotated.

Therefore, when evaluating the performance of an AL technique, cost should be taken into account. To illustrate this, consider some basic AL algorithms evaluated using several simple cost metrics. The results are presented against a random baseline which selects sentences at random; the learning curves represent the average of five runs starting from a random initial sentence. If annotators are paid by the sentence, Figure 6.1a presents a learning curve indicating that the AL policy that selects the longest sentence (LS) performs rather well. Figure 6.1a also shows that given this cost model, QBU and QBC are essentially tied, with QBU enjoying a slight advantage. This indicates that if annotators are paid by the sentence, QBU is the best solution, and LS is a reasonable alternative. Next, Figure 6.1b illustrates that the results differ substantially if annotators are paid by the word. In this case, using LS as an AL policy is worse than random selection. Furthermore, QBC outperforms QBU. Finally, Figure 6.1c shows what happens if annotators are paid by the number of word labels corrected. Notice that in this case, the random selector marginally outperforms the other

techniques. This is because QBU, QBC, and LS tend to select data that require many corrections. Considered together, Figures 6.1a-Figure 6.1c show the significant impact of choosing a cost model on the relative performance of AL algorithms. This leads us to conclude that AL techniques should be evaluated and compared with respect to a specific cost function.

While not all of these cost functions are necessarily used in real-life annotation, each can be regarded as an important component of a cost model of payment by the hour. Since each of these functions depends on factors having a significant effect on the perceived performance of the various AL algorithms, it is important to combine them in a way that will accurately reflect the true performance of the selection algorithms.

In prior work, we describe such a cost model for POS annotation on the basis of the time required for annotation [74]. We refer to this model as the "hourly cost model". This model is computed from data obtained from a user study involving a POS annotation task. In the study, timing information was gathered from many subjects who annotated both sentences and individual words. This study included tests in which words were pre-labeled with a candidate labeling obtained from an automatic tagger (with a known error rate) as would occur in the context of AL. Linear regression on the study data yielded a model of POS annotation cost:

$$h = (3.795 \cdot l + 5.387 \cdot c + 12.57)/3600 \tag{6.1}$$

where h is the time in hours spent on the sentence, l is the number of tokens in the sentence, and c is the number of words in the sentence needing correction. For this model, the Relative Standard Error (RSE) is 89.5, and the adjusted correlation (R^2) is 0.181. This model reflects the abilities of the annotators in the study and may not be representative of annotators in other projects. However, the purpose of this paper is to create a framework for accounting for cost in AL algorithms. In contrast to the model presented by Ngai and Yarowsky [61], which predicts monetary cost given time spent, this model estimates time spent from characteristics of a sentence.

6.3 Evaluation Methodology and Results

Our test data consists of English prose from the POS-tagged Wall Street Journal text in the Penn Treebank (PTB) version 3. We use sections 2–21 as initially unannotated data. We employ section 24 as the development test set on which tag accuracy is computed at the end of every iteration of AL.

For tagging, we employ an order two Maximum Entropy Markov Model (MEMM). For decoding, we found that a beam of size five sped up the decoder with almost no degradation in accuracy from Viterbi. The features used in this work are typical for modern MEMM POS tagging and are mostly based on work by Toutanova and Manning [92].

In our implementation, QBU employs a single MEMM tagger. We approximate the entropy of the per-sentence tag sequences by summing over per-word entropy and have found that this approximation provides equivalent performance to the exact sequence entropy. We also consider another selection algorithm introduced in [73] that eliminates the overhead of entropy computations altogether by estimating per-sentence uncertainty with $1 - P(\hat{\underline{t}})$, where $\hat{\underline{t}}$ is the Viterbi (best) tag sequence. We label this scheme QBUOMM (OMM = "One Minus Max").

Our implementation of QBC employs a committee of three MEMM taggers to balance computational cost and diversity, following Tomanek et al. [91]. Each committee member's training set is a random bootstrap sample of the available annotated data, but is otherwise as described above for QBU. We follow Engelson and Dagan [27] in the implementation of vote entropy for sentence selection using these models.

When comparing the relative performance of AL algorithms, learning curves can be challenging to interpret. As curves proceed to the right, they can approach one another so closely that it may be difficult to see the advantage of one curve over another. For this reason, we introduce the "cost reduction curve". In such a curve, the accuracy is the independent variable. We then compute the percent reduction in cost (e.g., number of words or hours) over the cost of the random baseline

for the same accuracy a:

$$redux(a) = (cost_{rnd}(a) - cost(a))/cost_{rnd}(a)$$

Consequently, the random baseline represents the trajectory $redux(a) = 0.0$. Algorithms less costly than the baseline appear above the baseline. For a specific accuracy value on a learning curve, the corresponding value of the cost on the random baseline is estimated by interpolation between neighboring points on the baseline. Using hourly cost, Figure 6.2 shows the cost reduction curves of several AL algorithms, including those already considered in the learning curves of Figure 6.1 (except LS). Restricting the discussion to the random baseline, QBC, and QBU: for low accuracies, random selection is the cheapest according to hourly cost; QBU begins to be cost-effective at around 91%; and QBC begins to outperform the baseline and QBU around 80%.

6.4 Normalized Methods

One approach to convert existing AL algorithms into cost-conscious algorithms is to normalize the results of the original algorithm by the estimated cost. It should be somewhat obvious that many selection algorithms are inherently length-biased for sequence labeling tasks. For instance, since QBU is the sum of entropy over all words, longer sentences will tend to have higher uncertainty. The easiest solution is to normalize by sentence length, as has been done previously [27, 91]. This of course assumes that annotators are paid by the word, which may or may not be true. Nevertheless, this approach can be justified by the hourly cost model. Replacing the number of words needing correction, c, with the product of l (the sentence length) and the accuracy p of the model, equation 6.1 can be re-written as the estimate:

$$\hat{h} = ((3.795 + 5.387p) \cdot l + 12.57)/3600$$

Within a single iteration of AL, p is constant, so the cost is approximately proportional to the length of the sentence. Figure 6.2 shows that normalized AL algorithms (suffixed with "/N") generally

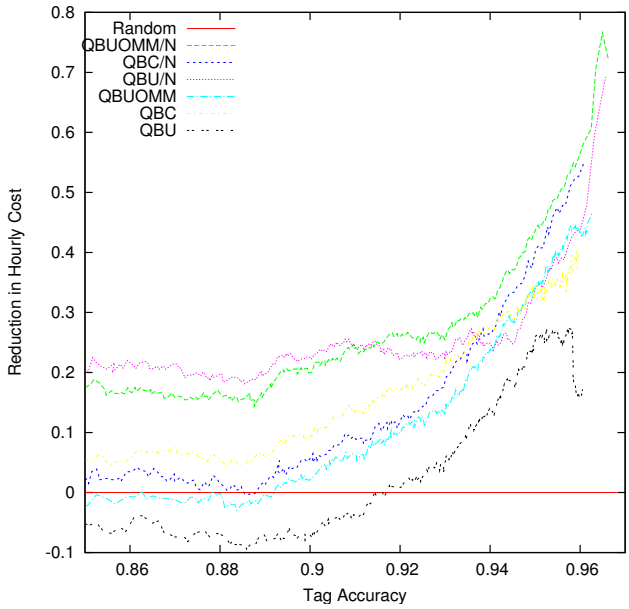


Figure 6.2: Cost reduction curves for QBU, QBC, QBUOMM, their normalized variants, and the random baseline on the basis of hourly cost

outperform the standard algorithms based on hourly cost (in contrast to the cost models used in Figures 6.1a - (c)). All algorithms shown have significant cost savings over the random baseline for accuracy levels above 92%. Furthermore, all algorithms except QBU depict trends of further increasing the advantage after 95%. According to the hourly cost model, QBUOMM/N has an advantage over all other algorithms for accuracies over 91%, achieving a significant 77% reduction in cost at 96.5% accuracy.

6.5 Conclusions

We have shown that annotation cost affects the assessment of AL algorithms used in POS annotation and advocate the use of a cost estimate that best estimates the true cost. For this reason, we employed an hourly cost model to evaluate AL algorithms for POS annotation. We have also introduced the cost reduction plot in order to assess the cost savings provided by AL. Furthermore, inspired by the notion of cost, we evaluated normalized variants of well-known AL algorithms and showed that these variants out-perform the standard versions with respect to the proposed hourly cost measure. In future work we will build better cost-conscious AL algorithms.

Chapter 7

Return on Investment for Active Learning[†]

Abstract

Active Learning (AL) can be defined as a selectively supervised learning protocol intended to present those data to an oracle for labeling which will be most enlightening for machine learning. While AL traditionally accounts for the value of the information obtained, it often ignores the cost of obtaining the information thus causing it to perform sub-optimally with respect to total cost. We present a framework for AL that accounts for this cost and discuss optimality and tractability in this framework. Using this framework we motivate Return On Investment (ROI), a practical, cost-sensitive heuristic that can be used to convert existing algorithms into cost-conscious active learners. We demonstrate the validity of ROI in a simulated AL part-of-speech tagging task on the Penn Treebank in which ROI achieves as high as a 73% reduction in hourly cost over random selection.

7.1 Introduction

Labeled data is a prerequisite for many algorithms in Natural Language Processing (NLP) and machine learning. While large amounts of annotated data are available for well-studied languages in well-studied domains and for well-studied problems such as part-of-speech (POS) tagging, this is not true for less common languages or domains. Unfortunately, obtaining human annotations for linguistic data is labor intensive and typically the costliest part of the acquisition of an annotated

[†]Robbie A. Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. Return on investment for active learning. In *Proceedings of the Neural Information Processing Systems Workshop on Cost Sensitive Learning*, 2008.

corpus; a great deal of funding continues to be devoted to the annotation of data. Hence, there should be substantial interest in reducing annotation costs while preserving quality.

Active Learning (AL) can be employed to reduce the costs of corpus annotation [27, 73, 91]. The primary responsibility of the active learner has been to choose items for which it believes the true annotations will provide the most benefit. However, cost must also be considered [8, 34], and AL should deliver annotations that balance cost and utility.

In this paper, we present AL as an order optimization and decision problem and discuss the optimal approach to AL based on decision theory. We motivate the need for this approach to be based on the "net utility" (NU), which incorporates both the utility and cost of annotating an entire sequence of items. However, due to the intractability of the optimal approach, greedy approaches and heuristics must be used. We present a practical, novel heuristic that combines utility and cost in a measure called Return On Investment (ROI). We apply this cost-sensitive perspective and demonstrate the advantage of this technique over traditional AL methods in a simulated AL part-of-speech tagging task on the Penn Treebank.

The rest of the paper will proceed as follows: Section 7.2 discusses the role of both benefit and cost in AL. Section 7.3 provides a framework for AL which we use to discuss previous work and motivate our approach, ROI, which is subsequently introduced in Section 7.4. We explain our methodology for conducting experiments in Section 7.5, and results are presented in Section 7.6. Finally, Section 7.7 discusses conclusions and indicates our plans for future work.

7.2 The Role of Cost and Benefit

Pivotal to AL is the observation that not all data are created equal: some data are inherently more beneficial than others. Key to this work is the fact that data also differ in how costly they are to annotate. Although this is true for nearly all annotation tasks, it is particularly obvious when annotating sequences or other structured objects, which are common in NLP and bioinformatics, among other fields. Consider for example the manual POS tagging task. All else being equal, it will clearly take more time, and hence cost more, to annotate longer sentences. On the other hand,

sentences containing frequent words or features that provide discriminatory information tend to be more valuable. An active learner should seek sentences that are most beneficial and least costly.

In order for an active learner to determine which data are "most beneficial" and "least costly," benefit (utility) and cost must be clearly defined. The exact determination of cost and utility are typically project-dependent and sometimes difficult to measure. Nevertheless, the performance of an AL algorithm is ultimately determined by both the true cost and utility, and hence, any attempt to characterize the performance of the algorithm should measure both aspects as closely as possible.

There are many facets to utility, but in this work we focus on measures of model goodness, in particular, accuracy. However, this may not capture all aspects of true utility. For example, Baldridge and Osborne [6] have suggested that reusability of the annotated data can be important. Similarly, the cost of an AL project depends on many variable and fixed costs [61], but how the annotators will be paid is one of the most significant factors. The usual assumption for AL is unit cost per label, which is unrealistic particularly for tasks involving labeling sequences or other structured data. Becker et al. [8] and Haertel et al. [34] show how the relative performance of different algorithms is determined in part by how cost is measured. Many of the costs employed in other work (e.g., Hwa [43], Osborne and Baldridge [63]) can be seen as estimating some portion of the hourly cost. The nature of the annotation user interface, the number and efficiency of annotators, etc. are additional components of cost [14]. There is also a human learning curve that inevitably impacts the cost of an AL system.

7.3 Background and Decision Theoretic Framework for Active Learning

Most work in AL assumes that a single unerring oracle provides annotations (however, this framework can also be applied to cases when the annotator is fallible or there is more than one annotator). AL is an order optimization problem [18]: a rational organization that desires to annotate a particular set of initially unannotated items $\mathscr{U}$, will seek the totally ordered subset $\mathscr{A}^* \subseteq \mathscr{U}$ such that $\mathscr{A}^* = \arg\max_{\mathscr{A} \subseteq \mathscr{U}} \mathrm{NU}(\mathscr{A})$ where $\mathrm{NU}(\mathscr{A}) = \mathrm{utility}(\mathscr{A}) - \mathrm{cost}(\mathscr{A})$, and "utility" and "cost"

(discussed in the previous section) are functions of an ordered set that must be on the same scale. Furthermore, no rational organization would undertake a project if $\mathrm{NU}(\mathscr{A}^*) < 0$.

Decision theory provides a statistically optimal approach to finding $\mathscr{A}^*$ with respect to the utility and cost functions. The test selection problem consists of using the Expected Value of Sampling Information (EVSI) [69] to choose which "test," if any, could be performed that would maximize the expected net utility (ENU). In AL, a "test" consists of requesting an annotation from the oracle. Since each query to the oracle potentially affects the NU of future queries, it is necessary to consider the NU of full sequences of tests/queries. Let $\mathscr{A}$ be the set of data with annotations and $\mathscr{U}$ be the unannotated data. The optimal choice of item for annotation is:

$$x^* = \arg\max_{x \in \mathscr{U}} \mathbb{E}_{Y=y|x,\mathscr{A}} \left[R\left(\mathscr{A} \cup \{(x,y)\}, \mathscr{U} - x\right) \right] \tag{7.1}$$

where Y is a r.v. representing the annotation for instance x and R is the maximum ENU:

$$R(\mathscr{A}, \mathscr{U}) = \max \left(\max_{x \in \mathscr{U}} \mathbb{E}_{Y=y|x,\mathscr{A}} \left[R\left(\mathscr{A} \cup \{(x,y)\}, \mathscr{U} - x\right) \right], NU\left(\mathscr{A}\right) \right)$$

which is computed by recursively choosing the next test instance that maximizes the ENU given the updated $\mathscr{A}$. R is at a maximum when future iterations are no longer expected to increase the value of R or there is no data left to consider (in other words, $R(\mathscr{A}, \emptyset) = NU(\mathscr{A})$).

The optimal AL algorithm therefore is to choose an item x^* according to equation 7.1, query the oracle for the annotation y, add the annotated pair (x,y) to $\mathscr{A}$, and repeat until R is not greater than the current NU. Note that although some have recognized their approach as greedy (e.g., Schohn and Cohn [77]), Carroll et al. [15] and this work are the first to explicate the fully optimal approach, as far as we are aware. Although computing expectations for every possible permutation of unannotated items (performed by the recursive function R) at each stage of AL is clearly intractable, knowing the optimal approach has guided the current work and should help direct future work towards better approximations.

Several authors have used a greedy version of EVSI-based AL in which no recursion is necessary (e.g., Anderson and Moore [2], Cohn et al. [17], Kapoor et al. [46], Margineantu [58], Roy and McCallum [75], Schohn and Cohn [77]). Even this greedy approach can be expensive, since posterior probabilities must be computed for every possible labeling of every possible unannotated item. Some gains in efficiency can be had using models that allow for efficient (albeit often approximate) computation of posterior probabilities (e.g., Cohn et al. [17], Kapoor et al. [46]), and it is also possible to consider a subset of possible tests and use sampling to compute ENU [75].

Even with such simplifications, the greedy approximation can be computationally costly for many problems and models. Rather than computing posterior probabilities, heuristic methods may be used which typically require training a manageable number of models on the previously annotated data and using these models to rank the remaining items according to some criterion. Most heuristics attempt to score instances in such a way that the scores are proportional to the expected change in utility. The most common heuristics assume that this change is proportional to some type of uncertainty. Let Y be the random variable representing a label assignment to the test instance x. Then we refer to the uncertainty of the distribution of $Y|x$ as *ambiguity*. However, since the true distribution of $Y|x$ is unknown, it is possible for the mode of any estimate to be incorrect, leading to an incorrect classification. The distribution over possible modes of $p(Y|x)$, i.e., $p\left(Z = \arg\max_y p(Y=y|x)\right)$ (c.f. Dagan and Engelson [19]), represents a type of uncertainty we call *correctness*.

Ambiguity and *correctness* can be quantified in several ways, including entropy. Although entropy takes into account the mass or density across the entire support, the most important question is whether or not the model is correct. A more direct measure is to use the probability that the model is wrong, i.e., $1 - p(mode)$ (One-Minus-Max; OMM) (c.f. Ringger et al. [73]). Another interesting measure of uncertainty when using committee-based approaches is K-L divergence from the mean [59]. This method measures the average distance (computed using K-L divergence) of the distribution of each member of a committee to the mean distribution of the committee. This

has the benefit that truly ambiguous distributions usually have low variance and hence are avoided. Anderson and Moore [2] prove that using entropy and K-L divergence are in fact the same.

Approaches in which a single model assesses *ambiguity* are referred to as uncertainty sampling [87] (also Query-By-Uncertainty; QBU). Monte Carlo techniques can be used to assess both *ambiguity* and *correctness*; approaches for the latter are known as Query-By-Committee (QBC) [27, 82].

7.4 Return on Investment

Optimizing ENU directly is difficult, even greedily. First, computing posterior probabilities and therefore ENU can often be prohibitively costly. Assuming annotators are paid while waiting for the active learner, long computation *should* result in lower NU. Lamentably, this has yet to be properly accounted for. A second challenge to using NU is that, in practice, it tends to be difficult to define cost and utility in the same units. Anderson and Moore [2] recognize that this is particularly problematic for entropy-based utility functions. It is also especially true when it is not possible or feasible to calculate the maximum EU of the unannotated data (the most prevalent utility function). The fact that the majority of work on AL assesses performance using graphs of cost vs. utility rather than number of queries vs. NU attests to this difficulty. Most AL work assumes constant cost (often unjustifiably) in which case the graphs are scaled variants of each other. Even those approaches that use more justifiable cost measures frequently measure performance on the basis of cost vs. utility (e.g., Engelson and Dagan [27], Haertel et al. [34], Tomanek et al. [91]).

In the absence of a good conversion between units of cost and utility, the use of plots of cost vs. utility is justified since they still encapsulate useful information. For instance, AL algorithms that produce lines having greater area under them tend to correspond to algorithms with higher maximum NUs. Furthermore, some algorithms tend to exhibit higher utility for nearly all levels of cost when compared to other algorithms, which translates to higher NU in the same regions.

For this reason, it is desirable to employ algorithms that select items that maximize utility per unit of cost, specifically in the case that it is not practical or even possible to compute ENU.

A straightforward method of maximizing this quantity is through the use of Return on Investment (ROI). ROI is defined as: $\text{ROI}(x) = \frac{\text{utility}^*(x) - \text{cost}^*(x)}{\text{cost}^*(x)} = \frac{\text{utility}^*(x)}{\text{cost}^*(x)} - 1$. When using ROI as a selection metric, the learner ranks items by the ratio of an estimate of the utility of annotating each item to an estimate of the cost to annotate it; the item that maximizes this quantity is selected for annotation. Note that these estimates can be functions of any form, including non-linear and unbounded functions. Conveniently, it is not necessary to compute the non-trivial conversion ratio required by NU, yet, unlike the traditional heuristics such as QBU and QBC, ROI is clearly cost-sensitive. As desired, ROI (greedily) maximizes utility per unit of cost. ROI selects items whose *estimated* slope on a learning curve of cost vs. utility are at a maximum given the previous points, provided that the cost and utility estimators used in ROI are the same (or similar) to those measured in the graph. This generally leads to learning curves that tend toward the upper-left corner of the graph (lower cost and higher utility), which is precisely what constitutes good AL algorithms.

7.5 Methodology

In the remainder of this work, we consider several AL selection algorithms that leverage the idea of ROI for the POS tagging task. As previously defined, these algorithms consist of two parts: a utility estimator and a cost estimator. By combining different cost and utility estimators in the ROI model introduced above, we create novel selection algorithms and then evaluate their performance.

7.5.1 Utility Estimation

Computation of uncertainty proceeds differently for *ambiguity* and *correctness*. In order to compute the *ambiguity* of a sentence by way of entropy, we approximate the per-sentence tag sequence entropy by summing over an estimate of the conditional per-word tag entropy where the previous tags are taken from the Viterbi-best sequence of tags [34]. For OMM, it is sufficient to subtract the probability of the Viterbi-best tag sequence from one.

Similar to Engelson and Dagan [27], for *correctness*, each member of a committee produces the most probable tag sequence for a sentence and a vote histogram is created for each word; each

member votes once for the tag it chose. The same measures of uncertainty can be computed for this vote distribution as above.

7.5.2 Cost Estimation

There are three cost estimates that we can use in the absence of the truth during the annotation process. First, we can assume constant cost per sentence. Second, we can assume that cost is proportional to the number of words in the sentence. Engelson and Dagan [27] as well as Tomanek et al. [91] do the latter and thereby implicitly assume the "length" cost function without explicitly acknowledging the question of cost. Lastly, Haertel et al. [34] argue that the most important cost is hourly cost. For the purposes of the experiments reported in this work, we estimate the length of time required to annotate any given sentence with POS tags using the "hourly cost model" of time required for annotation that Ringger et al. [73] developed based on a user study:

$$h = (3.80 \cdot l + 5.39 \cdot c + 12.57)/3600 \tag{7.2}$$

where h is the time in hours spent on the sentence, l is the number of tokens in the sentence, and c is the number of tags in the sentence that need correction. The expected value of this cost function can easily be estimated in ROI. Since l is known for each sentence, we need only estimate $\hat{c}$, the expected number of tags to be changed in a sentence $\underline{w}$ with the most-likely tags $\underline{t}$ selected by the tagger. In our model (described in the following section), this is $\hat{c} = \sum_{i=1}^{|\underline{w}|} [1 - p(t_i|\underline{w}, t_{i-2}, t_{i-1}, \boldsymbol{\phi}(\underline{w}, i, t_{i-2}, t_{i-1}))]$.

7.5.3 Experimental Setup

We employ an order-two Maximum Entropy Markov Model (MEMM) for all models. The features used in this work are mostly based on work by Toutanova and Manning [92]; the tagger is near the state-of-the-art, achieving 96.90% in tag accuracy once AL reaches completion.

Our data consists of English prose from the POS-tagged Wall Street Journal text in the Penn Treebank (PTB) version 3. Sections 2-21 serve as initially unannotated data and section 24 is the

set on which tag accuracy is computed at the end of every round of AL. By using the true labels and comparing them to the output of a tagger trained on the sentences annotated up through the end of the current round, it was possible to use equation 7.2 to estimate the amount of time that would have been required to annotate each sentence. Ideally, we would evaluate the effectiveness of the various algorithms by measuring the actual time of human annotators. Unfortunately, this would require that we pay annotators to tag an entire data set many times to achieve a statistically significant comparison of the algorithms. Clearly this is not feasible, although Hachey et al. [33] do this (once) for three algorithms on a set of 100 sentences. Our experiments show that the largest cost reductions over baseline come long after the first 100 sentences. In a real AL environment, it would be possible to estimate annotation cost during the annotation process, an idea we wish to pursue in future work.

In order to minimize the amount of time that the human waits for the computer to retrain the model and choose the next sentence to annotate, it is sometimes desirable to process sentences in batches, even though this can slightly decrease the efficiency of AL. Sentence selection and model training is relatively cheap in the early stages of AL and accuracy rises very quickly, whereas the opposite is true at the later stages. Based on initial tests, we found that a batch size that increases exponentially as a function of the number of previously annotated sentences had little perceivable effects on the accuracy achieved for any (human) hourly cost. In a real annotation task, batch size can be self-determined by the amount of time it takes the algorithm to select a sentence in the previous round. Following Engelson and Dagan [27], rather than scoring every unannotated sentence, we instead score only a sample of the sentences. We found that maintaining a candidate set size 500 times the size of the batch worked well. For committee-based approaches, the k committee members were trained using separate bootstrap samples from the available annotated data. Following Tomanek et al. [91], we use a committee of size three as a reasonable balance between computational burden and model diversity. Results are averaged over a minimum of 15 runs from different random initial sentences.

			Cost Estimator		
	Distr. Type	**Uncertainty Calc.**	**Constant**	**Num Words**	**Exp. Hourly Cost**
Utility Est.	ambiguity	entropy	QBUE*	QBUE/N*	QBUE/EHC
		one-minus-max	QBUOMM	QBUOMM/N	QBUOMM/EHC
	correctness	entropy	QBCE	QBCE/N	QBCE/EHC
		one-minus-max	QBCOMM*	QBCOMM/N*	QBCOMM/EHC
		K-L divergence	QBCKL*	QBCKL/N*	QBCKL/EHC

Table 7.1: Naming conventions for the various combinations of utility and cost estimators used in ROI for the experiments; asterisks indicate combinations not investigated in this work.

To assess the performance of each algorithm, we follow Haertel et al. [34] in their use of "cost reduction" plots. In these plots, the accuracy (utility) is the independent variable. The percent reduction in cost over the cost of the random baseline for the same accuracy a is then computed as $r(a) = (cost_{rnd}(a) - cost(a))/cost_{rnd}(a)$. Consequently, the random baseline has no reduction compared to itself and represents the trajectory $r = 0.0$. For a specific (cost, accuracy) point on a learning curve, the corresponding value of the cost on the baseline learning curve is estimated by linear interpolation between neighboring points on the baseline. In these graphs, a higher reduction represents a greater advantage.

7.6 Results

To show the efficacy of the ROI approach, we separately investigate the effects of different cost and utility estimators. The naming conventions for the various combinations are summarized in Table 7.1.

7.6.1 Cost Estimators

In Figure 7.1, we plot the reduction of hourly cost using each of the three different cost estimates for the ROI algorithm (constant per-sentence cost, length of sentence, and estimated hourly cost) and two different utility estimators (QBCE and QBCOMM). Each of these algorithms is evaluated on the basis of two different costs: pay-by-the-word, and pay-by-the-hour. We chose to use QBCE since it is the one employed by Engelson and Dagan [27] for POS tagging (the actual algorithm is

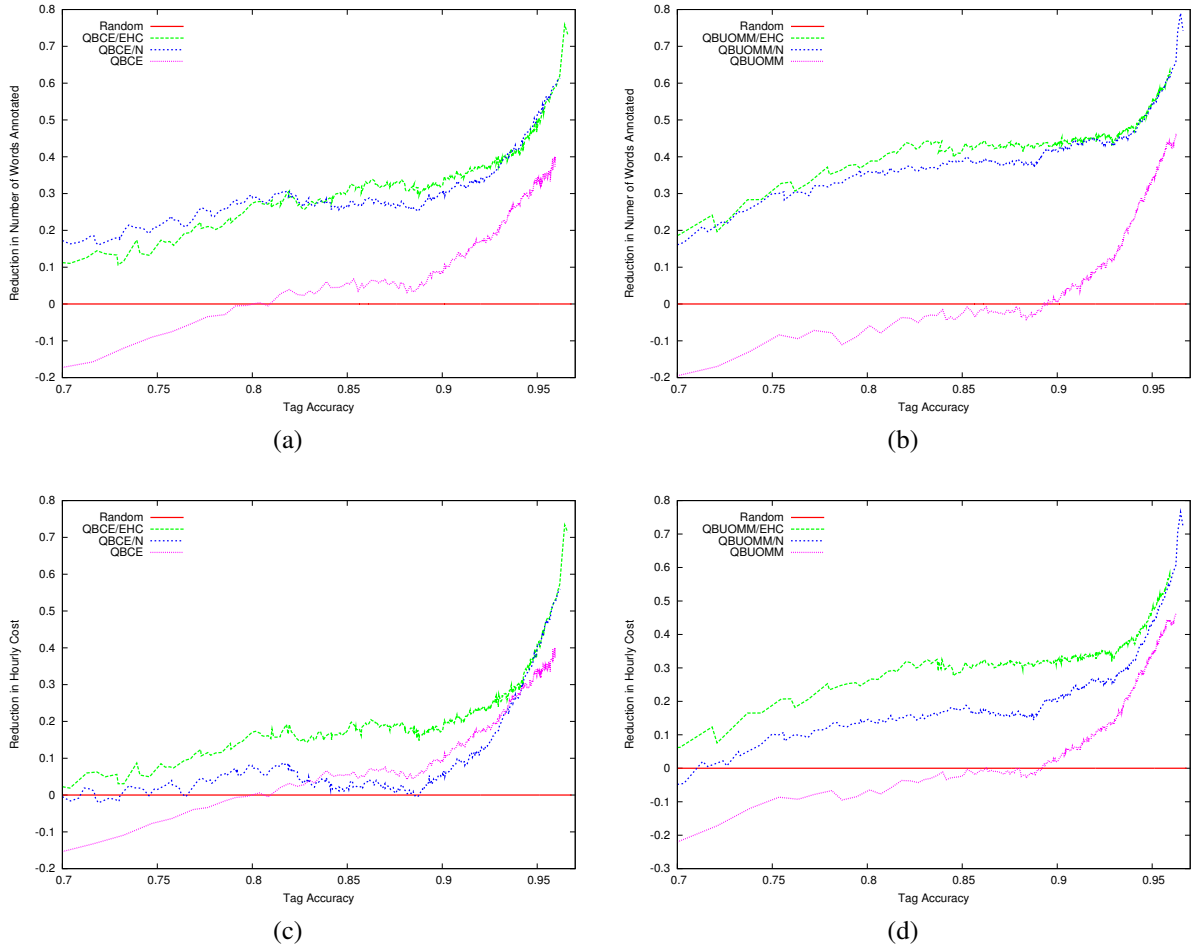


Figure 7.1: Comparison of three cost measures (EHC, N, and constant) for QBCE (a,c) and QBUOMM (b,d) when paying annotators per tagged word (a,b) or per hour (c,d).

our QBCE/N variant). Haertel et al. [34] have previously shown QBCOMM/N to be superior to QBCE/N when paying annotators by the hour on this task for this dataset.

The cost-conscious algorithms (i.e., the N- and EHC-variants) outperform those that assume constant cost. The similarity of these variants is due in part to the role of sentence length in the hourly cost model and the fact that after only a few dozen iterations, the tagger has reached high enough accuracy that the number of words needing correction is low. Interestingly, when paying annotators per word annotated (Figures 7.1a-b), the variants that use estimated hourly cost tend to (barely) out-perform those that directly estimate the per-word-cost. We note that the models are error-prone in the early stages of AL. We hypothesize that since the same error-prone model is

used in the computation of both the numerator and the denominator in the EHC-variants, errors effectively "cancel out" whereas for the N-variants, the model is used only in the computation of the numerator.

When evaluating the performance of the algorithms based on paying annotators by the hour (Figures 7.1c-d), QBCE/EHC reaches an advantage over baseline of around 73%; even greater advantages are reached by the cost-conscious QBUOMM variants. Importantly, the EHC-variants enjoy a greater reduction than the other cost estimators at all accuracy levels, providing empirical evidence that ROI typically maximizes utility per cost when the estimators match those being measured. However, the advantage of the EHC-variants compared to the N-variants diminishes at high levels of accuracy because EHC is approximately proportional to the length of the sentence for high levels of accuracy. Both variants exhibit a growing advantage over the non-cost-conscious QBCE and QBUOMM at these higher levels of accuracy. Interestingly, even these constant cost estimators have a clear advantage over the baseline (nearly a 40% reduction in cost for QBCE and 46% for QBUOMM). Finally, QBCE/N has little advantage over baseline when paying annotators by the hour until it reaches around 89% accuracy (Figure 7.1c).

7.6.2 Utility Estimators

Since the EHC estimate of cost outperforms the others, we fix this estimate of cost to evaluate the different utility (accuracy) estimators. Figure 7.2 depicts the relative advantage of five accuracy estimators: two single model *ambiguity*-based estimates, QBUE/EHC and QBUOMM/EHC; two committee-based *correctness* estimates, QBCE/EHC and QBCOMM/EHC; and KL-divergence to the mean, QBCKL/EHC.

All algorithms enjoy a relatively large reduction in cost over baseline, with QBCE/EHC and QBCKL/EHC achieving nearly a 73% reduction in cost. We were unable to collect as much data for QBUOMM/EHC, but it would appear to enjoy an even greater advantage. The OMM approach outperforms the entropy-based method for *ambiguity* estimators, although no such advantage exists

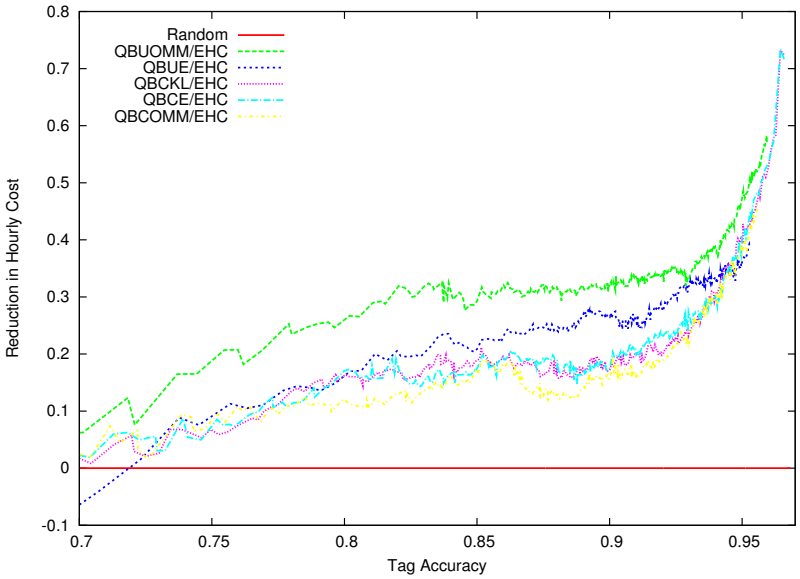


Figure 7.2: Comparison of utility estimators.

for *correctness* estimators. All of the committee-based variants performed nearly equally well. We suspect that one limiting factor is the committee size and possibly our use of bootstrap sampling.

We were somewhat surprised to see that the *ambiguity*-based estimates of accuracy outperformed the correctness-based estimates. The *ambiguity*-based estimates are known to struggle with cases that are truly ambiguous and hence *correctness*-based approaches were thought to perform better in this context. Small committee size could be a factor, but perhaps entire tag sequences are less ambiguous than individual tags, due to potentially disambiguating interactions between words.

7.7 Conclusions and Future Work

We began this paper by discussing the role and importance of utility and cost in AL. By explicating a framework for AL, we hope that better cost-conscious approximations can be built. To this end we introduced a new heuristic, ROI. One advantage to ROI is that any existing algorithm can become cost-conscious by simply normalizing existing utility estimators by an estimate of cost.

In our experiments, we focused on POS tagging of the Penn Treebank, comparing the accuracy of different combinations of cost and utility in ROI. We showed that, not surprisingly, one should usually employ that estimate of cost in the ROI algorithm which matches the one to be

measured. We also found that algorithms based on uncertainty due to ambiguity worked better than the correctness-based algorithms for POS tagging on the Penn Treebank.

To the best of our knowledge, whereas some prior work (e.g., Kapoor et al. [46]) acknowledges query cost and other prior work (e.g., Engelson and Dagan [27]) estimates query cost in an implicit fashion, this work is the first to present results of AL algorithms that explicitly incorporate a realistic query cost based on a predictive model of human annotation time. The ROI framework also generalizes those earlier efforts. Furthermore, our results suggest that, despite its simplicity, ROI is a valuable first step towards developing cost-conscious AL algorithms. We note that although ROI is a simple concept, the difficult part consists in developing appropriate estimates of cost and utility, and every project will need to develop their own estimates. Nevertheless, this work suggests that doing so is worthwhile since ROI can improve the efficiency of AL, making effective cost-conscious AL feasible.

It should be possible to allow the human to annotate while the computer selects a sentence. In future work, we intend to explore how to take better advantage of overlapping the work of the human and the computer to reduce overall cost. Finally, although our experiments dealt with POS tagging on the Penn Treebank, we have explained why ROI performs well and hypothesize that ROI performs equally well on many other AL tasks, especially other sequence-labeling problems. Testing this hypothesis is also the subject of future work.

7.8 Errata

The published version of this chapter states that we use one-minus-max (OMM; defined in Section 7.3) as a benefit estimator. We have since found that our implementation was actually computing $-\max_{\mathbf{t}} \log p(\mathbf{t}|\mathbf{w})$, which we term NLMP in Chapter 8; we use the same estimator in Chapter 9. Note that, although OMM and NLMP produce scores that would rank instances the same based solely on this estimate of benefit, in the context of ROI, the division by cost alters the rankings since the two benefit estimators are not linearly related. This issue is touched upon further in Chapter 8.

In addition, the published version of this paper contains an incorrect date for the publication by Raiffa and Schlaifer [69] which has been updated for this version.

Chapter 8

An Analytic and Empirical Evaluation of Return-on-Investment-Based Active Learning

Abstract

Return-on-Investment (ROI) is a cost-conscious approach to active learning (AL) that considers both estimates of cost and of benefit in active sample selection. In this chapter, we investigate the conditions for successful cost-conscious AL using ROI by proving the conditions under which ROI would optimize the area under the cost/benefit curve. We then empirically measure the degree to which optimality is jeopardized in practice when the conditions are violated. We find that the more linearly related a benefit estimator is to true benefit, the better it performs when paired with an imperfect cost estimate in ROI. Lastly, we use our analysis to explain the mixed results of previous work. Our results show that ROI can indeed successfully reduce total annotation costs.

8.1 Introduction

In active learning (AL), sample selection algorithms sequentially choose instances, or "samples," to be labeled/annotated by an oracle; each annotated instance results in a measurable benefit (e.g., increase in model accuracy) and incurs a specific cost (e.g., time needed to obtain the label). Unfortunately, until recently, AL research has ignored the fact that instances have varying costs. Recent decision-theoretic approaches (e.g., Liang et al. [52]) can incorporate per-instance cost, but typically ignore it during experimentation, due in part to the difficulty of subtracting cost from benefit when they are measured in different units [24, 36]. Return-on-investment (ROI) is a cost-conscious technique that avoids this requirement by selecting the instance x^* having maximum

net benefit per unit cost, i.e.,

$$x^* = \arg\max_x \frac{benefit(x) - cost(x)}{cost(x)}$$
$$= \arg\max_x \frac{benefit(x)}{cost(x)}. \quad (8.1)$$

This approach to AL was independently proposed by Donmez and Carbonell [24], Haertel et al. [36], and Settles et al. [81]; in addition, Tomanek and Hahn [89] evaluated the effectiveness of ROI. Unfortunately, the results are mixed. In addition, despite its intuitive appeal and recent attention as a practical cost-conscious algorithm, there has been little theoretical justification for ROI in the context of AL.

The purpose of this chapter is to provide an initial theoretical analysis of ROI that allows us to explain its behavior in a practical environment. We also empirically assess the degree to which violated conditions affect the overall performance of ROI.

The chapter is organized as follows: related work is presented in Section 2. Section 3 examines the conditions under which ROI would be optimal. Section 4 discusses the experimental methodology. Section 5 experimentally assesses the extent to which the conditions hold in practice, but outside the context of AL while Section 5 explores the overall effect on AL. Finally, Section 6 presents our conclusions and future work.

8.2 Related Work

The very essence of active learning is to select the next "best" instance to be annotated. This, of course, raises the question as to which selection function is optimal. Cohn et al. [17] derive a solution for selecting the instance that minimizes model variance. A related class of solutions based on optimal experimental design use Fisher information to select the optimal instance [98]. However, these approaches fail to account for problems in which instances are not equally costly to annotate.

Decision theory offers an elegant framework for (greedily) selecting the next best instance based on utility and which can handle variable query costs; some examples include Liang et al. [52],

Anderson and Moore [2], Margineantu [58], and Kapoor et al. [46]. In this framework, the optimal instance is the one which has maximum net utility, i.e., utility less cost. However, this approach requires that net utility and cost be in the same units. This requirement is particularly problematic when heuristics (such as entropy) are used to approximate expected utility.

Another approach, borrowed from the financial industry, is return-on-investment (ROI) [24, 36, 81]. ROI is related to the decision theoretic approach [36]; however, unlike the decision theoretic approach, ROI does not require conversion between units of utility (benefit) and cost.

ROI has explicitly been employed with mixed results on a variety of tasks. Donmez and Carbonell [24] show positive results with ROI on face detection, letter recognition, spam detection, and high revenue detection tasks, but do not evaluate ROI using variable instance costs. Settles et al. [81] evaluate ROI on entity-relation tagging, speculative text classification, and information extraction. They limit themselves to an N-best approximation to entropy for the sequence labeling tasks, but ROI does not outperform basic AL. Haertel et al. [36] show positive performance of ROI on English part-of-speech tagging. Finally, Tomanek and Hahn [89] find that ROI slightly outperforms two new cost-conscious algorithms when an appropriate benefit function is used.

In short, ROI is a promising approach to cost-conscious AL that can account for variable instance cost without requiring cost and benefit to be in the same units. However, unlike optimal experiment design and decision theory, the theoretical underpinnings of ROI for AL have hitherto remained largely unexplored and the mixed results require reconciliation.

8.3 Theoretical Analysis of ROI

The fact that ROI was successful in some previous work indicates that it is doing something right. The purpose of this section is to investigate those reasons by providing a bottom-up theoretical justification for ROI. That is, this section proposes Area Under the cost/benefit Curve (AUC) as a suitable objective function and then enumerates a set of conditions that, if true, would lead to ROI maximizing AUC. As a result of the bottom-up derivation, the assumptions made are somewhat

strong, but we dedicate the next two sections to analyzing the degree to which they hold in practice and their effect on results in practice.

We begin with a brief set of definitions. AL algorithms sequentially select instances from a set of unlabeled instances $\mathscr{U}$ ("the pool"). As an instance $x \in \mathscr{U}$ is annotated with label y, it results in a measurable benefit and also incurs a specific cost.[1] In general, the benefit and cost of obtaining a particular annotation may depend on previously obtained annotations. Thus, we define *total benefit* and *cumulative cost* to be functions ($b(\cdot)$ and $c(\cdot)$, respectively) of a sequence of labeled data $L = \langle (x_1, y_1), \ldots, (x_n, y_n) \rangle$. For simplicity, we assume that the cost to annotate an instance is independent of its order, although it can be shown that this assumption has no bearing on the final analysis. Therefore, $c(L_{1\ldots i}) = \sum_{i'} c(L_{i'})$.

There are multiple metrics for comparing the performance of AL algorithms. In a real annotation project, the only metric that matters is the final benefit that is achieved. However, the algorithm must be selected *before* annotation begins and the algorithm is chosen based on the outcomes of prior experiments under similar circumstances. Due to the inherent uncertainty surrounding the applicability of the results to the new annotation project, selecting an algorithm based on the difference in benefit at any one specific cost (e.g., the budget available for the new project) is dangerous. Instead, the trends over sufficiently large ranges of cost should be considered.

This information is conveyed in a cost/benefit curve (a generalization of standard learning curves) which parametrically plots $b(L_{1\ldots i})$ against $c(L_{1\ldots i})$ for $i \in \{1, \ldots, |L|\}$. All previous work of which we are aware evaluate AL using cost/benefit curves or derivations thereof. In general, curves that tend to the upper-left corner of the graph are preferable, even when they end at the same point as other curves. This notion of preference can be formalized into an objective function in which AUC is to be maximized. Note that Settles and Craven [80] and Baldridge and Osborne [6] use AUC to evaluate AL algorithms.

We now formally define AUC. Assuming linear interpolation between discrete neighboring points, AUC is the sum of the area of the right trapezoids defined by adjacent points on the curve.

[1] For the purposes of this section, we follow previous work in assuming a single annotator.

Let $a_i(L)$ be the area of the i^{th} trapezoid:

$$a_i(L) = \frac{1}{2}\left[c\left(L_{1\ldots i}\right) - c\left(L_{1\ldots i-1}\right)\right] \cdot \left[b\left(L_{1\ldots i-1}\right) + b\left(L_{1\ldots i}\right)\right] \tag{8.2}$$

(where $c(\emptyset) = b(\emptyset) = 0$). Then, the AUC defined by the sequence L is:

$$auc(L) = \sum_{i=1}^{|L|} a_i(L). \tag{8.3}$$

Maximizing AUC using AL can be seen as a sequential decision problem in which each decision consists of selecting an instance for annotation. The optimal instance to select given previous annotations L, will depend on the decision's effect on the next decision, and the effect of the second decision on the third, and so forth, until all decisions have been made. To account for this recursive dependence, we must consider entire sequences of decisions. Note that if we do not allow instances to be selected more than once from $\mathscr{U}$ and we will eventually choose every instance, then the number of decisions per sequence is $N = |\mathscr{U}|$. Since the actual annotations that the oracle will provide is unknown, they must be considered in expectation represented with random variables Y_i. Given a sequence of already annotated data L, one approach to maximizing AUC in expectation that accounts for this recursive effect of decisions is:[2]

$$x_1^*, \ldots, x_N^* = \\ \underset{x_1,\ldots,x_N}{\arg\max}\, \mathbb{E}_{Y_1,\ldots,Y_N|x_1,\ldots,x_N,L}\left[auc\left(L \oplus \langle (x_1,y_1),\ldots,(x_N,y_N)\rangle\right)\right], \tag{8.4}$$

where $\oplus$ represents sequence concatenation. Although finding the optimal sequence accounts for the effects each decision has on successive decisions, in fact, the sequential decision process protocol requires only the first instance in this sequence, viz., x_1^*. We then append x_1^* and the oracle's annotation y_1 for the instance to L. The result is an updated belief reflected in the expectations (via the new L) used to select the next instance.

[2]Recall that this derivation is bottom-up; see Haertel et al. [36] for a decision-theoretic variant.

We now derive ROI from equation 8.4 under the following conditions:

1. The covariance of cost and benefit is zero.
2. The cost and benefit of each instance are independent of the order in which instances are annotated.
3. Each r.v. Y_i (i.e., label) is conditionally independent of all other $Y_{j\neq i}$, given x_i and L.
4. Cost and benefit are exactly computed up to a scalar constant.

The reader is reminded that these conditions are derived in a bottom-up fashion and that we do not necessarily presume them to hold in practice; we briefly overview their practicality herein and examine the degree to which they hold in our experiments. While it is conceivable that some annotation problems have zero covariance between cost and benefit, there are certainly cases where there may be some correlation. This correlation is especially evident in structured learning problems, e.g., "larger" instances (e.g., long sentences) will tend to contain more information but be more costly. However, to our knowledge, the amount of correlation in such instances has not been studied previously; we provide some data in the next section. Although cost may be independent of annotation order (as implicitly assumed by previous work, e.g., Settles et al. [81]), the benefit of an instance will, in fact, usually depend on the order in which it is annotated. For example, instances annotated earlier will lead to larger increases in accuracy than if the same instances were annotated later. In fact, if both cost and benefit are independent of the order in which they are annotated, active learning would be unnecessary. Nevertheless, traditional active learning algorithms can be seen as making this very assumption. Active learning presumably works because this assumption is approximately true for a single decision. Since beliefs are updated after each new annotation, it continues to be approximately true each decision. Condition 3 is similar to the assumption that benefit is independent of the order in which instances are annotated, but applies distributionally and is more mathematically precise. Finally, optimal (exact) benefit estimators are computationally intractable. While some approaches are optimal for the last decision and perform very well (e.g., Roy and McCallum [75]), these approaches are impractical for structured prediction tasks; we will

examine the effectiveness of several benefit estimators in our empirical examination. Similarly, although cost is sometimes knowable *a priori* it often is not. However, Settles et al. [81] showed that cost can often be reliably learned in practice.

While we defer the question of the degree to which these assumptions are violated in practice to our experiments, we proceed with the analysis as if they held true to better understand the theoretical underpinnings of ROI. In the context of maximization, the scalar constants allowed in condition 4 can be ignored. The linearity property of expectations allows us to move the expectation in equation 8.4 inside of the sum in equation 8.3. The first condition then allows us to move the expectation further into the area calculation so that equation 8.2 becomes (omitting expectation indices for brevity):

$$a_i(L) = \frac{1}{2}\left(\mathbb{E}\left[c\left(L_{1\ldots i}\right)\right] - \mathbb{E}\left[c\left(L_{1\ldots i-1}\right)\right]\right) \cdot \left(\mathbb{E}\left[b\left(L_{1\ldots i-1}\right)\right] + \mathbb{E}\left[b\left(L_{1\ldots i}\right)\right]\right).$$

Condition 2 implies that $b\left(L_{1\ldots i}\right) = \sum_{i'=1}^{i} b\left(L_{i'}\right)$; applying linearity, we obtain:

$$\mathbb{E}\left[b\left(L_{1\ldots i}\right)\right] = \sum_{i'=1}^{i} \mathbb{E}_{y_{i'}|x_{1\ldots i'},y_{1\ldots i'-1},L}\left[b\left(L_{i'}\right)\right]$$

(idem. for cost). Finally, condition 3 implies that:

$$\mathbb{E}\left[b\left(L_{1\ldots i}\right)\right] = \sum_{i'=1}^{i} \mathbb{E}_{y_{i'}|x_{i'},L}\left[b\left(L_{i'}\right)\right]$$

(idem. for cost). This result allows us to statically precompute the expected cost and benefit of each instance. Because these quantities can be computed independently of one another, we can represent each instance x_i by a line segment with fixed width and height—the expected cost and benefit, respectively, according to the current model–and statically compute the area using these line segments.

We now prove that, under these conditions, selecting instances in non-strict slope-decreasing order (i.e., according to ROI) maximizes AUC.

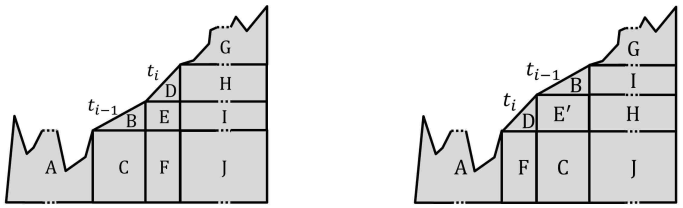


Figure 8.1: A comparison of the area under the curve when swapping two adjacent line segments to put them in slope-decreasing order. (a) total ordering T (b) T' (see text).

Theorem 1. *Consider a set S of n arbitrary line segments $s_1, \ldots, s_n$. The area under the curve that results from joining the segments end to end is a maximum if the segments are in non-strict slope-decreasing order.*

Proof. The proof is by contradiction. Suppose that joining the segments end-to-end in non-strict slope-decreasing order does not result in maximal area. Then, there exists a (non-strict) total ordering on S that is *not* non-strict slope-decreasing but whose resulting area is a maximum; call this ordering T. For notational convenience, we will let t_i denote the segment in S corresponding to the i^{th} element according to T. Since T is not in not-strict slope-decreasing order, there must exist an i such that the slope of t_i is greater than t_{i-1}. Define a new total ordering T' on S by swapping the order of t_i and t_{i-1} relative to T.

We sketch the rest of the proof with the aid of Figure 8.1, which depicts the aforementioned swap. It can be seen in the figure that all of the area under the curve remains the same after the swap, except for the areas marked E and E'. Since T leads to maximal area, the area of E must be greater than the area of E'. Let h_i and w_i represent the height and width of segment i. Then,

$$area(E) > area(E')$$

$$w_i \cdot h_{i-1} > w_{i-1} \cdot h_i$$

$$\frac{h_{i-1}}{w_{i-1}} > \frac{h_i}{w_i},$$

i.e., the slope of t_{i-1} must be greater than the slope of t_i. But this is a contradiction; therefore, T cannot lead to maximal area. Consequently, there is no T *not* in non-strict slope-decreasing order that leads to maximal area; therefore, the only order that produces maximal area is slope-non-increasing. □

As previously mentioned, the aforementioned conditions create a series of line segments as required by Theorem 1. Thus, if the conditions are met, then the sequence $x_1^*, \ldots, x_N^*$ that maximizes AUC is the sequence that is in non-strict slope-non-increasing order, by Theroem 1. Consequently, the optimal instance to select is the one with the highest slope. Therefore, under these conditions, ROI (see equation 8.1) would maximize AUC.

8.4 Methods

In this section, we describe our methodology for empirically assessing the degree to which the conditions of Section 2 hold in practice and define what we mean by practical contexts. As previously mentioned, testing these conditions across several combinations of benefit and cost estimators produces a large quantity of results required for in-depth analysis; we therefore limit our experiments to a single task: English part-of-speech (POS) tagging on the POS-tagged Wall Street Journal text in the Penn Treebank version 3 [56].

For this task, we employ Maximum Entropy Markov Models (MEMMs) to model the distribution of tags given words, $p(\mathbf{t}|\mathbf{w})$. All simulations are run on dual hex-core Intel Westmere 2.67 GHz CPUs equipped with 24 GB of RAM. AL typically begins with a small set of randomly selected instances; we use 100 instances that are annotated "from scratch," i.e., without AL. However, we do account for the cost incurred by annotating the seed set using the cost simulation described below. Each experiment is run 5 times with a different random seed. For TVE (a committee-based approach; see below), we use a committee size of 5 and train all members in parallel. We additionally score instances in parallel, using 4 threads; the remaining processors are used for training the cost model, evaluating benefit, and garbage collection. For non-committee methods, we found that extra scoring threads do not improve results.

8.4.1 Cost Simulation

Since AL is evaluated on the basis of cost, an AL simulation must be capable of simulating the cost of annotating any given instance, including any random noise. We use the linear model from Ringger et al. [74] to predict the length of time required to annotate a sentence. This model assumes that instances are pre-annotated using an automatic annotation model and the task of the annotator is to correct the errors from the predictive model. The cost to annotate a sequence $\mathbf{w}$ pre-annotated with tags $\mathbf{t}$, where we will denote the true tags of the instance as $\mathbf{y}$, is:

$$cost(\mathbf{w},\mathbf{y},\mathbf{t}) = \alpha + \beta \cdot |\mathbf{w}| + \gamma \cdot \sum_{i=1}^{|\mathbf{y}|} \mathbb{1}(y_i \neq t_i) \tag{8.5}$$

The latter sum represents the number of tags from the pre-annotation that the annotator changed. We estimate the parameters of the linear model ($\alpha = 50.534, \beta = 2.638, \gamma = 4.440$) using the user-study data from Ringger et al. [74]. To add noise to the simulated times, we generate a random deviate from a shifted Gamma distribution having mean equal to the time predicted by the model, a variance of 5063.35 (the empirical variance of the user-study data), and a shift of 10.0 (near the minimum time). We chose a (shifted) Gamma distribution because the data from the user study appear to be Gamma distributed; in addition, the generated values are guaranteed to always be positive.

8.4.2 Cost Estimation

For our ROI estimators and other experiments, we use the same linear model for cost estimation as above (i.e., equation 8.5). However, we learn the coefficients using the data obtained during AL (ultimately obtained from the noisy simulation). Since we do not know which of the automatically pre-annotated tags are incorrect during estimation, we must compute the expected number of incorrect tags in place of the sum in equation 8.5.

The results of our experiments are potentially better than in practice since our cost estimate has exactly the same form as the simulated true cost. However, the results are still useful because

(1) the gamma-distributed noise in the true cost has high variance and (2) the estimate is computed in expectation (using the learned model).

8.4.3 Benefit Estimation

ROI's numerator is an estimate of the benefit of obtaining a label for a given instance. As previously mentioned, optimal benefit estimators are impractical for structured learning problems; uncertainty-based heuristics are typically employed instead. Let the r.v. $\mathbf{T}$ represent a sequence of tag assignments for sentence $\mathbf{w}$. Drawing mostly from previous studies, we consider the following:

Constant (CONST). This estimator assumes all instances have equal benefit.

Approximate Token Entropy (ATE). Token entropy [80] approximates the true sequence entropy as the sum of the entropy of the individual marginal distributions $p(t_i|\mathbf{w})$. The marginal distributions can in turn be approximated as $p(t_i|\mathbf{w}) \approx p(t_i|t^*_{i-1}, \mathbf{w})$ where t^*_{i-i} is the $(i-1)th$ tag in the Viterbi best sequence $\mathbf{t}^*$; a beam search can significantly reduce computation.

Monte Carlo Entropy (MCE). MCE uses a Monte Carlo approximation to compute the entropy, i.e., $\mathbb{E}[-\log p(\mathbf{t}|\mathbf{w})]$, using samples taken from $\mathbf{T}|\mathbf{w}$ (the trained MEMM).

N-best Sequence Entropy (NSE). NSE [80] approximates sequence entropy by computing the entropy of the top-n sequences, where the probabilities are re-normalized to sum to unity.

Least Confidence (LC). LC [18], in contrast to entropy, is not concerned with the distribution over the entire support, but rather focuses on the best option and its complement (the rest of the support). It is the probability of being wrong, i.e., $1 - \max_{\mathbf{t}} p(\mathbf{t}|\mathbf{w})$.

Negative Max Log Probability (NMLP). NMLP [35] is defined as $-\max_{\mathbf{t}} \log p(\mathbf{t}|\mathbf{w})$; it ranks instances the same as LC but with different scores under the assumption that the relationship between probabilities and change in accuracy is logarithmic rather than linear.

Token Vote Entropy (TVE). TVE [27] uses a committee of classifiers trained from bootstrapped samples of the annotated data. For each word, each committee member votes for the tag it

predicts for its word; the entropy of the distribution over votes is summed over each word in the sentence.

8.4.4 Practical Active Learning

We are interested in empirically testing ROI outside of the clean mathematical environment implied by Section 2. Previous work has assumed that annotators are constantly and instantly available so that the machine may request an annotation at its leisure; we call this approach learner-initiated AL. However, the reverse is true in practice (e.g., crowd-sourcing), i.e., annotators request instances from the machine on demand, a scenario we call annotator-initiated AL. In either paradigm, the time an annotator spends waiting for the machine affects cost, either directly (when annotators are paid by the hour), or indirectly. We use the "Parallel No-Wait" active learning framework [35] that follows the annotator-initiated paradigm. Although waiting cost is eliminated, selected instances are not guaranteed to be "optimal" w.r.t. to the most recent models, a phenomenon known as staleness. We further extend the framework by training the cost model in parallel to model training and instance scoring.

Realistic annotation environments also often involve multiple annotators (cf. [24]). We take an incremental step towards allowing multiple annotators by assuming that all annotators are infallible and have the same distribution over the amount of time to annotate any given instance. Under these circumstances, each instance needs to be annotated only once, and annotators are interchangeable. In our simulations, we simulate in real time 20 tireless oracles who continuously and simultaneously annotate instances for 50 hours each. Note that having annotators that are constantly and instantly available (as in our experiments) is the worst possible case for the no-wait framework since models are maximally out-of-date. Thus, this simulation provide a lower-bound (empirically) on the AUC for more realistic annotation scenarios.

8.5 From Theory to Practice: To What Degree Are the Conditions Met?

In this section, we empirically test some of the conditions from the preceding analysis in practical contexts. For the purposes of this work, we are mostly interested in examining conditions 1 and 4. While condition 3 is assumed in most previous work, we leave quantification of the effects of violating this condition and the related condition 2 to future work.

For these experiments, it is necessary to estimate true benefit and cost. Due to the complexity of so doing, we compute the various metrics along a *passive* learning curve (i.e., without AL). We compute the true cost of each instance as described above. In order to estimate the true benefit of a particular instance at a particular point on the learning curve, we assume that the true benefit of an instance is the change in held-out accuracy that would result from incorporating the instance with its annotation into the training data; we ignore the effects of future choices. We compute the change in accuracy (benefit) by adding the instance and its true label to the training data, retraining the model, and then computing the model's new accuracy on the held-out set. The process is repeated to compute the true benefit of at least 1,000 instances and the statistics noted below are averaged over 5 random initial training sets.

Is the covariance of cost and benefit zero? Using the aforementioned methodology, we compute Pearson's correlation coefficient (a normalized form of covariance) between benefit and cost. As seen in figure 8.2a, true benefit and cost have virtually no correlation when model quality is high, and is only weakly correlated in the early stages. Thus, condition 2 roughly holds.

To what extent is the cost estimate a scalar multiple of true cost? Using the technique mentioned above, we produce pairs of true cost and estimated cost at various locations along the learning curve and compute R^2 values of a linear model estimated with the y-intercept fixed at zero. Pearson's correlation coefficient is inappropriate since it would allow for the cost estimate to additionally be shifted. Note that an R^2 of 1.0 would indicate that the cost estimate was an exact scalar multiple of the true cost, while a zero would indicate no scalar relationship. We repeat this test with differing amounts of variance in the simulated cost, which allows us to assess the effect of poor cost models (good models will account for most of the variance). The results are shown in

Figure 8.2b. The exponential decay as variance increases underscores the importance of accounting for as much variance as possible in the cost model. We found the R^2 values to be around 0.745 and 0.785 when the variance was equal to that of the aforementioned user study (and the one used through the remainder of the experiments). We note that Settles et al. [81] and Arora and Nyberg [4] report R^2 values for cost models for *different* tasks on the order of 0.3–0.4. However, timing for POS-tagging is much more predictable than the other tasks, resulting in orders of magnitude lower variance in the observed cost; at the comparable levels of variance, our R^2 values more closely match those of previous work. In short, the high R^2 values on our problem are indicative of a highly scalar relationship between true and estimated cost as per condition 4.

To what extent are various benefits estimators scalar multiples of true benefit? We repeat the experiment described for cost, but reporting the R^2 values for the fit between true benefit and several benefit estimators; Figure 8.2c depicts the results. Although the R^2 values are much worse than for the cost estimate, they are still reasonable. Most of the separation of algorithms (where it exists statistically) occurs during the beginning stages of learning. NMLP has a slight (though not statistically significant) advantage over ATE and MCE while all three are more linearly related to true benefit than NSE and LC. Once the model achieves 91% accuracy, there is no separation. We suspect that the quality of the estimate decreases as a function of model accuracy due to the fact that well-trained MEMMs often produce peaked estimates of $p(\mathbf{t}|\mathbf{w})$ and all of the benefit estimates are functions of this value. It may also be that the benefit estimates do not exhibit the same level of diminishing returns as true benefit. The results suggest that condition 4 holds weakly for benefit estimators.

Are instances with the highest slopes being selected? The success of ROI depends on its ability to select the instance with the highest slope. Using the aforementioned setup, we compute the largest slope in the sampled instances on the basis of estimated benefit and cost and divide it by the largest slope according to the true values; we call this value the Percentage of Maximum Attainable Slope (POMAS). Since multiple instances can be selected using the same model in the no-wait framework, we repeat this procedure for the second highest slopes, etc., for the top-20

slopes and average them. The results are in Figure 8.2d. The separation between algorithms at the beginning mirror those of Figure 8.2c. In addition, the relatively low scores further suggest that the estimators are falling far short of their potential (perhaps due in part to batching). Although the previous experiments showed that the conditions hold reasonably well in practice, their combined shortcomings lead to relatively sub-optimal selections.

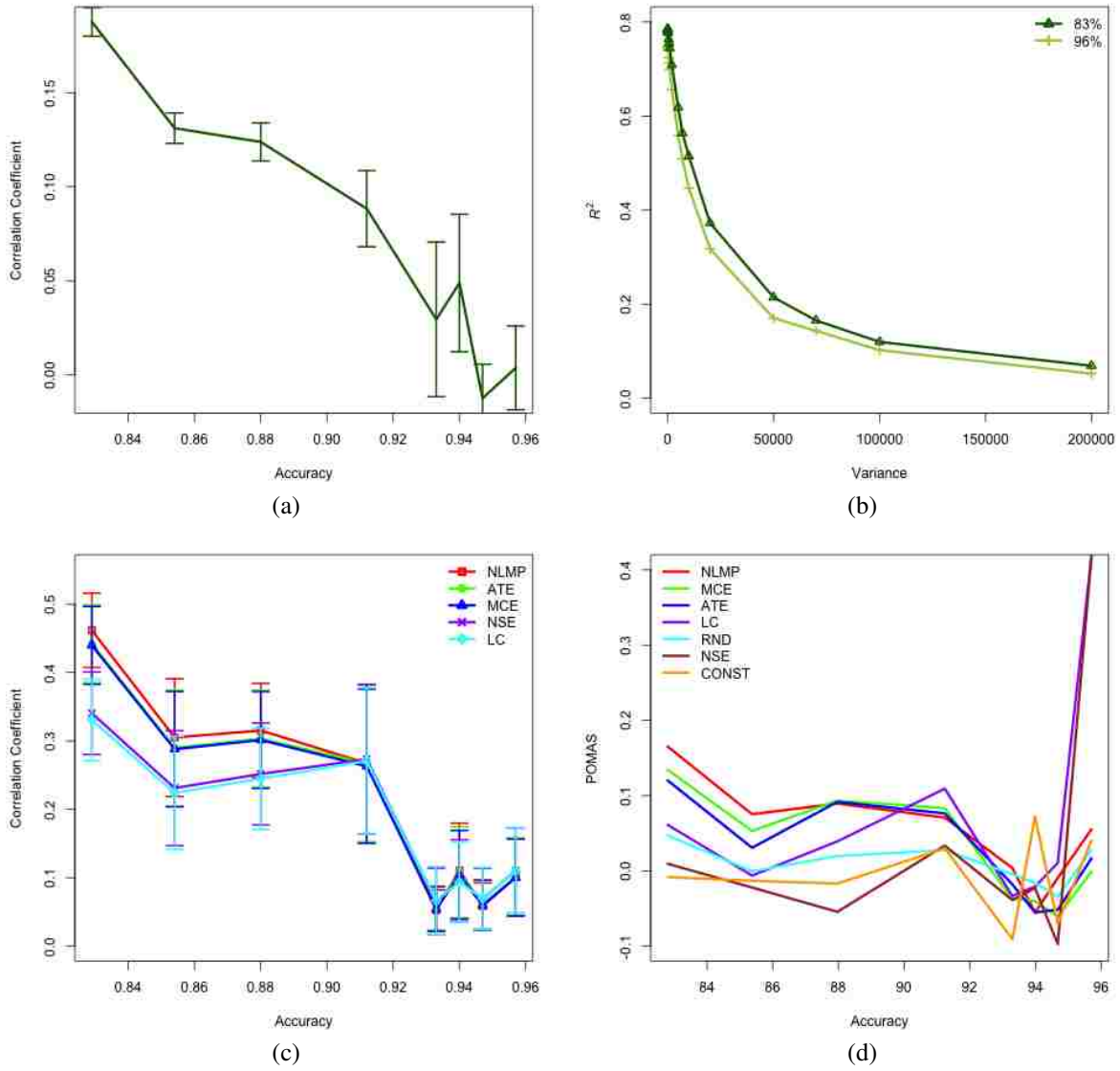


Figure 8.2: At various points on the learning curve: (a) correlation between true cost and true benefit (b) R^2 values representing the degree to which the cost estimate is a scalar multiple of the true cost, for varying amounts of variance in the noise model of the true cost simulation (c) R^2 values representing the degree to which various benefit estimators are scalar multiples of true benefit (d) POMAS of the top-20 instances.

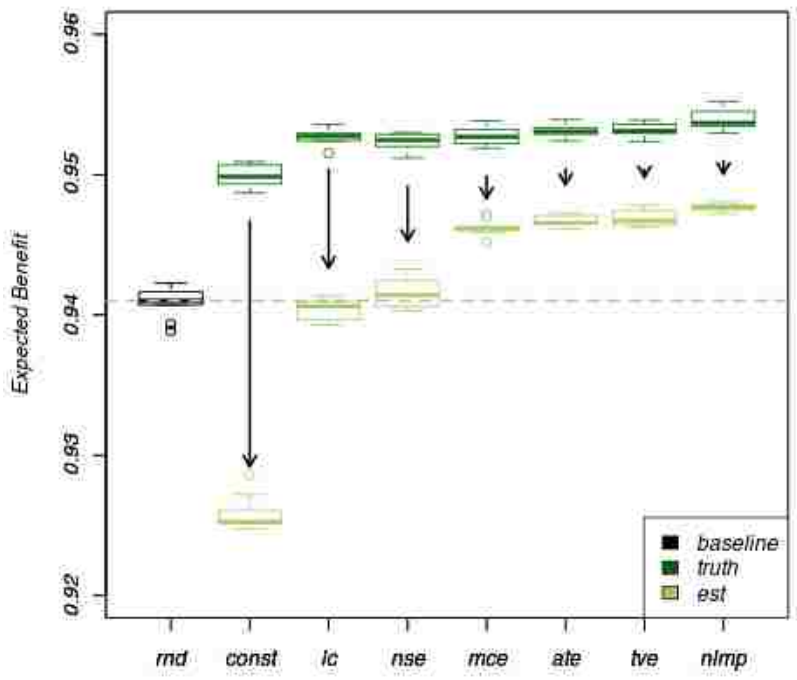


Figure 8.3: Expected benefit for various benefit estimators with true and estimated costs. The median baseline performance (rnd) is depicted as a dashed line and is the same for both experiments. Estimated cost affects the benefit estimates to different degrees.

8.6 Active Learning Results and Discussion

The previous experiments were conducted outside of the context of AL in order to gain insight into how well the conditions of section 2 are met in practice. However, the most direct and important evaluation is the comparison of the actual quantity of interest, AUC, in the type of practical AL defined above. We compare normalized AUC (which represents the expected benefit across all costs) for several benefit estimators and two cost estimates and combine the results from the previous section with the theory from Section 2 to explain the outcome.

Although not predicted by the theory per se, we would expect AUC to decrease with degradations in the cost and/or benefit estimates. First, we compare AUC using the true cost in the ROI calculations (thus satisfying one half of condition 4) and compare the results to using estimated cost learned during AL. The results are displayed in Figure 8.3. Interestingly, when cost is exactly known (perfectly predictable), all estimators–even CONST–readily outperform the random

baseline. Furthermore, the difference between most of the estimators (except perhaps CONST) is not statistically significant, which suggests that a good cost estimate may be capable of overcoming deficiencies in even very poor benefit estimators like CONST. Not surprisingly, all algorithms perform worse when using the learned estimate of cost during AL (indicated by the downward arrows). Although we found the quality of the learned cost models to be quite good in terms of learned coefficients, the MSE is still high (on the order of the variance in the simulated times).

Further support that AUC follows the quality of estimates (an approximation to condition 4) comes from the fact that the performance of the various algorithms exactly follows the quality of the corresponding benefit estimate (see Figures 8.2c and 8.2d). In fact, LC and NSE do no better than random and CONST does much worse. Upon further examination, we found a common property between these benefit estimators, namely, that most scores fall within an extremely narrow range; i.e., there was very little difference between the scores of most instances (CONST being the extreme case). NSE differs from the other entropy estimates primarily in the re-normalization. Due to their nature, structured prediction problems have very large supports which tend to have long tails. Therefore the top-n probabilities grossly underrepresent these distributions. In these circumstances, renormalization has the effect of making scores very similar to each other—even for instances of differing lengths. Similarly, although LC and NMLP would rank instances the same (before dividing by cost; dividing by cost alters the rankings), the *log* in NLMP produces greater spread in the score. In contrast, the cost estimates are better dispersed and therefore tend to dominate ROI. To illustrate, consider the extreme case of CONST by substituting an arbitrary constant for benefit in equation 8.1; instances are selected lowest-expected-cost first. On our particular task, this scenario is particularly undesirable as the shortest sentences are nearly always the cheapest but disproportionately information poor (a contributing factor to the non-zero correlation). In more general terms, as the spread in the benefit estimates approaches zero (as in CONST), the cost estimates increasingly become the discriminating factor. While this behavior is correct for perfect benefit and cost estimates, it is problematic when condition 4 is violated.

A few additional results are noteworthy. The Monte-Carlo estimate of entropy (MCE) may perform slightly worse than ATE because of the additional expense incurred in the sampling process (recall that the expensive scoring algorithms are naturally penalized in the parallel framework). Also, TVE appears to be on-par with entropy-based selectors for this problem, but slightly inferior to NMLP. We suspect that the primary difference is the extra expense incurred in training the additional models.

The mixed results of previous work are explainable based on our analysis. While condition 4 *requires* that cost and benefit estimators be scalar multiples of the true values, our empirical results suggest that better estimates yield higher AUC. We have explained why NSE has poor mathematical properties for structured learning tasks and is therefore expected to produce relatively low AUC, hence the negative results on the structured prediction tasks of Settles et al. [81]. In contrast, the authors report positive results on a standard classification problem using a much better benefit estimator, i.e., entropy (analogous to the other entropy-based estimates we investigated). We have also explained the poor properties of LC for structured prediction; the results of Tomanek and Hahn [89] present further evidence. Interestingly, they find that exponentiating LC leads to positive results. Mathematically, $exp(\beta(1-p))$ behaves similarly to $-log(p)$ (NLMP) in that they both separate scores that are close together—the former much more so than the latter, especially for probabilities of the very low magnitudes seen in structured prediction problems. In sum, the negative results of previous work are due to poor benefit estimators, in particular LC and NSE; in contrast, positive results are due to better benefit estimators.

8.7 Conclusions and Future Work

Return-on-investment-based AL successfully reduces annotation costs in practice by maximizing the area under the cost/benefit curve. This chapter has provided an initial theoretical justification for ROI-based AL in a bottom-up fashion. It is the first step towards allowing us to predict when ROI should work. Although this analysis is predicated on conditions that are violated in practice, we have empirically demonstrated that better estimators lead to higher AUC. Although we focused our

empirical analysis on a single task, other studies have applied ROI to several tasks and problem types, and their results are consistent with our analysis. As a result of this work, we recommend that practitioners carefully select their benefit and cost estimators, ensuring that they are "good" estimators as described above. In particular, estimators that produce scores with relatively little difference relative to the differences in cost should be avoided; thus, NSE and LC are generally poor choices for structured prediction whereas NLMP and other entropy-based methods work reasonably well when accuracy is the benefit. In sum, cost-conscious active learning can be successful in practice when using ROI, so long as care is taken to choose reasonably good benefit and cost estimators.

Our empirical results suggest that there is yet room for discovering improved benefit estimators; future work could focus on directly and tractably estimating true cost and benefit for structured prediction problems, and automatically tuning heuristic estimators to match true benefit during AL. Future work may also benefit from investigating a different set of conditions for simplifying equation 8.4.

Chapter 9

Parallel Active Learning: Eliminating Wait Time with Minimal Staleness[†]

Abstract

A practical concern for Active Learning (AL) is the amount of time human experts must wait for the next instance to label. We propose a method for eliminating this wait time independent of specific learning and scoring algorithms by making scores always available for all instances, using old (stale) scores when necessary. The time during which the expert is annotating is used to train models and score instances–in parallel–to maximize the recency of the scores. Our method can be seen as a parameterless, dynamic batch AL algorithm. We analyze the amount of staleness introduced by various AL schemes and then examine the effect of the staleness on performance on a part-of-speech tagging task on the Wall Street Journal. Empirically, the parallel AL algorithm effectively has a batch size of one and a large candidate set size but eliminates the time an annotator would have to wait for a similarly parameterized batch scheme to select instances. The exact performance of our method on other tasks will depend on the relative ratios of time spent annotating, training, and scoring, but in general we expect our parameterless method to perform favorably compared to batch when accounting for wait time.

[†]Robbie Haertel, Paul Felt, Eric Ringger, and Kevin Seppi. Parallel active learning: Eliminating wait time with minimal staleness. In *Proceedings of the HLT-NAACL 2010 Workshop on Active Learning for Natural Language Processing*, pages 3341. Association for Computational Linguistics, 2010.

9.1 Introduction

Recent emphasis has been placed on evaluating the effectiveness of active learning (AL) based on realistic cost estimates [5, 36, 81]. However, to our knowledge, no previous work has included in the cost measure the amount of time that an expert annotator must wait for the active learner to provide instances. In fact, according to the standard approach to cost measurement, there is no reason not to use the theoretically optimal (w.r.t. a model, training procedure, and utility function) (but intractable) approach (see [36]).

In order to more fairly compare complex and time-consuming (but presumably superior) selection algorithms with simpler (but presumably inferior) algorithms, we describe "best-case" (minimum, from the standpoint of the payer) and "worst-case" (maximum) cost scenarios for each algorithm. In the best-case cost scenario, annotators are paid only for the time they spend actively annotating. The worst-case cost scenario additionally assumes that annotators are always on-the-clock, either annotating or waiting for the AL framework to provide them with instances. In reality, human annotators work on a schedule and are not always annotating or waiting, but in general they expect to be paid for the time they spend waiting for the next instance. In some cases, the annotator is not paid directly for waiting, but there are always opportunity costs associated with time-consuming algorithms, such as time to complete a project. In reality, the true cost usually lies between the two extremes.

However, simply analyzing only the best-case cost, as is the current practice, can be misleading, as illustrated in Figure 9.1. When excluding waiting time for a particular selection algorithm[1] ("AL Annotation Cost Only"), the performance is much better than the cost of random selection ("Random Total Cost"), but once waiting time is accounted for ("AL Total cost"), the AL approach can be worse than random. Given only the best-case cost, this algorithm would appear to be very desirable. Yet, practitioners would be much less inclined to adopt this algorithm knowing that the worst-case cost is potentially no better than random. In a sense, waiting time serves as a

[1] We use the ROI-based scoring algorithm [36] and the zero-staleness technique, both described below.

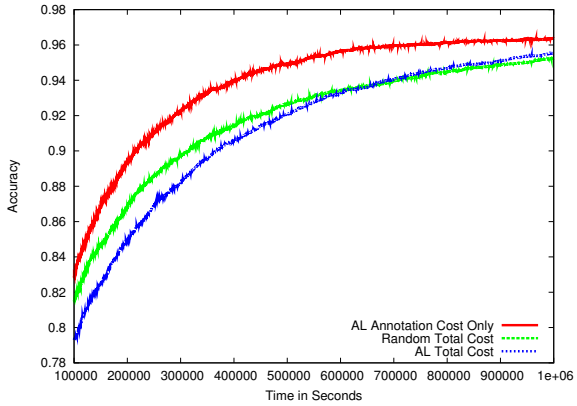


Figure 9.1: Accuracy as a function of cost (time). Side-by-side comparison of best-case and worst-case cost measurement scenarios reveals that not accounting for the time required by AL to select instances affects the evaluation of an AL algorithm.

natural penalty for expensive selection algorithms. Therefore, conclusions about the usefulness of AL selection algorithms should take both best-case and worst-case costs into consideration.

Although it is current practice to measure only best-case costs, Tomanek et al. [91] mention as a desideratum for practical AL algorithms the need for what they call fast selection time cycles, i.e., algorithms that minimize the amount of time annotators wait for instances. They address this by employing the batch selection technique of Engelson and Dagan [27]. In fact, most AL practitioners and researchers implicitly acknowledge the importance of wait time by employing batch selection.

However, batch selection is not a perfect solution. First, using the traditional implementation, a "good" batch size must be specified beforehand. In research, it is easy to try multiple batch sizes, but in practice where there is only one chance with live annotators, specifying a batch size is a much more difficult problem; ideally, the batch size would be set during the process of AL. Second, traditional methods use the same batch size throughout the entire learning process. However, in the beginning stages of AL, models have access to very little training data and retraining is often much less costly (in terms of time) than in the latter stages of AL in which models are trained on large amounts of data. Intuitively, small batch sizes are acceptable in the beginning stages, whereas large batch sizes are desirable in the latter stages in order to mitigate the time cost of training. In fact, Haertel et al. [36] mention the use of an increasing batch size to speed up their simulations, but details are scant and the choice of parameters for their approach is task- and dataset-dependent.

Also, the use of batch AL causes instances to be chosen without the benefit of all of the most recently annotated instances, a phenomenon we call *staleness* and formally define in Section 9.2. Finally, in batch AL, the computer is left idle while the annotator is working and vice-versa.

We present a parallel, parameterless solution that can eliminate wait time irrespective of the scoring algorithm and training method. Our approach is based on the observation that instances can always be available for annotation if we are willing to serve instances that may have been selected without the benefit of the most recent annotations. By having the computer learner do work while the annotator is busy annotating, we are able to mitigate the effects of using these older annotations.

The rest of this paper will proceed as follows: Section 9.2 defines staleness and presents a progression of four AL algorithms that strike different balances between staleness and wait time, culminating in our parallelized algorithm. We explain our methodology and experimental parameters in Section 9.3 and then present experimental results and compare the four AL algorithms in Section 9.4. Conclusions and future work are presented in Section 9.5.

9.2 From Zero Staleness to Zero Wait

We work within a pool- and score-based AL setting in which the active learner selects the next instance from an unlabeled pool of data $\mathscr{U}$. A scoring function σ (aka scorer) assigns instances a score using a model θ trained on the labeled data $\mathscr{A}$; the scores serve to rank the instances. Lastly, we assume that an unerring oracle provides the annotations. These concepts are demonstrated in Algorithm 3.

In this section, we explore the trade-off between staleness and wait time. In order to do so, it is beneficial to quantitatively define staleness, which we do in the context of Algorithm 3. After each model θ is trained, a stamp is associated with that θ that indicates the number of annotated instances used to train it (see line 3). The staleness of an item is defined to be the difference between the current number of items in the annotated set and the stamp of the scorer that assigned the instance a score. This concept can be applied to any instance, but it is particularly informative to speak of the staleness of instances at the time they are actually annotated (we will simply refer

Input: A seed set of annotated instances $\mathscr{A}$, a set of pairs of unannotated instances and their initial scores $\mathscr{S}$, scoring function σ, the candidate set size N, and the batch size B
Result: $\mathscr{A}$ is updated with the instances chosen by the AL process as annotated by the oracle

```
1  while 𝒮 ≠ ∅ do
2      θ ← TrainModel(𝒜)
3      stamp ← |𝒜|
4      𝒞 ← ChooseCandidates(𝒮,N)
5      𝒦 ← {(c[inst], σ(c[inst], θ)) | c ∈ 𝒞}
6      𝒮 ← S − C ∪ K
7      𝒯 ← pairs from 𝒦 with c[score] in the top B scores
8      for t ∈ 𝒯 do
9          𝒮 ← 𝒮 − t
10         staleness ← |A| − stamp // unused
11         𝒜 ← 𝒜 ∪ Annotate(t)
12     end
13 end
```

Algorithm 3: Pool- and score-based active learner.

to this as staleness, disambiguating when necessary; see line 10). Intuitively, an AL scheme that chooses instances having less stale scores will tend to produce a more accurate ranking of instances.

9.2.1 Zero Staleness

There is a natural trade-off between staleness and the amount of time an annotator must wait for an instance. Consider Algorithm 3 when $B = 1$ and $N = \infty$ (we refer to this parameterization as **zerostale**). In line 8, a single instance is selected for annotation ($|\mathscr{T}| = B = 1$); the staleness of this instance is zero since no other annotations were provided between the time it was scored and the time it was removed. Therefore, this algorithm will never select stale instances and is the only way to guarantee that no selected instances are stale.

However, the zero staleness property comes with a price. Between every instance served to the annotator, a new model must be trained and every instance scored using this model, inducing potentially large waiting periods. Therefore, the following options exist for reducing the wait time:

1. Optimize the learner and scoring function (including possible parallelization)
2. Use a different learner or scoring function

3. Parallelize the scoring process

4. Allow for staleness

The first two options are specific to the learning and scoring algorithms, whereas we are interested in reducing wait time independent of these in the general AL framework. We describe option 3 in section 9.2.4; however, it is important to note that when training time dominates scoring, the reduction in waiting time will be minimal with this option. This is typically the case in the latter stages of AL when models are trained on larger amounts of data.

We therefore turn our attention to option 4: in this context, there are at least three ways to decrease the wait time: (A) train less often, (B) score fewer items, or (C) allow old scores to be used when newer ones are unavailable. Strategies A and B are the batch selection scheme of Engelson and Dagan [27]; an algorithm that allows for these is presented as Algorithm 3, which we refer to as "traditional" batch, or simply **batch**. We address the traditional batch strategy first and then address strategy C.

9.2.2 Traditional Batch

In order to train fewer models, Algorithm 3 can provide the annotator with several instances scored using the same scorer (controlled by parameter B); consequently, staleness is introduced. The first item annotated on line 11 has zero staleness, having been scored using a scorer trained on all available annotated instances. However, since a model is not retrained before the next item is sent to the annotator, the next items have staleness $1, 2, \cdots, B-1$. By introducing this staleness, the time the annotator must wait is amortized across all B instances in the batch, reducing the wait time by approximately a factor of B. The exact effect of staleness on the *quality* of instances selected is scorer- and data-dependent.

The parameter N, which we call the candidate set size, specifies the number of instances to score. Typically, candidates are chosen in round-robin fashion or with uniform probability (without replacement) from $\mathscr{U}$. If scoring is expensive (e.g., if it involves parsing, translating, summarizing, or some other time-consuming task), then reducing the candidate set size will reduce

the amount of time spent scoring by the same factor. Interestingly, this parameter does not affect staleness; instead, it affects the probability of choosing the same B items to include in the batch when compared to scoring all items. Intuitively, it affects the probability of choosing B "good" items. As N approaches B, this probability approaches uniform random and performance approaches that of random selection.

9.2.3 Allowing Old Scores

One interesting property of Algorithm 3 is that line 7 guarantees that the only items included in a batch are those that have been scored in line 5. However, if the candidate set size is small (because scoring is expensive), we could compensate by reusing scores from previous iterations when choosing the best items. Specifically, we change line 7 to instead be:

$$\mathscr{T} \leftarrow \text{pairs from } \mathscr{S} \text{ with } c[score] \text{ in the top } B \text{ scores}$$

We call this **allowold**, and to our knowledge, it is a novel approach. Because selected items may have been scored many "batches" ago, the expected staleness will never be less than in **batch**. However, if scores do not change much from iteration to iteration, then old scores will be good approximations of the actual score and therefore not all items necessarily need to be rescored every iteration. Consequently, we would expect the quality of instances selected to approach that of **zerostale** with less waiting time. It is important to note that, unlike **batch**, the candidate set size does directly affect staleness; smaller N will increase the likelihood of selecting an instance scored with an old model.

9.2.4 Eliminating Wait Time

There are portions of Algorithm 3 that are trivially parallelizable. For instance, we could easily split the candidate set into equal-sized portions across P processors to be scored (see line 5). Furthermore, it is not necessary to wait for the scorer to finish training before selecting the candidates. And,

as previously mentioned, it is possible to use parallelized training and/or scoring algorithms. Clearly, wait time will decrease as the speed and number of processors increase. However, we are interested in parallelization that can guarantee zero wait time independent of the training and scoring algorithms without precluding these other forms of parallelization.

All other major operations of Algorithm 3 have serial dependencies, namely, we cannot score until we have trained the model and chosen the candidates, we cannot select the instances for the batch until the candidate set is scored, and we cannot start annotating until the batch is prepared. These dependencies ultimately lead to waiting.

The key to eliminating this wait time is to ensure that all instances have scores at all times, as in **allowold**. In this way, the instance that currently has the highest score can be served to the annotator without having to wait for any training or scoring. If the scored instances are stored in a priority queue with a constant time *extract-max* operation (e.g., a sorted list), then the wait time will be negligible. Even a heap (e.g., binary or Fibonacci) will often provide negligible overhead. Of course, eliminating wait time comes at the expense of added staleness as explained in the context of **allowold**.

This additional staleness can be reduced by allowing the computer to do work while the oracle is busy annotating. If models can retrain and score most instances in the amount of time it takes the oracle to annotate an item, then there will be little staleness.[2]

Rather than waiting for training to complete before beginning to score instances, the old scorer can be used until a new one is available. This allows us to train models and score instances in parallel. Fast training and scoring procedures result in more instances having up-to-date scores. Hence, the staleness (and therefore quality) of selected instances depends on the relative time required to train and score models, thereby encouraging efficient training and scoring algorithms. In fact, the other forms of parallelization previously mentioned can be leveraged to reduce staleness rather than attempting to directly reduce wait time.

[2]Since the annotator requests the next instance immediately after annotating the current instance, the next instance is virtually guaranteed to have a staleness factor of at least 1.

These principles lead to Algorithm 4, which we call **parallel** (for clarity, we have omitted steps related to concurrency). `AnnotateLoop` represents the tireless oracle who constantly requests instances. The call to `Annotate` is a surrogate for the actual annotation process and most importantly, the time spent in this method is the time required to provide annotations. Once an annotation is obtained, it is placed on a shared buffer $\mathscr{B}$ where it becomes available for training. While the annotator is, in effect, a producer of annotations, `TrainLoop` is the consumer which simply retrains models as annotated instances become available on the buffer. This buffer is analogous to the batch used for training in Algorithm 3. However, the size of the buffer changes dynamically based on the relative amounts of time spent annotating and training. Finally, `ScoreLoop` endlessly scores instances, using new models as soon as they are trained. The set of instances scored with a given model is analogous to the candidate set in Algorithm 3.

9.3 Experimental Design

Because the performance of the parallel algorithm and the "worst-case" cost analysis depend on wait time, we hold computing resources constant, running all experiments on a cluster of Dell PowerEdge M610 servers equipped with two 2.8 GHz quad-core Intel Nehalem processors and 24 GB of memory.

All experiments were on English part of speech (POS) tagging on the POS-tagged Wall Street Journal text in the Penn Treebank (PTB) version 3 [56]. We use sections 2-21 as initially unannotated data and randomly select 100 sentences to seed the models. We employ section 24 as the set on which tag accuracy is computed, but do not count evaluation as part of the wait time. We simulate annotation costs using the cost model from Ringger et al. [74]: $cost(\mathbf{s}) = (3.80 \cdot l + 5.39 \cdot c + 12.57)$, where l is the number of tokens in the sentence, and c is the number of pre-annotated tags that need correction, which can be estimated using the current model. We use the same model for pre-annotation as for scoring.

We employ the return on investment (ROI) AL framework introduced by Haertel et. al (2008). This framework requires that one define both a cost and benefit estimate and selects instances

Input: A seed set of annotated instances $\mathscr{A}$, a set of pairs of unannotated instances and their initial scores $\mathscr{S}$, and a scoring function σ

Result: $\mathscr{A}$ is updated with the instances chosen by the AL process as annotated by the oracle

```
1  ℬ ← ∅, θ ← null
2  Start(AnnotateLoop)
3  Start(TrainLoop)
4  Start(ScoreLoop)
5  procedure AnnotateLoop()
6      while 𝒮 ≠ ∅ do
7          t ← c from 𝒮 having max c[score]
8          𝒮 ← 𝒮 − t
9          ℬ ← ℬ ∪ Annotate(t)
10     end
11 end
12 procedure TrainLoop()
13     while 𝒮 ≠ ∅ do
14         θ ← TrainModel(𝒜)
15         𝒜 ← 𝒜 ∪ ℬ
16         ℬ ← ∅
17     end
18 end
19 procedure ScoreLoop()
20     while 𝒮 ≠ ∅ do
21         c ← ChooseCandidate(𝒮)
22         𝒮 ← 𝒮 − {c} ∪ {(c[inst], σ(c[inst], θ)) | c ∈ 𝒮}
23     end
24 end
```

Algorithm 4: parallel

that maximize $\frac{benefit(x)-cost(x)}{cost(x)}$. For simplicity, we estimate cost as the length of a sentence. Our benefit model estimates the utility of each sentence as follows: $benefit(\mathbf{s}) = -\log(\max_{\mathbf{t}} p(\mathbf{t}|\mathbf{s}))$ where $p(\mathbf{t}|\mathbf{s})$ is the probability of a tagging given a sentence. Thus, sentences having low average (in the geometric mean sense) per-tag probability are favored. We use a maximum entropy Markov model to estimate these probabilities, to pre-annotate instances, and to evaluate accuracy.

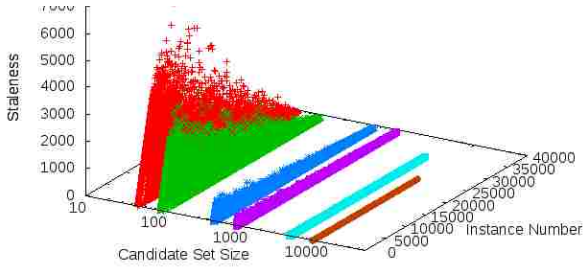


Figure 9.2: Staleness of the **allowold** algorithm over time for different candidate set sizes

9.4 Results

Two questions are pertinent regarding staleness: how much staleness does an algorithm introduce? and how detrimental is that staleness? For **zerostale** and **batch**, the first question was answered analytically in a previous section. We proceed by addressing the answer empirically for **allowold** and **parallel** after which we examine the second question.

Figure 9.2 shows the observed staleness of instances selected for annotation over time and for varying candidate set sizes for **allowold**. As expected, small candidate sets induce more staleness, in this case in very high amounts. Also, for any given candidate set size, staleness decreases over time (after the beginning stages), since the effective candidate set includes an increasingly larger percentage of the data.

Since **parallel** is based on the same allow-old-scores principle, it too could potentially see highly stale instances. However, we found the average per-instance staleness of **parallel** to be very low: **1.10**; it was never greater than **4** in the range of data that we were able to collect. This means that for our task and hardware, the amount of time that the oracle takes to annotate an instance is high enough to allow new models to retrain quickly and score a high percentage of the data before the next instance is requested.

We now examine effect that staleness has on AL performance, starting with **batch**. As we have shown, higher batch sizes guarantee more staleness so we compare the performance of several batch sizes (with a candidate set size of the full data) to **zerostale** and **random**. In order to tease out the effects that the staleness has on performance from the effects that the batches have on

wait time (an element of performance), we purposely ignore wait time. The results are shown in Figure 9.3. Not surprisingly, **zerostale** is slightly superior to the batch methods, and all are superior to random selection. Furthermore, **batch** is not affected much by the amount of staleness introduced by reasonable batch sizes: for $B < 100$ the increase in cost of attaining 95% accuracy compared to **zerostale** is 3% or less.

Recall that **allowold** introduces more staleness than **batch** by maintaining old scores for each instance. Figure 9.4 shows the effect of different candidate set sizes on this approach while fixing batch size at 1 (wait time is excluded as before). Larger candidate set sizes have less staleness, so not surprisingly performance approaches **zerostale**. Smaller candidate set sizes, having more staleness, perform similarly to random during the early stages when the model is changing more drastically each instance. In these circumstances, scores produced from earlier models are not good approximations to the actual scores so allowing old scores is detrimental. However, once models stabilize and old scores become better approximations, performance begins to approach that of **zerostale**.

Figure 9.5 compares the performance of **allowold** for varying batch sizes for a fixed candidate set size (5000; results are similar for other settings). As before, performance suffers primarily in the early stages and for the same reasons. However, a batch exacerbates the problem since multiple instances with poor scores are selected simultaneously. Nevertheless, the performance appears to mostly recover once the scorers become more accurate. We note that batch sizes of 5 and 10 increase the cost of achieving 95% accuracy by 3% and 10%, respectively, compared to **zerostale**. The implications for **parallel** are that staleness may not be detrimental, especially if batch sizes are small and candidate set sizes are large in the beginning stages of AL.

Figure 9.6 compares the effect of staleness on all four algorithms when excluding wait time ($B = 20$, $N = 5000$ for the batch algorithms). After achieving around 85% accuracy, **batch** and **parallel** are virtually indistinguishable from **zerostale**, implying that the staleness in these algorithms is mostly ignorable. Interestingly, **allowold** costs around 5% more than **zerostale** to

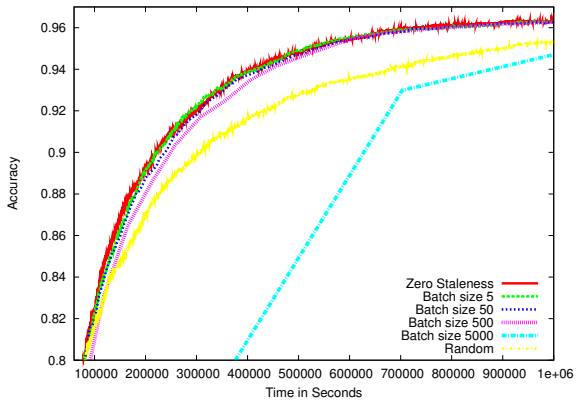


Figure 9.3: Effect of staleness due to batch size for **batch**, $N = \infty$

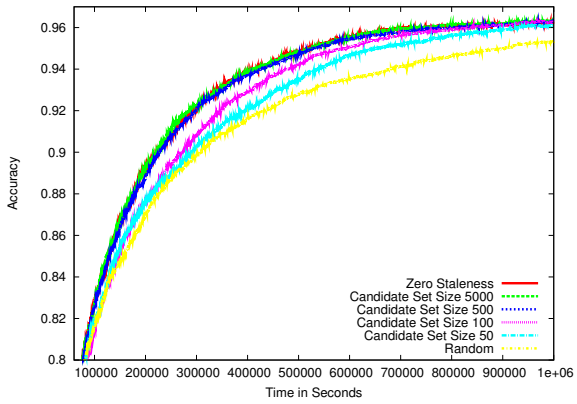


Figure 9.4: Effect of staleness due to candidate set size for **allowold**, $B = 1$

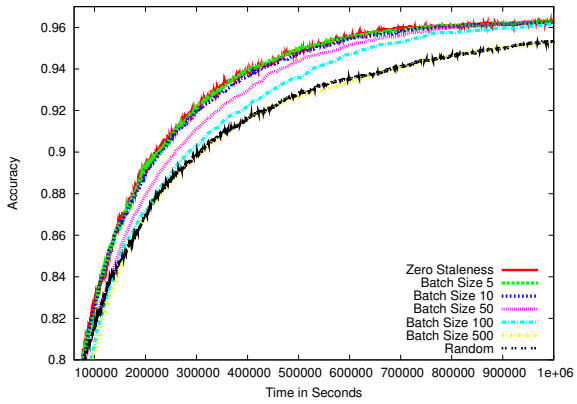


Figure 9.5: Effect of staleness due to batch size for **allowold**, $N = 5000$

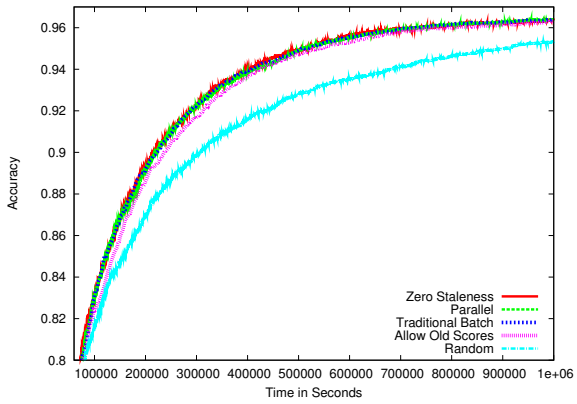


Figure 9.6: Comparison of algorithms (not including wait time)

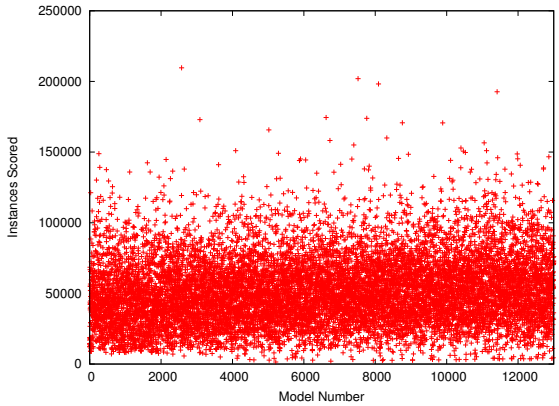


Figure 9.7: Effective candidate set size of **parallel** over time

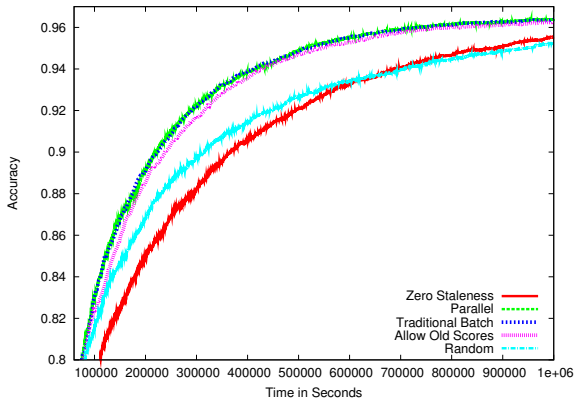


Figure 9.8: Comparison of algorithms (including wait time)

achieve an accuracy of 95%. We attribute this to increased levels of staleness which **parallel** combats by avoiding idle time.

Since the amount of data **parallel** uses to train models and score instances depends on the amount of time instances take to annotate, the "effective" candidate set sizes and batch sizes over time is of interest. We found that the models were always trained after receiving exactly one instance, within the data we were able to collect. Figure 9.7 shows the number of instances scored by each successive scorer, which appears to be very large on average: over 75% of the time the scorer was able to score the entire dataset. For this task, the human annotation time is much greater than the amount of time it takes to train new models (at least, for the first 13,000 instances). The net effect is that under these conditions, **parallel** is parameterized similar to **batch** with $B = 1$ and N very high, i.e., approaching **zerostale**, and therefore has very low staleness, yet does so without incurring the waiting cost.

Finally, we compare the performance of the four algorithms using the same settings as before, but include wait time as part of the cost. The results are in Figure 9.8. Importantly, **parallel** readily outperforms **zerostale**, costing 40% less to reach 95% accuracy. **parallel** also appears to have a slight edge over **batch**, reducing the cost to achieve 95% accuracy by a modest 2%; however, had the simulation continued, we may have seen greater gains given the increasing training time that occurs later on. It is important to recognize in this comparison that the purpose of **parallel** is not necessarily to significantly outperform a well-tuned batch algorithm. Instead, we aim to eliminate wait time without requiring parameters, while hopefully maintaining performance. These results suggest that our approach successfully meets these criteria.

Taken as a whole, our results appear to indicate that the net effect of staleness is to make selection more random. Models trained on little data tend to produce scores that are not reflective of the actual utility of instances and essentially produce a random ranking of instances. As more data is collected, scores become more accurate and performance begins to improve relative to random selection. However, stale scores are by definition produced using models trained with less data than is currently available, hence more staleness leads to more random-like behavior. This explains

why batch selection tends to perform well in practice for "reasonable" batch sizes: the amount of staleness introduced by batch ($\frac{B-1}{2}$ on average for a batch of size B) introduces relatively little randomness, yet cuts the wait time by approximately a factor of B.

This also has implications for our **parallel** method of AL. If a given learning algorithm and scoring function outperform random selection when using **zerostale** and excluding wait time, then any added staleness should cause performance to more closely resemble random selection. However, once waiting time is accounted for, performance could actually degrade below that of random. In **parallel**, more expensive training and scoring algorithms are likely to introduce larger amounts of staleness, and would cause performance to approach random selection. However, **parallel** has no wait time, and hence our approach should always perform at least as well as random in these circumstances. In contrast, poor choices of parameters in **batch** could perform worse than random selection.

9.5 Conclusions and Future Work

Minimizing the amount of time an annotator must wait for the active learner to provide instances is an important concern for practical AL. We presented a method that can eliminate wait time by allowing instances to be selected on the basis of the most recently assigned score. We reduce the amount of staleness this introduces by allowing training and scoring to occur in parallel while the annotator is busy annotating. We found that on PTB data using a MEMM and a ROI-based scorer that our parameterless method performed slightly better than a hand-tuned traditional batch algorithm, without requiring any parameters. Our approach's parallel nature, elimination of wait time, ability to dynamically adapt the batch size, lack of parameters, and avoidance of worse-than-random behavior, make it an attractive alternative to **batch** for practical AL.

Since the performance of our approach depends on the relative time spent annotating, training, and scoring, we wish to apply our technique in future work to more complex problems and models that have differing ratios of time spent in these areas. Future work could also draw on the

continual computation framework [39] to utilize idle time in other ways, e.g., to predict annotators' responses.

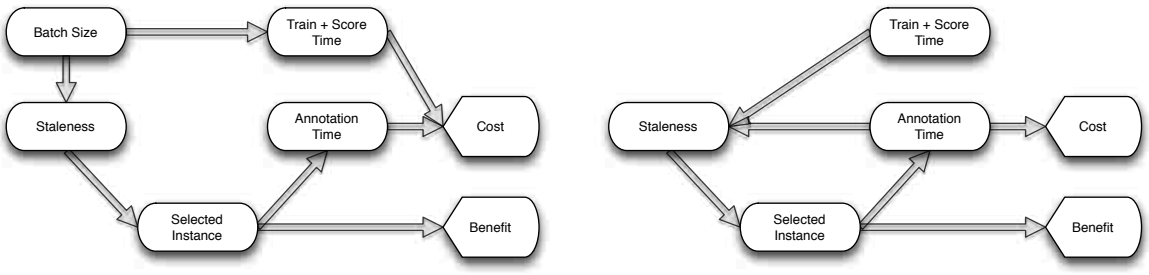
9.6 Addendum: Strengthening the Case for the Parallel Framework

The purpose of this addendum is to investigate the effect that changes to the relative amounts of time spent annotating versus training models and scoring instances have on performance. Various factors affect the performance of active learning algorithms; Figure 9.9 depicts these factors for **batch** and **parallel** during active learning. As can be seen in the figure, training and instance scoring times and annotation time have different effects on the cost and benefit in **batch** versus in **parallel**. Specifically, while training and instance scoring times directly affect cost in **batch**, in **parallel** they only affect staleness, which in turn determines the instance to be selected, which itself affects the annotation time. Additional pathways of interest in **parallel** not present in **batch** are the indirect influence of training, scoring, and annotation times on benefit.

This addendum is organized as follows: first, we analyze the effects that the relative time spent annotating has on performance in Section 9.6.1. We follow this up with experiments on real data in Section 9.6.2. Conclusions can be found in Section 9.6.3.

9.6.1 Analysis of Effects of Relative Time Spent Annotating on Performance

Certain effects of annotation, training, and scoring times can be explained analytically. As before, we assume that the annotator is continually and constantly requesting instances (the worst-case scenario for **parallel**). For the purposes of this analysis, we additionally assume that each annotation is equally costly to annotate. In this section, we let a represent the amount of time required to annotate an instance and let t represent the amount of time required to train all models and score all instances. In this analysis we examine the effect of decreasing a while holding t constant, first for **batch** then for **parallel**.



(a) **batch** (b) **parallel**

Figure 9.9: Factors affecting the performance of **batch** and **parallel**.

Batch

In **batch**, the average amount of time a user has to wait for an instance is the amount of time required to train models and score instances divided by the batch size, i.e., $\frac{t}{B}$. Therefore, the total cost is $\frac{t}{B} + a$. All else being equal, the total cost for an algorithm with no wait time (e.g., **random** and **parallel**) is a. Thus, the ratio of the total cost of **batch** to the total cost of a wait-free algorithm is a power law, $(\frac{\frac{t}{B}+a}{a})$.

Parallel

While changes in annotation time have a direct effect on cost in **batch**, the effect on cost in **parallel** is indirect (see Figure 9.9b). Instead, staleness is directly affected. When $a > t$, **parallel** will have time to train all models and score all instances while the annotator is still annotating. When the annotator finishes and requests the next instance, the staleness will therefore be one (the instance that the annotator has just annotated is the only source of staleness). Consequently, **parallel** will perform nearly identically to the optimal (w.r.t. staleness) **zerostale** (which always trains models and scores all instances using the latest annotation before serving the next instance to score, but usually at significant wait cost). On the other hand, if $a \leq t$, then instances will queue up. For example, if $\frac{t}{a}$ is just under three, then the annotator can annotate three instances before all of the remaining instances have been scored. The corresponding staleness of the instances will be one (**parallel** can never use the just annotated instance for scoring), two, three, and then back to one

again. In general, when annotation time is constant, the queue size (analogous to batch size) is t/a and leads to an average staleness of $\frac{\frac{t}{a}+1}{2}$. Thus, the ratio of staleness to annotation time is also a power law, but the actual effect of this staleness on cost will depend on the selection function, dataset, etc. In contrast, the average staleness of **batch** depends solely on the batch size, but the total cost is directly affected by annotation time.

Fortunately, under certain conditions, **parallel** can be guaranteed to never perform worse than **batch**. Although we leave a formal proof to future work, a basic sketch is as follows. Suppose we have a scoring function that is guaranteed to always rank instances at least as well as **random** on the basis of benefit and annotation cost (but *not* necessarily total cost including wait time). Then, no matter how stale instances are, we are guaranteed to be no worse than **random**. On the other hand, even with the same scoring function, **batch** will perform worse than **random** when the ratio of wait time to annotation time is sufficiently high since the wait time may negate any benefits gained by selecting instances in a better order than **random**.

This is a strong guarantee and a compelling reason to use **parallel**. However, there is no guarantee that **parallel** will outperform a well-tuned **batch** since **batch** has constant staleness while **parallel**'s staleness depends on the difference between the train + score times and annotation time. As a result, staleness may exceed that of **batch**. This will occur when training and/or scoring are particularly expensive, causing very stale scores. While these same conditions will certainly increase the waiting time of **batch**, the wait time is amortized across the entire batch. It is therefore conceivable that in some annotation projects, stale instances may be more detrimental to performance than an (amortized) increase in wait time. However, under such conditions it is particularly important to select an appropriate batch size: too high and the (noticeably detrimental) staleness increases; too low and the wait time increases, potentially risking performance degradation below that of random.

Of course, the guarantee that **parallel** does not degrade below **random** is predicated on a strong assumption. In practice, it is unknowable *a priori* whether an algorithm will produce a ranking that outperforms **random**. However, such a selection function is likely to affect **batch** very

similarly and will therefore also likely be worse than **random**, especially after incurring wait cost. Also note that active learning itself relies on a slightly weaker premise: that scoring functions do not produce rankings that outperform **random** on average. We hypothesize that, in practice, an algorithm that outperforms random without accounting for waiting cost (e.g., using **batch**) will be good enough to ensure that the results do not drop below the performance of **random** using **parallel**.

9.6.2 Empirical Comparison

To demonstrate the effects of annotation time on the algorithms, we run active learning simulations in which we fix the annotation time for each instance to 50, 100, 500, and 1000 seconds and compare the resultant cost-reduction curves of **batch** and **parallel** relative to **random**. In keeping with our previous experiments, we chose a batch size of 20 for **batch**. The results are in Figure 9.10.[3] Please note the change in scale between plots.

When annotation time is high (1,000 secs, i.e., over 17 minutes), both algorithms perform similarly well with **batch** being only negligibly affected by its small amount of staleness (9.5) compared to **parallel**'s (presumed) 1.0. As expected, the rate of degradation of **batch** is clearly a power law in these results. Although **parallel** does seem to be affected by the increase in staleness as annotation times decrease, unlike **batch**, its performance never degrades below that of **random**, as predicted in the previous section.

9.6.3 Conclusions

In this addendum, we have investigated the effect of the ratio of annotation time to training and scoring times on performance. The **batch** algorithm has the undesirable property that the ratio of total time of **batch** to algorithms with no wait time is a power law. In contrast, the effect on total time in **parallel** is indirectly determined by the staleness that decreasing annotation creates. While the ratio of staleness to annotation is also a power law, under some (strict) conditions, we know

[3] Special thanks to co-author Paul Felt who ran the experiments and provided the graphs.

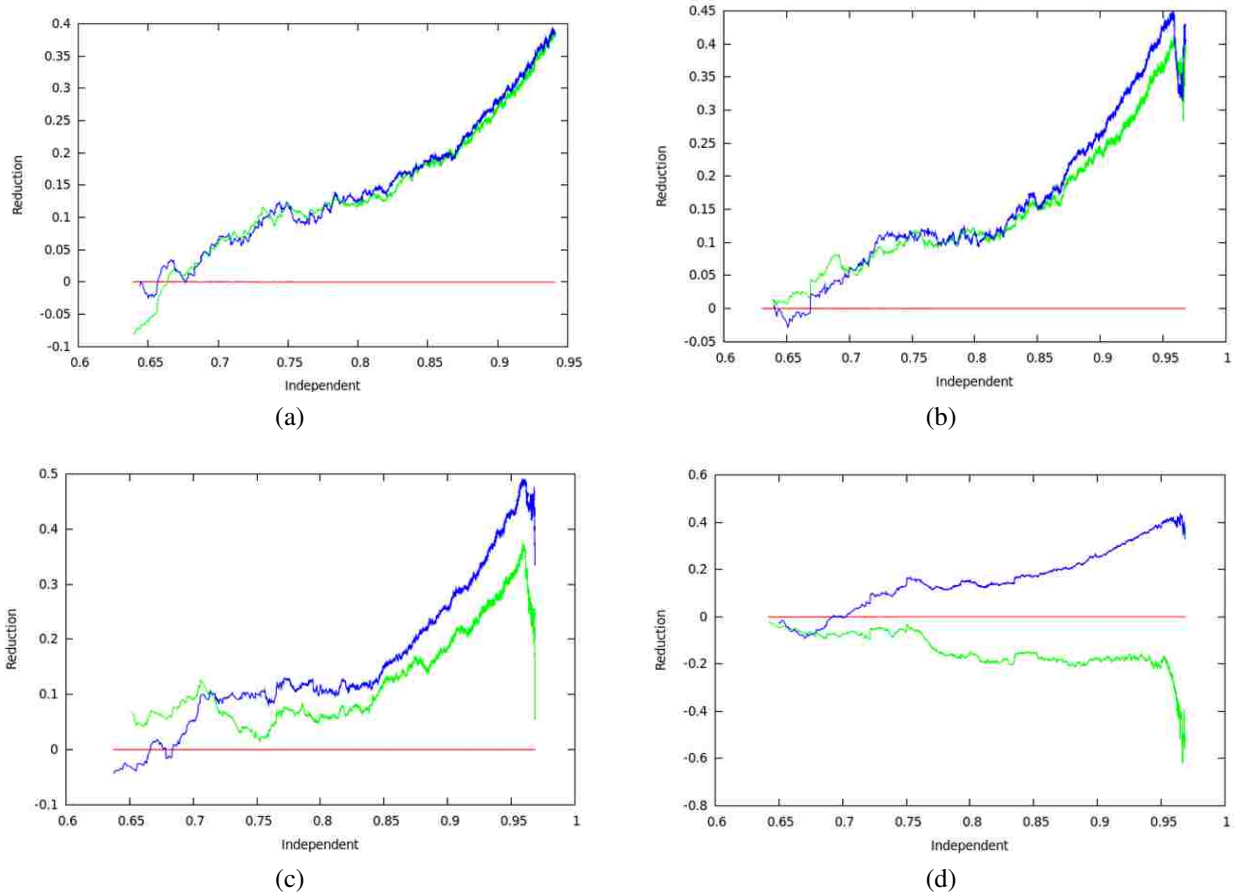


Figure 9.10: A comparison of **batch** (green) vs. **parallel** (blue) on the basis of percent reduction in cost relative to **random** (red) as a function of fixed per-instance annotation times: (a) 1,000 secs, (b) 500 secs, (c) 100 secs, and (d) 50 seconds.

that **parallel** will not perform worse than random. However, even when the conditions are not fully met, the performance of **batch** is also highly likely to suffer in similar ways. While a well-tuned **batch** may outperform **parallel**, tuning **batch** to maximize performance in this way is an unsolved problem. In fact, mistuning **batch** may cause performance to degrade below that of random. Thus, the likelihood that **batch** will outperform **parallel** is low and unknowable. This is why we expect practitioners to prefer **parallel** to **batch** moving forward.

Chapter 10

A Design for Multi-Annotator Active Learning

Abstract

Active learning in practice often involves multiple, noisy annotators. These annotations can be used to infer a ground truth annotation, learn each individual annotator's accuracy, and estimate the accuracy of the machine predicted annotations. In addition, existing methods for selecting instances for annotation are not always applicable or effective with multiple annotators. While previous work has examined some of these issues, no work has attempted to address them all—especially not with a unified approach. In this chapter, we describe a simple and intuitive Bayesian model of annotation that is designed to infer ground truth and estimate accuracies. We then describe how this model can be used in an active learning setting to select instances for annotation. We are particularly interested in the annotator-initiated case in which annotators request instances to annotate "on-demand". Our solution includes the use of return-on-investment to pick an annotator-specific instance for annotation.

10.1 Introduction

In this final chapter, we address the issue of active learning with multiple, fallible annotators with differing costs and accuracies. We are specifically interested in annotation environments in which annotators request instances to annotate on-demand, which we call annotator-initiated active learning (see Chapter 2). All known real-world annotation tasks operate in this manner. This chapter presents a detailed solution to this problem and describes a set of experiments designed to highlight the strengths and weaknesses of the approach.

In the context of active learning for annotation with multiple annotators in an annotator-initiated environment, we are interested in the following tasks:

- Inferring a "ground truth" annotation (i.e., selecting a final annotation for the corpus) in the presence of noisy annotators with differing accuracies and costs. We refer to this task as the **ground truth** task.
- Explicitly modeling and learning individual annotators' accuracies, which we term **annacc**.
- Estimating the accuracy of the machine predicted annotations on the unlabeled data. We call this task **machacc**.
- Selecting instances for annotators, in order to optimize the cost/benefit trade-off of obtaining annotations (instances and annotators are of varying costs and benefits). This is the **selection** task.

No previous work addresses the **machacc** task, nor does any work simultaneously address the other three.

This chapter is organized as follows. We first review previous work in Section 10.2. We then present our methods for addressing each of the aforementioned tasks using our Bayesian model and a novel selection algorithm in Section 10.3. We present some preliminary results for multi-annotator active learning in Section 10.4. Finally, conclusions and future work are discussed in Section 10.5.

10.2 Previous Work

There is a sufficient body of work surrounding the **ground truth** issue, especially in light of recent interest in crowd-sourcing; Sheng et al. [83] tackle both **ground truth** and **selection** tasks in the context of crowd-sourcing. In other particularly relevant work, Smyth et al. [84] applied a technique from Dawid and Skene [22] to infer the locations of volcanoes in surface photographs of Venus (**ground truth**); however, they were not concerned about the **selection** task and instead requested that all five annotators annotate all instances. There is a growing body of variations and extension to this simple "item-response" model to account for correlations among annotators or

item difficulty [12, 40, 53, 72, 95, 96]. This work addresses **ground truth** and **annacc**, but neither of the other two tasks.

The **ground truth** and **annacc** tasks can be seen as special cases of the fact-finding task [37, 65] in which the annotators are sources of information and their annotations are claims. However, the general task of fact-finding is much more broad, and existing approaches to solving the problem are generally overly complex for this simplified special case.

Donmez and Carbonell [24] propose a decision-theoretic framework specifically aimed at answering the **selection** task. Their framework is capable of handling noisy annotators with different accuracies and costs, but in order to model annotators' accuracies, they spend a portion of the budget on an initial exploration stage in order to estimate annotator accuracy; this initial estimate is not updated during active learning, thus not truly addressing the issue of **annacc**. Furthermore, they do not allow for redundant annotations: a single annotator is selected per instance and this noisy annotation is assumed to be the ground truth, which does not directly address the **ground truth** task. In a follow-up paper [25], the authors recognize the importance of redundant annotations. They use active learning to select an instance, then use a reinforcement learning-inspired approach to select the most accurate annotators for that instance in such a way that they are able to effectively balance learning annotator accuracies and exploiting the most accurate annotators. They infer ground truth via a majority vote. Perhaps the most problematic aspect of the paper is that it does not extend to the more realistic annotator-initiated environment.

10.3 Methods

In this section, we describe our novel approach to the four tasks identified in the introduction to this chapter. We begin by presenting a Bayesian model that can infer ground truth from multiple, noisy annotators; infer the accuracy of each annotator; and predict the accuracy of the model on the unlabeled portion of a corpus without any held-out data. We then describe how this model can be used to select instances in an annotator-initiated environment.

10.3.1 Model

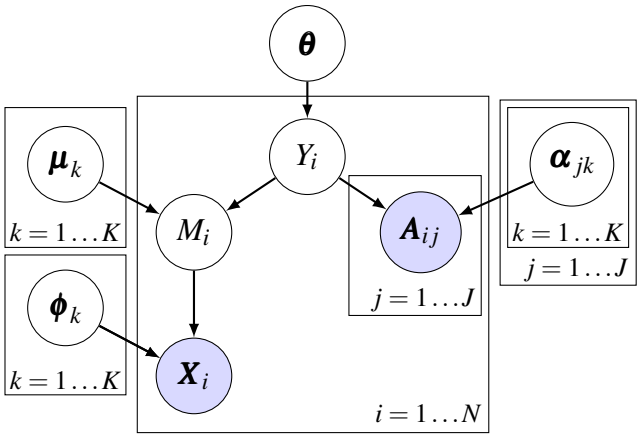
Given a set of instances and the annotations for these instances (each instance may have zero or more annotations), we are interested in the task of inferring a ground truth label, each annotator's accuracy, as well as the accuracy of the machine on the data that lacks annotations (without holding out any data).

The model that we will use to address the first three tasks mentioned in the introduction (i.e., everything but the **selection** task) is based on a Bayesian model we proposed in previous work [13, 14], but which we extend and more fully develop in this work. The model is based on three main principles: (1) the true labels are unobservable, (2) the machine itself should be considered an annotator, and (3) all annotations may carry useful information. We additionally assume that annotations are independent of one another given the true (unobservable) label. Since we do not explicitly exclude multiple annotations from the same annotator on the same instance, this independence assumptions extends to such annotations.

Figure 10.1 presents the model. We abbreviate the Dirichlet distribution as *Dir*, the symmetric Dirichlet distribution as *SymDir*, the categorical distribution (i.e., a multinomial distribution with size parameter equal to one) as *Cat*, and the multinomial distribution as *Mulitnom*, parameterized by size and probabilities. In addition, $|\cdot|_1$ denotes the L_1-norm. The relevant constants for the model are: J is the number of annotators, K is the number of labels, N is the number of instances, and F is the number of features.

In addition, we adopt the following notational conventions. In statistics, random variables are typically rendered in capital (Roman) letters, while values that they can assume are presented in lowercase, e.g., $Y_i = y_i$. However, following common practice, we are sloppy with the distinction, typically preferring the lowercase letters for both so long as context allows for proper disambiguation. We also use lowercase p to represent both pmfs and pdfs.

We assume that for the i^{th} data instance $\boldsymbol{x}_i$, the annotations $\boldsymbol{a}_{ij}$ are conditionally i.i.d. of each other and the instance ($\boldsymbol{x}_i$) given the (unknown) label y_i, whose *a priori* distribution is $\boldsymbol{\theta}$. Therefore, the annotations for each instance can be represented as count vectors for each annotator, i.e., $\boldsymbol{a}_{ij}$ is a



$$
\begin{aligned}
\boldsymbol{\theta} &\sim SymDir(b_\theta), & dim(\boldsymbol{\theta}) &= K \\
\boldsymbol{\mu}_k &\sim Dir(\boldsymbol{b}_{\mu_k}), & dim(\boldsymbol{\mu}_k) &= K \\
\boldsymbol{\phi}_k &\sim SymDir(b_\phi), & dim(\boldsymbol{\phi}_k) &= F \\
\boldsymbol{\alpha}_{jk} &\sim Dir(\boldsymbol{b}_{\alpha_{jk}}), & dim(\boldsymbol{\alpha}_{jk}) &= K \\
y_i|\boldsymbol{\theta} &\sim Cat(\boldsymbol{\theta}), & y_i &\in \{1 \ldots K\} \\
m_i|y_i, \boldsymbol{\mu} &\sim Cat(\boldsymbol{\mu}_{y_i}), & m_i &\in \{1 \ldots K\} \\
\boldsymbol{x}_i|m_i, \boldsymbol{\phi} &\sim Multinom(|\boldsymbol{x}_i|_1, \boldsymbol{\phi}_{m_i}), & dim(\boldsymbol{x}_i) &= F \\
\boldsymbol{a}_{ij}|y_i, \boldsymbol{\alpha}_j &\sim Multinom(|\boldsymbol{a}_{ij}|_1, \boldsymbol{\alpha}_{j,y_i}), & dim(\boldsymbol{a}_{ij}) &= K
\end{aligned}
$$

Figure 10.1: Generative Bayesian model for inferring ground truth from multiple annotators while modeling their individual accuracies.

vector of length K that holds the counts of the number of times annotator j labeled instance i with each distinct label (typically, the vector is all zeroes—no annotations—or all zeroes with a single one, but the model does not explicitly disallow the case where an annotator provides more than one annotation for the same instance).

Imagine that, in addition to the human annotations, we were to use an off-the-shelf classifier to produce machine predictions from each of the raw (unlabeled) data instances, $\boldsymbol{x}_i$. We could then treat this prediction as yet another label from an imperfect annotator. In fact, we could use this annotation on instances for which no human annotations are available. In our model, each m_i roughly represents this notion of a features-only, machine annotation, which contrasts with the ground truth label, y_i.

We would also like to capture the accuracy of each annotator and the machine (recall that we essentially treat the machine as another annotator). Each of the $\boldsymbol{\alpha}_j$ represents the "label confusion matrix" for the j^{th} annotator, i.e., the probability that the j^{th} annotator will select annotation k' when the true label is k. The machine's accuracy is similarly encapsulated by $\boldsymbol{\mu}$.

This model can be seen as a conglomeration of ideas from previous work. First, if we consider only the nodes M and X, then we have a class-conditional classifier (e.g., Naïve Bayes); including priors over $\boldsymbol{\mu}$ and $\boldsymbol{\phi}$ produces the Bayesian extension (c.f. Meilă and Heckerman [60], Walker and Ringger [94]). This class of model, of course, is not capable of directly incorporating multiple annotations. Item response models, such as the model proposed by Dawid and Skene [22], do take into account the annotations A, but they do not consider the machine as an annotator; consequently, Y is the only other random variable in their model (inference is done in a frequentist setting where $\boldsymbol{\theta}$ and $\boldsymbol{\alpha}$ are not random variables, but constants to be learned). Smyth et al. [84] augment David and Skene's model to incorporate the features $\boldsymbol{X}$. However, they ignore M; instead, $\boldsymbol{X}$ depends directly on Y. This prevents them from learning the accuracy of the model. Furthermore, they, too, take a frequentist approach and therefore treat $\boldsymbol{\phi}$, $\boldsymbol{\theta}$, and $\boldsymbol{\alpha}$ as constants. Carpenter [12] proposes a model that is essentially a Bayesian version of Dawid and Skene's model [22]; Pasternack and Roth [67] also present essentially that same model. Carroll et al. [14] propose a similar Bayesian model that accounts for the data $\boldsymbol{X}$ and makes the same independence assumptions about Y and $\boldsymbol{A}$. However, their model does not treat the machine as an annotator (our M) and their model of $\boldsymbol{X}, Y$ is conditional, not generative.

10.3.2 Inference

In our generative Bayesian model that includes all of the needed data and hidden variables, the solution to the first three tasks from the introduction (i.e., **ground truth**, **annacc**, and **machacc**) become simply a matter of posterior inference. To select a label for the i^{th} instance, we simply find the most probable label given all of the observations (across all instances), i.e., $\arg\max_y p(Y_i = y|\boldsymbol{a}, \boldsymbol{x})$. Similarly, the distribution over accuracies for annotator j is simply the posterior distribution

of $\boldsymbol{\alpha}_j$ given the observations, i.e., $p(\boldsymbol{\alpha}_j|\boldsymbol{a},\boldsymbol{x})$. Although these distributions cannot be computed analytically, approximate inference techniques such as Gibbs sampling or variational inference can be employed to approximate the distributions.

In this work, we construct a Gibbs sampler for inference. Liu [54] provides both empirical and theoretical evidence supporting the superiority of collapsed and block samplers to traditional Gibbs sampler. Thus, we analytically integrate out all of the parameters of the model ($\boldsymbol{\theta}$, $\boldsymbol{\mu}$, $\boldsymbol{\phi}$, $\boldsymbol{\alpha}$) to construct a collapsed sampler and we jointly sample y_i and m_i. These variables, in particular, are highly correlated and stand to benefit most from blocking. Since each sample from a block of categorical variables requires time exponential in the number of variables (assuming the same size domain), these pairs are the only variables we sample in blocks.

The derivation of the complete conditionals (the probability of y_i and m_i given all other random variables to be sampled) is in Appendix B; here we simply present the distributions.

For notational simplicity, we define the following count variables, which are disambiguated by the letters of their subscripts. For the purpose of the complete conditionals, each count variable excludes the counts associated with the instance corresponding to the complete conditional (indexed by i):

$$n^{\theta}_{k} = \sum_{i' \neq i}^{N} \mathbb{1}(y_{i'} = k)$$

$$n^{\mu}_{kk'} = \sum_{i' \neq i}^{N} \mathbb{1}(y_{i'} = k \wedge m_{i'} = k')$$

$$n^{\alpha}_{jkk'} = \sum_{i' \neq i}^{N} a_{i'jk'}^{\mathbf{1}(y_{i'}=k)}$$

$$n^{\phi}_{kf} = \sum_{i' \neq i}^{N} x_{i'f}^{\mathbf{1}(m_{i'}=j)}.$$

That is, n^{θ}_{k} is the number of instances currently labeled k; $n^{\mu}_{kk'}$ is the number of instances labeled k that have a features-based prediction (i.e., r.v. M) of k'; $n^{\alpha}_{jkk'}$ is the number of times that annotator j

chose annotation k' on instances labeled k; and n^{ϕ}_{kf} is the number of times feature f occurs with instances having a feature-based prediction of k.

Using these count variables, the complete conditionals are:

$$
\begin{aligned}
[Y_i = c, M_i = d] \propto (b_\theta + n^{\theta}_{c}) \frac{1}{\prod_{k'=1}^{K} \Gamma(b_\mu + n^{\mu}_{ck'})} (b_\mu + n^{\mu}_{cd}) \frac{1}{\sum_{k'=1}^{K} (b_\mu + n^{\mu}_{ck'})} \\
\cdot \prod_{j=1}^{J} \frac{1}{\left(\sum_{k'=1}^{K} b_{\alpha_j} + n^{\alpha}_{jck'}\right)^{(\sum_{k'=1}^{K} a_{ijk'})}} \prod_{k'=1}^{K} \left(b_{\alpha_j} + n^{\alpha}_{jck'}\right)^{(\sum_{k'=1}^{K} a_{ijk'})} \\
\cdot \frac{1}{\left(\sum_{f=1}^{F} \left(b_\phi + n^{\phi}_{df}\right)\right)^{(\sum_{f=1}^{F} x_{if})}} \prod_{f=1}^{F} \left(b_\phi + n^{\phi}_{df}\right)^{(\sum_{f=1}^{F} x_{if})}
\end{aligned}
\tag{10.1}
$$

where the notation $x^{(n)}$ represents the rising factorial defined as $x^{(n)} \stackrel{def}{=} x(x+1)(x+2)\ldots(x+n-1)$.

10.3.3 Instance Selection

To address the problem of instance selection with multiple annotators in an annotator-initiated environment (i.e., **selection**), we extend ROI-based active learning to the multi-annotator case. Specifically, we maintain a priority queue of instances for each annotator in which the priority is a ROI-score computed from the expected benefit and cost of employing a particular annotator on a particular instance. Expected benefit can be approximated using samples from the posterior obtained from Gibbs sampling. Specifically, given S samples of y_i from the posterior, each sample denoted as $y_i^{(s)}$, the expected benefit (denoted as function $b(\cdot)$) is:

$$
\mathbb{E}\left[b(y_i|\mathbf{x},\mathbf{a})\right] \approx \frac{1}{S} \sum_{s=1}^{S} b(y_i^{(s)}|\mathbf{x},\mathbf{a}). \tag{10.2}
$$

In the limit as S approaches infinity, the approximation approaches the true value of the expectation [30].

This scheme allows for different combinations of annotators and instances to result in different expected benefits and costs, as desired. Similar to our analysis of ROI in Chapter 8, this

approach should (greedily) maximize the performance on a cost/benefit curve. We will employ our parallel framework [35] to constantly update each annotator's priority queue while annotators are annotating.

10.4 Preliminary Results

In this section, we present the results from some simple experiments similar to those performed by Sheng et al. [83]. In these experiments, we use their generalized round robin (GRR) approach. Each round, we select an instance at random (without replacement) and then annotate the instance K separate times. We repeat this process until all of the instances have been annotated, and then repeat. Thus, after R rounds, each instance will have $K \cdot R$ annotations. Annotators are allowed to annotate each instance more than once (including, more than once per round, and of course, more than once across rounds). We try values of $K = 1 \ldots 9$ corresponding to each sub-figure in Figure 10.2. The experiments are run using a modification of the annotator-initiated environment explained in Chapter 9. This can be seen as the multi-annotator generalization of the random baselines presented in standard active learning. Note, however, that these results are obtained in an annotator-initiated environment with an emphasis on transductive learning. That is, we are mainly interested in the overall accuracy of both labeled and unlabeled data—more so than the accuracy on held-out data, though we examine all aspects of accuracy.

These experiments simulate three annotators with accuracies 0.7, 0.8, and 0.9, with mean annotation times of 35, 30, and 25 seconds, respectively. The dataset is a reduced sample of the 20 newsgroups data set [44] that contains 400 documents from 4 different classes. Each experiment is averaged over ten different random runs.

While Sheng et al. [83] focus on held-out accuracy in their experiments, we are primarily interested in total accuracy of the corpus for the purposes of corpus annotation. It is also of interest to examine the component parts of the total accuracy: the accuracy over instances having no annotations ('unlabeled accuracy') and the accuracy over instances having at least one annotation ('labeled accuracy'). We also track the held-out accuracy.

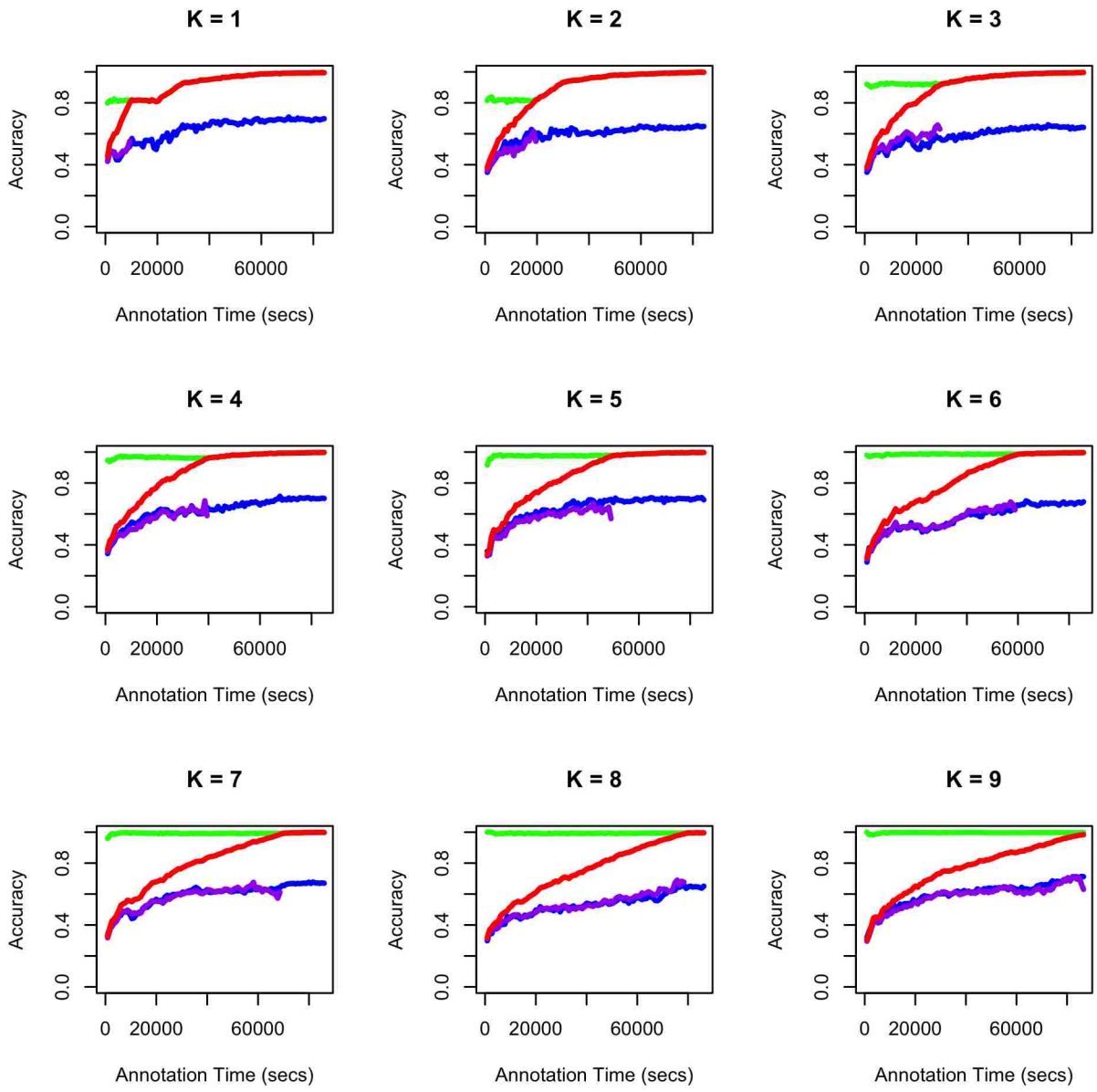


Figure 10.2: Results of GRR experiments with different K. The legend is as follows: green, accuracy of data with at least one label; purple: accuracy of data without any labels; red: combined accuracy of labeled/unlabeled; blue: held-out accuracy

As can be seen in Figure 10.2, labeled accuracy, and therefore overall accuracy, approaches 1.0 as the number of annotations increases. For this dataset and these annotators, accuracy approaches 1.0 more quickly with lower K, suggesting that the best strategy may be to singly label as many instances as possible.

Unlabeled accuracy tracks held-out accuracy almost perfectly; the maximum acheivable accuracy on this dataset with a naïve Bayes model is around 60%. This is expected since GRR draws instances at random; this may not be the case for non-random selection strategies such as the one described in Section 10.3.3.

Finally, note that held-out accuracy *may* be rising slightly quicker for larger K. If the goal is a good inductive model, it may be advantageous to annotate the same instance multiple times before moving on to the next (e.g., adjudicated annotation).

10.5 Conclusions and Future Work

To begin this chapter, we identified four important tasks inherent with active learning with multiple, noisy annotators. We then presented a single, unified Bayesian model that is capable of inferring ground truth from multiple annotations, inferring each annotator's accuracy, and predicting its own accuracy on unlabeled instances (the first three tasks). In addition, we have shown how the model can be used as a benefit estimator in ROI-based active learning. The result is a model that is potentially capable of reducing annotation costs in the presence of multiple noisy annotators.

While this chapter has presented the detailed design of this Bayesian model and its use in active learning, experimentation will be necessary in future work to validate these approaches. The first step in validating the approach should involve comparing the ability of the model to infer ground truth. This can be done using GRR (mentioned in the previous section) to produce learning curves that measure the accuracy of the inferred annotations. A similar curve can be produced using a simple majority vote scheme for comparison. Additionally, a comparison can be made to the most prevalent method in actual annotation projects: adjudicated annotation, in which each annotation is

independently annotated by two annotators, and in case of disagreement an adjudicator makes the final decision.

In conjunction with the previous experiments, the quality of the model's estimate can also be quantified by comparing the model's estimate to the parameter used to generate the dataset and computing the root mean-squared error across all annotators. As a reasonable baseline for comparison, one might first inferring ground truth and then computing the percentage of times that each annotator agreed with the ground truth. Similarly, the accuracy of the model's estimate of held-out accuracy can be compared to that of the actual accuracy.

The proposed method for selecting instances using the Bayesian model can be validated by comparing cost/benefit curves using this approach to some baselines. One baseline is the aforementioned GRR [83]. Higher bars for comparison are the label uncertainty (LU), model uncertainty (MU), and combined label and model uncertainty (LMU) approaches of Sheng et al. [83]. LU is not as easily computed in the multi-class case as it is for binary classification; however, Monte Carlo sampling can be used to estimate the same quantity. For MU, we use an ensemble of Naïve Bayes classifiers (c.f. Roy and McCallum [75]) rather than a random forest.

The problem of inferring ground truth from noisy labels has been receiving increased attention recently. Although most of the models are essentially extensions and variations on the item-response model (as is ours), many of these extensions could be incorporated into our model, for instance, item difficulty, which serves to create some correlation between annotators [12, 40, 53, 72, 95, 96]. Also, related work in the field of fact-finding deals with truthfulness and bias of information [66], which could not only provide useful statistics about annotators and their annotations, but also improve instance selection.

Chapter 11

Conclusions and Future Work

Annotation projects such as the one that the Maxwell Institute is undertaking for the Syriac project (see Chapter 1) are being initiated at an unprecedented rate, presumably due to the corresponding technological explosion which has provided the tools, hardware, bandwidth, and storage needed by these products at (large) scale. Yet the number of projects is not the only thing growing rapidly. In the age of "big data," raw corpora are so immense that complete manual annotation is simply impossible. However, a feasible portion of a corpus may be manually annotated and the rest of the corpus automatically annotated using machine learning methods trained on the hand annotated data. Although the labor to undertake such projects can be costly, the majority of annotation projects decide not to use active learning as a cost-saving approach (see Tomanek and Olsson [90]). The biggest reason for their aversion to active learning is a lack of confidence in the technique for reducing annotation costs in practice. This dissertation has begun to address these concerns by developing new methods for active learning that reduce annotation costs in more realistic environments.

One mismatch between research and practice is historical in nature. Active learning was initially proposed as a learning protocol, i.e., as a means of training a model (it was inductive, in the terminology of Chapter 2) using fewer instances. With the increase in costly annotation projects, active learning came to be seen as a potential means of producing annotated corpora at lower overall cost. Chapter 4 is the first to explicitly propose the use of transductive active learning for annotation purposes: active learning is used to obtain a high quality model at lower cost (as in previous work), which is subsequently used to annotate the remainder of the corpus. On the surface, this suggestion

appears to be minor and almost obvious. However, this simple suggestion allows corpus creators to annotate even more data than their budget allows, carefully balancing the need for more data with the need for higher quality annotations. It also suggests that the best way to evaluate transductive active learning is to measure the overall accuracy of a corpus (both labeled and unlabeled data). This metric is part of the design presented in Chapter 10.

Frequently, fewer instances were implicitly equated with lower cost. Clearly, this is not so, at least for structured learning problems as Chapter 6 clearly demonstrates: a technique may pick a small number of information-rich instances that are very expensive to annotate (see Figure 6.1). In order to perform more realistic simulations, it is necessary to have a realistic model of cost that can be used to evaluate the cost of annotating each instance. Chapter 5 creates one such model from data obtained from a user study.

In addition, cost-conscious algorithms such as ROI require an explicit model of cost. Thus, our model of cost was used to select instances using cost-conscious algorithms. In Chapter 7, we used the coefficients that were inferred from the user study in Chapter 5. In other words, we treated cost as if it were fixed and known. As Settles et al. [81] point out, such costs are not realistically known *a priori*. Thus, in Chapters 8 and 10, we learn the coefficients to the model during the active learning process, following Settles et al. [81].

An important, though not particularly surprising finding from Chapter 6 is that cost-conscious algorithms outperform non-cost-conscious algorithms. Nevertheless, even non-cost-conscious algorithms often out-performed the random baseline and reduced the cost of annotation. This reduction is due to the fact that non-cost-conscious algorithms have an implicit cost bias. The advantage of algorithms that explicitly model cost is that they should continue to work successfully even when the cost changes—so long as the new cost is correctly modeled. This flexibility is an important step towards producing credible results for practitioners.

However, explicitly modeling cost necessarily couples the results of a method to a particular cost measure, effectively invalidating the results for tasks for which the cost model does not apply. This limitation essentially renders the results useless in practice. Furthermore, active learning has

traditionally been based on more generally applicable principles that transcend a given problem, model, or dataset. For instance, in transductive active learning where the machine is used to label instances that were not labeled by annotators, such automatically produced labels are most likely to be wrong for instances whose label is least certain, lending credence to the query-by-uncertainty benefit estimator. Note that this principle is true irrespective of project, data set, output space, etc.

Chapters 7 & 8 reconcile this conflict between the imperative need for evaluating active learning using a realistic cost and results that are generally applicable. The solution is simple, yet effective: simply provide an estimate of benefit (e.g., based on the uncertainty principle mentioned earlier) and divide by an estimate of cost. Simple as it may be, Chapter 8 provides some theoretical motivation for return-on-investment-based active learning. Essentially, assuming reasonably accurate estimators of benefit and cost, instances with the highest return-on-investment—or, colloquially, "bang-per-buck"—will greedily produce results that tend to maximize benefit for a given cost. We show this approach to be successful in practical settings, and the theory gives some assurances as to the applicability more generally.

Another contribution of this dissertation has been the recognition of annotator-initiated active learning, in which the annotator requests work from the computer "on demand," as opposed to learner-initiated active learning, wherein the computer sends work to the annotators at its leisure. To the best of our knowledge, annotator-initiated annotation is nearly ubiquitous in current annotation practice. Yet all previous research of which we are aware can be seen as learner-initiated.

While one approach is not necessarily better than the other, research up until this point has ignored the costs associated with the learner-initiated approach, e.g., the opportunity cost related to the time spent waiting for an annotator to begin work or the cost of paying an annotator to wait for the next instance. Chapter 9 empirically demonstrates the danger of ignoring the costs of an otherwise successful learner-initiated approach (like those seen in research): once accounted for, the result can be worse than not using active learning. This pitfall is another example of where research has ignored a cost, contributing to the mistrust that practitioners have of active learning.

Ideally, annotators will not have to wait for instances in annotator-initiated environments as doing so increases the cost of annotation. While annotators are not always paid by the hour, having to wait for instances directly affects the amount of work they can accomplish and therefore affects the pay they are willing to work for. Chapter 9 presents a simple solution: rather than computing the next best instance upon request, the computation occurs while the annotator is annotating (and, in fact, continuously). Chapters 8 & 9 found this technique to perform well for a variety of selection techniques. Based on intuition and our empirical observations, we hypothesize that "reasonable" selection functions will not perform worse than random if the same function does not perform worse than random using learner-initiated approaches. In short, the no-wait framework of Chapter 9 helps bridge the gap between active learning in research and active learning used in practice. We strongly recommend that it be used in all future research using active learning techniques.

Most annotation projects involve multiple annotators. The solution we detail in Chapter 10 combines ideas from previous work in the space into a single, simple Bayesian model. In essence, the model uses the output of a machine learning model as another annotator and combines the results of all annotators (including the machine's prediction) in order to infer a "true" annotation for each instance, the accuracy of each annotator, and the accuracy of the predictions on the unlabeled portion of the corpus. We also detail how to use this model for selecting instances in ROI-based active learning. The result is a simple yet elegant solution for annotator-initiated projects with multiple annotators.

While this work was focused on the *creation* of annotated corpora, the larger field of machine learning can be seen as a consumer of such corpora. As such, this work may have implications more generally to the field of machine learning. For example, by freeing up monetary resources, we may expect to see a larger number of corpora produced. Similarly, by reducing the cost of producing labeled data, it may be possible to produce datasets that were otherwise prohibitively costly.

In conclusion, the techniques described herein have been shown to work in realistic settings; they reduce the total cost of producing annotated corpora in realistic environments. As a result of

this dissertation, annotation projects should have growing confidence that active learning will not only help reduce the cost of annotating its corpus, but is necessary to work at the scale they need.

11.1 Future Work

Despite the steps made in this dissertation towards bridging the gap between research and practice, much is left to be done. Each of the individual chapters are peppered with suggestions about items that can be done. In this section, we emphasize a few additional avenues of future research.

Since the experiments in this dissertation necessitated a cost model, which itself required a non-trivial user study, the work herein was focused on a single task and dataset. Although there are good reasons to believe these results will hold more generally based on related work and our theoretical analyses, future work should demonstrate empirically that such is the case. Along the way, new problems—and solutions—are bound to surface.

Even though the techniques explicated in this dissertation are more realistic than previous research, they have yet to be evaluated in real time on a real project. The interactive nature of active learning makes obtaining repeated results (required for statistical significance) prohibitively expensive, and certainly infeasible for the number of experiments performed in this work. However, a carefully designed comparison between a ROI-based approach in the no-wait framework compared to random selection on an interesting task will further instill confidence in would-be adopters of active learning for annotation.

The generic nature of ROI is simultaneously a strength and a weakness. The basic requirement for ROI is that the implementer use a reasonably accurate estimator of cost and benefit. Chapter 8 showed that the bar is somewhat low for "reasonably accurate"; nevertheless, it is entirely possible to use a poor benefit estimator that leads to worse-than-random results. For simple (i.e., non-structured) tasks, a greedy-EVSI approach to benefit estimation may be feasible (e.g., Roy and McCallum [75]). However, more research is needed in the area of structured learning where such an approach is strictly intractable. It remains to be seen whether sampling can be done efficiently

and effectively. Alternatively, benefit estimates may be able to be learned, much like we showed it possible to learn cost estimates in Chapter 8.

Finally, in Chapter 10, we presented a design for active learning with multiple annotators and suggested several methods for validating the approach that should be carried out in future work. In addition, we hinted at the possibility of using benefit and cost functions that are annotator-dependent. While we did use annotator-specific cost functions, our benefit function did not directly take into account annotator-specific information (e.g., accuracy). Using a per-annotator greedy-EVSI approach is promising, albeit expensive, but tractable—particularly if sampling is used [75].

Appendix A

Efficient Computation of Expectations Using Latent Variable Markov Models

Abstract

Latent variable Markov models (LVMMs), which include hidden Markov models, conditional Markov models, maximum entropy Markov models, and conditional random fields, are an important class of models that are used for sequence modeling problems such as named-entity recognition, segmentation, part-of-speech tagging, speech recognition, gene prediction, etc. In scenarios such as active and semi-supervised learning, it is often necessary to compute expectations such as entropy, utility, accuracy, etc. using these models. We show how two general classes of expectations for LVMMs can be computed efficiently using dynamic programming, even when constraining the expectations. When used to derive entropy for CRFs, the resulting dynamic programming algorithm requires half as many passes as a previous solution built specifically for computing entropy.

A.1 Introduction

Latent variable Markov models (LVMMs) are graphical models in which the latent variables $\mathbf{y}$ form a chain wherein each y_t is directly connected only to its predecessor y_{t-1} (higher order Markov models can be converted to order-1 Markov models). Examples of such models include hidden Markov models (HMMs), conditional Markov models (such as a maximum entropy Markov model), and linear-chain CRFs. Such models are useful for many sequence modeling problems including named-entity recognition, segmentation, part-of-speech tagging, speech recognition, gene prediction, etc. Typically, the distribution of interest is the conditional distribution $p(\mathbf{y}|\mathbf{x})$, where the $\mathbf{x}$ are the known data, usually one data point per latent variable. The conditional distribution of

LVMMs is in fact that of a linear-chain CRF:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, \mathbf{x}),$$

where the ψ_t are potential functions, and $Z(\mathbf{x})$ is the partition function (for a review of CRFs, see Sutton and McCallum [85]). An LVMM is an HMM if $\psi_t(y_t, y_{t-1}, \mathbf{x}) = p(x_t|y_t)p(y_t|y_{t-1})$, in which case $Z(\mathbf{x}) = p(\mathbf{x})$; an LVMM is a CMM if $\psi(y_t, y_{t-1}, \mathbf{x}) = p(y_t|y_{t-1}, \mathbf{x})$, in which case $Z(\mathbf{x}) = 1$.

It is frequently necessary to compute expectations of functions of the latent variables $\mathbf{y}$, given the data $\mathbf{x}$, i.e.:

$$\mathbb{E}_{\mathbf{Y}|\mathbf{x}}[f(\mathbf{y}, \mathbf{x})] = \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) f(\mathbf{y}, \mathbf{x}).$$

One example of such an expectation for sequences is entropy, which is useful for computing uncertainty in active learning [11, 80] or performing semi-supervised learning in linear-chain CRFs [55]. Another common function that is computed in expectation is utility (such as expected accuracy), which can be useful in active learning [36, 75].

Unfortunately, naïve computation of such expectations is exponential in the length of the sequence, in general. Previous work (see below) has shown that entropy can be computed in time linear in the length of the sequence using dynamic programming. In this work, we more generally show that expectations of certain classes of functions can be computed in linear time. Specifically, we investigate the cases where the function to be expected over is either a product or a sum of functions of each individual latent variable y_t and its predecessor y_{t-1}. To the best of our knowledge, this work represents the first attempt at presenting an efficient algorithm for expectations of general classes of functions for LVMMs. When our general approach is applied to derive a dynamic programming solution to computing entropy for CRFs, the result requires half as many passes of dynamic programming as a previously proposed solution built specifically for computing entropy.

A.2 Previous Work

Previous work has examined efficient methods specifically for computing entropy in LVMMs: Hernando et al. [38] investigated the case for HMMs, Busby and Ringger [11] independently derived the same for CMMs, and Mann and McCallum [55] present a solution for CRFs in the context of semi-supervised learning using entropy regularization. The latter was also adapted to allow for the entropy of a sequence, holding constant some of the latent variables. Although she uses a different, more complex model, Hwa [43] presents an algorithm for the efficient calculation of tree entropy using an inside-outside-like algorithm.

A.3 Forward Probability

We begin with a brief review of forward probability, which is defined for HMMs as $\alpha_t(y_t) \stackrel{def}{=} p(\mathbf{x}_{\langle 1\ldots t\rangle}, y_t)$. Although with different semantics, the forward variables can be defined more generally for linear-chain CRFs as:

$$\begin{aligned}
\alpha_t(y_t) &\stackrel{def}{=} \sum_{\mathbf{y}_{\langle 1\ldots t-1\rangle}} \prod_{t'=1}^{t} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) \\
&= \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) \cdot \sum_{\mathbf{y}_{\langle 1\ldots t-2\rangle}} \prod_{t'=1}^{t-1} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) \\
&= \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) \alpha_{t-1}(y_{t-1}), \qquad \text{(A.1)}
\end{aligned}$$

with a base case of $\alpha_1(y_1) = \psi_1(y_1, y_0, x_1)$ (y_0 is the initial state). This recurrence relation can elegantly and efficiently be implemented using dynamic programming (c.f. Rabiner and Juang [68]).

Forward probability is needed to compute the normalizing constant $Z(x) = \sum_{y_T} \alpha_T(y_T)$ in CRFs, which, as previously noted, is $p(\mathbf{x})$ in the case of HMMs. It also is needed to compute the marginal probabilities $p(y_t|\mathbf{x})$.

A.4 Product of Functions

Suppose the function $f(\cdot)$ to be expected over has the form

$$f(\mathbf{y},\mathbf{x}) = \prod_{t}^{T} g(y_t, y_{t-1}, \mathbf{x}),$$

for some function $g(\cdot)$ (for brevity, we hereafter omit the $\mathbf{x}$). The expectation is:

$$\begin{aligned}
\mathbb{E}_{\mathbf{Y}|\mathbf{x}}[f(\mathbf{y})] &= \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) f(\mathbf{y}) \\
&= \sum_{\mathbf{y}} \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, \mathbf{x}) \cdot \prod_{t=1}^{T} g(y_t, y_{t-1}) \\
&= \frac{1}{Z(\mathbf{x})} \sum_{\mathbf{y}} \prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, \mathbf{x}) \cdot g(y_t, y_{t-1}). \qquad \text{(A.2)}
\end{aligned}$$

Now, let

$$\pi_t(y_t) \stackrel{def}{=} \sum_{\mathbf{y}_{\langle 1\ldots t-1\rangle}} \prod_{t'=1}^{t} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) g(y_{t'}, y_{t'-1}). \qquad \text{(A.3)}$$

Comparing equations A.2 and A.3, we can see that

$$\mathbb{E}_{\mathbf{Y}|\mathbf{x}}[f(\mathbf{y})] = \frac{1}{Z(\mathbf{x})} \sum_{y_T} \pi_T(y_T). \qquad \text{(A.4)}$$

A recurrence relation can be established from equation A.3:

$$\begin{aligned}
\pi_t(y_t) &= \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) g(y_t, y_{t-1}) \\
&\quad \cdot \sum_{\mathbf{y}_{\langle 1\ldots t-2\rangle}} \prod_{t'=1}^{t-1} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) g(y_{t'}, y_{t'-1}) \\
&= \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) g(y_t, y_{t-1}) \pi_{t-1}(y_{t-1}),
\end{aligned}$$

with a base case of $\pi_{t-1} = 1$. From this relation, a linear-time dynamic programming solution analogous to the forward algorithm can be used to compute the expectation. Note that the forward algorithm is needed to compute $Z(\mathbf{x})$ for final computation of equation A.4.

A.5 Sum of Functions

Suppose the function $f(\cdot)$ to be expected over has the form

$$f(\mathbf{y}, \mathbf{x}) = \sum_t g(y_t, y_{t-1}, \mathbf{x}), \tag{A.5}$$

for some function $g(\cdot)$ (for brevity, we once again omit the $\mathbf{x}$ hereafter). The expectation is:

$$\begin{aligned}
\mathbb{E}_{\mathbf{Y}|\mathbf{x}}[f(\mathbf{y})] &= \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) f(\mathbf{y}) \\
&= \sum_{\mathbf{y}} \frac{1}{Z(\mathbf{x})} \left[\prod_{t=1}^{T} \psi_t(y_t, y_{t-1}, \mathbf{x}) \right] \cdot \left[\sum_{t=1}^{T} g(y_t, y_{t-1}) \right].
\end{aligned} \tag{A.6}$$

Now, let

$$\sigma_t(y_t) \stackrel{def}{=} \sum_{\mathbf{y}_{\langle 1 \ldots t-1 \rangle}} \left[\prod_{t'=1}^{t} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) \right] \cdot \left[\sum_{t'=1}^{t} g(y_{t'}, y_{t'-1}) \right]. \tag{A.7}$$

From equations A.6 and A.7, it follows that $\mathbb{E}_{\mathbf{Y}|\mathbf{x}}[f(\mathbf{y})] = \frac{1}{Z(\mathbf{x})} \sum_{y_T} \sigma_T(y_T)$.

By pushing the appropriate sums inside the product and splitting the sum of functions, we can establish the following recurrence relation:

$$\begin{aligned}
\sigma_t(y_t) &= \sum_{y_{t-1}} \left[\psi_t(y_t, y_{t-1}, \mathbf{x}) \cdot \sum_{\mathbf{y}_{\langle 1 \ldots t-2 \rangle}} \prod_{t'=1}^{t-1} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) \cdot \left[g(y_t, y_{t-1}) + \sum_{t'=1}^{t-1} g(y_{t'}, y_{t'-1}) \right] \right] \\
&= \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) g(y_t, y_{t-1}) \cdot \left[\sum_{\mathbf{y}_{\langle 1 \ldots t-2 \rangle}} \prod_{t'=1}^{t-1} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) \right] \\
&\quad + \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) \left[\sum_{\mathbf{y}_{\langle 1 \ldots t-2 \rangle}} \prod_{t'=1}^{t-1} \psi_{t'}(y_{t'}, y_{t'-1}, \mathbf{x}) \cdot \sum_{t'=1}^{t-1} g(y_{t'}, y_{t'-1}) \right] \\
&= \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) g(y_t, y_{t-1}) \alpha_{t-1}(y_{t-1}) + \sum_{y_{t-1}} \psi_t(y_t, y_{t-1}, \mathbf{x}) \sigma_{t-1}(y_{t-1}),
\end{aligned} \tag{A.8}$$

with a base case of $\sigma_0(y_t) = 0$. This recurrence relation can be implemented using a dynamic programming solution analogous to the forward algorithm, and require computation of the forward variables.

A common example is entropy:

$$\begin{aligned} f(\mathbf{y}) &= -\log p(\mathbf{y}|\mathbf{x}) \\ &= -\log \left\{ \frac{1}{Z(\mathbf{x})} \prod_t \psi_t(y_t, y_{t-1}, \mathbf{x}) \right\} \\ &= \log Z(\mathbf{x}) - \sum_t \log \psi_t(y_t, y_{t-1}, \mathbf{x}). \end{aligned}$$

Now let $g(y_t, y_{t-1}) = \log \psi_t(y_t, y_{t-1}, \mathbf{x})$. Then due to linearity,

$$\mathbb{E}_{\mathbf{Y}|\mathbf{x}}[f(\mathbf{y})] = \log Z(\mathbf{x}) - \mathbb{E}_{\mathbf{Y}|\mathbf{x}} \left[\sum_t g(y_t, y_{t-1}) \right]; \tag{A.9}$$

equation A.8 provides the solution to the latter expectation and yields the same recursive relation as reported by Busby and Ringger [11]. Although the resultant relation differs from Mann and McCallum [55] and Hernando et al. [38], they are equivalent and share similarities. For instance, Hernando et al. [38] solution requires simultaneous computation of the forward probability and the recursive entropy. The main conceptual difference is that their solution mathematically incorporates $\log Z(\mathbf{x})$ during recursion. We note that by applying our more general solution to the specific case of entropy, the resultant solution requires a single pass of the forward algorithm and an additional pass for recursive entropy; this contrasts to the four passes required by the approach of Mann and McCallum [55] which could equate to significant gains in practice for problems with a large output space or large datasets.

A.6 Extensions and Conclusions

In this section, we consider a special case of the sum of functions as well as an extension to the basic algorithm that can account for constraints.

As a special case of equation A.5, consider the case when the function $g(\cdot)$ depends only on the current latent variable y_t (as is the case with accuracy), only the marginal $p(y_t|\mathbf{x})$ is necessary:

$$\begin{aligned}
\mathbb{E}_{\mathbf{Y}|\mathbf{x}}\left[\sum_t g(y_t)\right] &= \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) \sum_t g(y_t) \\
&= \sum_t \sum_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}) g(y_t) \\
&= \sum_t \sum_{y_t} \sum_{\mathbf{y}_{\setminus t}} p(\mathbf{y}_{\setminus t}, y_t|\mathbf{x}) g(y_t) \\
&= \sum_t \sum_{y_t} g(y_t) \sum_{\mathbf{y}_{\setminus t}} p(\mathbf{y}_{\setminus t}, y_t|\mathbf{x}) \\
&= \sum_t \sum_{y_t} p(y_t|\mathbf{x}) g(y_t)
\end{aligned}$$

The marginal probability itself can be computed using the forward and backward probabilities. In many cases, this may be simpler to implement, if methods are already available that compute the marginal.

Accuracy and utility are examples of this type of function. In the case of accuracy, $g(y_t) = \mathbb{1}(y_t = \hat{y}_t)$, where $\hat{y}_t$ is the gold standard.

Another extension is based on what Mann and McCallum [55] call *subsequence constrained entropy*: some of the labels of a sequence are observed and others are not. The sum in the expectation is only over assignments of unobserved variables and the full joint probability of all variables $p(\mathbf{y}|\mathbf{x})$ is used. The more general expectations presented herein can also be constrained in a similar fashion; the only change required is that for the recursive relation for the known Y_t, we omit the sum over possible assignments to that value and simply use the known value.

In conclusion, when using LVMMs, it is possible to efficiently compute expectations of two classes of functions: products and sums of functions of the individual latent variables and their predecessors. This generic approach is more efficient than some previously published approaches for entropy and is easily adapted to computing expectations using subsequence constraints. Future

work could focus on similar generalizations for expectations using trees or other structured models.

Appendix B

Derivation of the Complete Conditionals for the Multiple Annotator Model

In this appendix, we derive the complete conditionals for the collapsed and blocked Gibbs sampler for the model presented in Figure 10.1. For this sampler, it is necessary to derive the complete conditional distributions of Y_i and M_i (a block), given the values of all the other Ys and Ms, denoted $\boldsymbol{y}_{\neg i}$ and $\boldsymbol{m}_{\neg i}$, respectively (the parameters are analytically integrated out in the collapsed model). The complete conditionals, $p(y_i, m_i | \boldsymbol{y}_{\neg i}, \boldsymbol{m}_{\neg i}, \boldsymbol{a}, \boldsymbol{x})$, are denoted in shorthand as $[y_i, m_i]$.

We begin with the joint distribution over all random variables in the model:

$$
\begin{aligned}
& p(\boldsymbol{\theta}, \boldsymbol{\mu}, \boldsymbol{\alpha}, \boldsymbol{\phi}, \boldsymbol{y}, \boldsymbol{m}, \boldsymbol{a}, \boldsymbol{x}) \\
& \quad = p(\boldsymbol{\theta}|b_\theta) \left(\prod_{k=1}^{K} p(\boldsymbol{\mu}_k | \boldsymbol{b}_{\mu_k}) \right) \left(\prod_{j=1}^{J} \prod_{k=1}^{K} p(\boldsymbol{\alpha}_{jk} | \boldsymbol{b}_{\alpha_{jk}}) \right) \left(\prod_{k=1}^{K} p(\boldsymbol{\phi}_k | b_\phi) \right) \\
& \qquad \cdot \left(\prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}) p(m_i|y_i, \boldsymbol{\mu}) \left(\prod_{j=1}^{J} p(\boldsymbol{a}_{ij}|y_i, \boldsymbol{\alpha}_j) \right) p(\boldsymbol{x}_i|m_i, \boldsymbol{\phi}) \right) \\
& \quad \propto \left(\prod_{k=1}^{K} \theta_k^{b_\theta - 1} \right) \left(\prod_{k=1}^{K} \prod_{k'=1}^{K} \mu_{kk'}^{b_{\mu_k} - 1} \right) \left(\prod_{j=1}^{J} \prod_{k=1}^{K} \prod_{k'=1}^{K} \alpha_{jkk'}^{b_{\alpha_{jk}} - 1} \right) \left(\prod_{k=1}^{K} \prod_{f=1}^{F} \phi_{kf}^{b_\phi - 1} \right) \\
& \qquad \cdot \left(\prod_{i=1}^{N} \theta_{y_i} \mu_{y_i m_i} \left(\prod_{j=1}^{J} \frac{|\boldsymbol{a}_{ij}|_1!}{\prod_{k=1}^{K} a_{ijk}!} \prod_{k=1}^{K} \alpha_{jy_ik}^{a_{ijk}} \right) \left(\frac{|\boldsymbol{x}_i|_1!}{\prod_{f=1}^{F} x_{if}!} \prod_{f=1}^{F} \phi_{m_i f}^{x_{if}} \right) \right)
\end{aligned}
$$

where $|\cdot|_1$ denotes the L_1-norm. Also note that the normalizing constants for the Dirichlet distributions have been absorbed into the constant of proportionality.

For notational brevity, we define the following count variables, which are disambiguated by their superscripts ($\blacksquare$ is an indicator function)

$$n_k^{\theta} = \sum_i^N \blacksquare(y_i = k)$$
$$n_{kk'}^{\mu} = \sum_i^N \blacksquare(y_i = k \wedge m_i = k')$$
$$n_{jkk'}^{\alpha} = \sum_i^N a_{ijk'}^{\mathbf{1}(y_i=k)}$$
$$n_{kf}^{\phi} = \sum_i^N x_{if}^{\mathbf{1}(m_i=j)}.$$

That is, n_k^{θ} is the number of instances currently labeled k; $n_{kk'}^{\mu}$ is the number of instances labeled k that have a features-based prediction (i.e., r.v. M) of k'; $n_{jkk'}^{\alpha}$ is the number of times that annotator j chose annotation k' on instances labeled k; and n_{kf}^{ϕ} is the number of times feature f occurs with instances having a feature-based prediction of k.

Next, we apply the commutative property of multiplication and combine like terms:

$$
\begin{aligned}
p(\boldsymbol{\theta},\boldsymbol{\mu},\boldsymbol{\alpha},\boldsymbol{\phi},\boldsymbol{y},\boldsymbol{m},\boldsymbol{a},\boldsymbol{x}) \propto & \left(\prod_{k=1}^{K}\theta_k^{b_\theta-1}\right)\left(\prod_{i=1}^{N}\theta_{y_i}\right)\left(\prod_{k=1}^{K}\prod_{k'=1}^{K}\mu_{kk'}^{b_{\mu_k}-1}\right)\left(\prod_{i=1}^{N}\mu_{y_i m_i}\right) \\
& \cdot\left(\prod_{j=1}^{J}\prod_{k=1}^{K}\prod_{k'=1}^{K}\alpha_{jkk'}^{b_{\alpha_{jk}}-1}\right)\left(\prod_{i=1}^{N}\prod_{j=1}^{J}\frac{|\boldsymbol{a}_{ij}|_1!}{\prod_{k=1}^{K}a_{ijk}!}\prod_{k=1}^{K}\alpha_{jy_ik}^{a_{ijk}}\right) \\
& \cdot\left(\prod_{k=1}^{K}\prod_{f=1}^{F}\phi_{kf}^{b_\phi-1}\right)\left(\prod_{i=1}^{N}\frac{|\boldsymbol{x}_i|_1!}{\prod_{f=1}^{F}x_{if}!}\prod_{f=1}^{F}\phi_{m_i f}^{x_{if}}\right) \\
\propto & \left(\prod_{k=1}^{K}\theta_k^{b_\theta-1}\right)\left(\prod_{k=1}^{K}\theta_k^{n_k^\theta}\right)\left(\prod_{k=1}^{K}\prod_{k'=1}^{K}\mu_{kk'}^{b_{\mu_k}-1}\right)\left(\prod_{k=1}^{K}\prod_{k'=1}^{K}\mu_{kk'}^{n_{kk'}^\theta}\right) \\
& \cdot\left(\prod_{j=1}^{J}\prod_{k=1}^{K}\prod_{k'=1}^{K}\alpha_{jkk'}^{b_{\alpha_{jk}}-1}\right)\left(\prod_{j=1}^{J}\prod_{k=1}^{K}\prod_{k'=1}^{K}\alpha_{jkk'}^{n_{jkk'}^\alpha}\right)\left(\prod_{i=1}^{N}\prod_{j=1}^{J}\frac{|\boldsymbol{a}_{ij}|_1!}{\prod_{k=1}^{K}a_{ijk}!}\right) \\
& \cdot\left(\prod_{k=1}^{K}\prod_{f=1}^{F}\phi_{kf}^{b_\phi-1}\right)\left(\prod_{k=1}^{K}\prod_{f=1}^{F}\phi_{kf}^{n_{kf}^\phi}\right)\left(\prod_{i=1}^{N}\frac{|\boldsymbol{x}_i|_1!}{\prod_{f=1}^{F}x_{if}!}\right) \\
\propto & \prod_{k=1}^{K}\theta_k^{b_\theta+n_k^\theta-1}\prod_{k=1}^{K}\prod_{k'=1}^{K}\mu_{kk'}^{b_\mu+n_{kk'}^\mu-1} \\
& \cdot\prod_{j=1}^{J}\prod_{k=1}^{K}\prod_{k'=1}^{K}\alpha_{jkk'}^{b_{\alpha_j}+n_{jkk'}^\alpha-1}\prod_{k=1}^{K}\prod_{f=1}^{F}\phi_{kf}^{b_\phi+n_{kf}^\phi-1} \\
& \cdot\left(\prod_{i=1}^{N}\prod_{j=1}^{J}\frac{|\boldsymbol{a}_{ij}|_1!}{\prod_{k=1}^{K}a_{ijk}!}\right)\left(\prod_{i=1}^{N}\frac{|\boldsymbol{x}_i|_1!}{\prod_{f=1}^{F}x_{if}!}\right)
\end{aligned}
$$

Since we are interested in a fully collapsed sampler wherein all of the parameters are integrated out:

$$
\begin{aligned}
p(\boldsymbol{y},\boldsymbol{m},\boldsymbol{a},\boldsymbol{x}) &= \int \dots \int p(\boldsymbol{\theta},\boldsymbol{\mu},\boldsymbol{\alpha},\boldsymbol{\phi},\boldsymbol{y},\boldsymbol{m},\boldsymbol{a},\boldsymbol{x})\, d\boldsymbol{\theta}\, d\boldsymbol{\mu}\, d\boldsymbol{\alpha}\, d\boldsymbol{\phi} \\
&\propto \left(\int \prod_{k=1}^{K} \theta_k^{b_\theta + n_k^\theta - 1} d\boldsymbol{\theta} \right) \left(\prod_{k=1}^{K} \int \prod_{k'=1}^{K} \mu_{kk'}^{b_\mu + n_{kk'}^\mu - 1} d\boldsymbol{\mu}_k \right) \\
&\quad \cdot \left(\prod_{j=1}^{J} \prod_{k=1}^{K} \int \prod_{k'=1}^{K} \alpha_{jkk'}^{b_{\alpha_j} + n_{jkk'}^\alpha - 1} d\boldsymbol{\alpha}_{jk} \right) \left(\prod_{k=1}^{K} \int \prod_{f=1}^{F} \phi_{kf}^{b_\theta + n_{kf}^\phi - 1} d\boldsymbol{\phi}_k \right) \\
&\quad \cdot \left(\prod_{i=1}^{N} \prod_{j=1}^{J} \frac{|\boldsymbol{a}_{ij}|_1!}{\prod_{k=1}^{K} a_{ijk}!} \right) \left(\prod_{i=1}^{N} \frac{|\boldsymbol{x}_i|_1!}{\prod_{f=1}^{F} x_{if}!} \right) \\
&= \left(\frac{\prod_{k=1}^{K} \Gamma\left(b_\theta + n_k^\theta\right)}{\Gamma\left(\sum_{k=1}^{K} \left(b_\theta + n_k^\theta\right)\right)} \right) \left(\prod_{k=1}^{K} \frac{\prod_{k'=1}^{K} \Gamma\left(b_\mu + n_{kk'}^\mu\right)}{\Gamma\left(\sum_{k'=1}^{K} \left(b_\mu + n_{kk'}^\mu\right)\right)} \right) \\
&\quad \cdot \left(\prod_{j=1}^{J} \prod_{k=1}^{K} \frac{\prod_{k'=1}^{K} \Gamma\left(b_{\alpha_j} + n_{jkk'}^\alpha\right)}{\Gamma\left(\sum_{k'=1}^{K} \left(b_{\alpha_j} + n_{jkk'}^\alpha\right)\right)} \right) \left(\prod_{k=1}^{K} \frac{\prod_{f=1}^{F} \Gamma\left(b_\phi + n_{kf}^\phi\right)}{\Gamma\left(\sum_{f=1}^{F} \left(b_\phi + n_{kf}^\phi\right)\right)} \right) \\
&\quad \cdot \left(\prod_{i=1}^{N} \prod_{j=1}^{J} \frac{|\boldsymbol{a}_{ij}|_1!}{\prod_{k=1}^{K} a_{ijk}!} \right) \left(\prod_{i=1}^{N} \frac{|\boldsymbol{x}_i|_1!}{\prod_{f=1}^{F} x_{if}!} \right)
\end{aligned}
$$

The last step follows from the definition of the pdf of a Dirichlet distribution, in particular, the values of the integrals in the equation are the normalizing constant of the Dirichlet distribution.

We define a block sampler in which y_i and m_i are sampled jointly. First, we note that the multinomial coefficients are constant within the complete conditionals, since $\boldsymbol{a}$ and $\boldsymbol{x}$ are given. Next, let $\boldsymbol{y}_{\neg i} = \{y_j | j \neq i\}$; $\boldsymbol{m}_{\neg i}$ is similarly defined. Please note that in the following equations, we have redefined the count variables defined previously (i.e., $n_k^\theta, n_{kk'}^\mu, n_{jkk'}^\alpha, n_{kf}^\phi$) to exclude counts from the current instance (i), i.e., they are sums over $\boldsymbol{y}_{\neg i}$ and $\boldsymbol{m}_{\neg i}$; we then manually add in the

appropriate counts from the current instance.

$$
\begin{aligned}
[Y_i = c, M_i = d] &\overset{def}{=} p(Y_i = c, M_i = d | \boldsymbol{y}_{\neg i}, \boldsymbol{m}_{\neg i}, \boldsymbol{a}, \boldsymbol{x}) \\
&\propto p(Y_i = c, M_i = d, \boldsymbol{y}_{\neg i}, \boldsymbol{m}_{\neg i}, \boldsymbol{a}, \boldsymbol{x}) \\
&\propto \frac{\prod_{k \neq c} \Gamma\left(b_\theta + n_k^\theta\right) \Gamma\left(b_\theta + n_c^\theta + 1\right)}{\Gamma\left(\sum_{k \neq c}\left(b_\theta + n_k^\theta\right) + b_\theta + n_c^\theta + 1\right)} \\
&\quad \cdot \prod_{k \neq c} \frac{\prod_{k'=1}^{K} \Gamma\left(b_\mu + n_{kk'}^\mu\right)}{\Gamma\left(\sum_{k'=1}^{K}\left(b_\mu + n_{kk'}^\mu\right)\right)} \frac{\prod_{k' \neq d} \Gamma\left(b_\mu + n_{ck'}^\mu\right) \Gamma\left(b_\mu + n_{cd}^\mu + 1\right)}{\Gamma\left(\sum_{k' \neq d}\left(b_\mu + n_{ck'}^\mu\right) + b_\mu + n_{cd}^\mu + 1\right)} \\
&\quad \cdot \prod_{j=1}^{J} \prod_{k \neq c} \frac{\prod_{k'=1}^{K} \Gamma\left(b_{\alpha_j} + n_{jkk'}^\alpha\right)}{\Gamma\left(\sum_{k'=1}^{K}\left(b_{\alpha_j} + n_{jkk'}^\alpha\right)\right)} \frac{\prod_{k'=1}^{K} \Gamma\left(b_{\alpha_j} + n_{jck'}^\alpha + a_{ijk'}\right)}{\Gamma\left(\sum_{k'=1}^{K}\left(b_{\alpha_j} + n_{jck'}^\alpha + a_{ijk'}\right)\right)} \\
&\quad \cdot \prod_{k \neq d} \frac{\prod_{f=1}^{F} \Gamma\left(b_\phi + n_{kf}^\phi\right)}{\Gamma\left(\sum_{f=1}^{F}\left(b_\phi + n_{kf}^\phi\right)\right)} \frac{\prod_{f=1}^{F} \Gamma\left(b_\phi + n_{df}^\phi + x_{if}\right)}{\Gamma\left(\sum_{f=1}^{F}\left(b_\phi + n_{df}^\phi + x_{if}\right)\right)}
\end{aligned}
\tag{B.1}
$$

To simplify, we eliminate the products whose index involves the exclusion of c or d. To do so, we simply multiply by unity, e.g.,

$$
\begin{aligned}
g(c) \prod_{k \neq c} f(k) &= g(c) \frac{f(c)}{f(c)} \prod_{k \neq c} f(k) \\
&= g(c) \frac{1}{f(c)} \prod_{k} f(k) \\
&\propto \frac{g(c)}{f(c)}.
\end{aligned}
$$

Noting that the sum in the denominator of the first term of equation B.1 is constant ($\sum_{k \neq c} n_k^\theta + n_c + 1$ is equal to the number of instances) and can be dropped, the result of applying the aforementioned

simplification is:

$$
\begin{aligned}
[Y_i = c, M_i = d] \propto & \frac{\Gamma(b_\theta + n_c^\theta + 1)}{\Gamma(b_\theta + n_c^\theta)} \\
& \cdot \frac{\Gamma\left(\sum_{k'=1}^{K}\left(b_\mu + n_{ck'}^\mu\right)\right)}{\prod_{k'=1}^{K}\Gamma\left(b_\mu + n_{ck'}^\mu\right)} \frac{1}{\Gamma(b_\mu + n_{cd}^\mu)} \frac{\Gamma(b_\mu + n_{cd}^\mu + 1)}{\Gamma\left(\sum_{k' \neq d}(b_\mu + n_{ck'}^\mu) + b_\mu + n_{cd}^\mu + 1\right)} \\
& \cdot \prod_{j=1}^{J} \frac{\Gamma\left(\sum_{k'=1}^{K}\left(b_{\alpha_j} + n_{jck'}^\alpha\right)\right)}{\prod_{k'=1}^{K}\Gamma\left(b_{\alpha_j} + n_{jck'}^\alpha\right)} \frac{\prod_{k'=1}^{K}\Gamma\left(b_{\alpha_j} + n_{jck'}^\alpha + a_{ijk'}\right)}{\Gamma\left(\sum_{k'=1}^{K}\left(b_{\alpha_j} + n_{jck'}^\alpha + a_{ijk'}\right)\right)} \\
& \cdot \frac{\Gamma\left(\sum_{f=1}^{F}\left(b_\phi + n_{df}^\phi\right)\right)}{\prod_{f=1}^{F}\Gamma\left(b_\phi + n_{df}^\phi\right)} \frac{\prod_{f=1}^{F}\Gamma\left(b_\phi + n_{df}^\phi + x_{if}\right)}{\Gamma\left(\sum_{f=1}^{F}\left(b_\phi + n_{df}^\phi + x_{if}\right)\right)} \\
= & \frac{\Gamma(b_\theta + n_c^\theta + 1)}{\Gamma(b_\theta + n_c^\theta)} \\
& \cdot \frac{1}{\prod_{k'=1}^{K}\Gamma(b_\mu + n_{ck'}^\mu)} \frac{\Gamma(b_\mu + n_{cd}^\mu + 1)}{\Gamma(b_\mu + n_{cd}^\mu)} \frac{\Gamma(\sum_{k'=1}^{K}(b_\mu + n_{ck'}^\mu))}{\Gamma\sum_{k' \neq d}(b_\mu + n_{ck'}^\mu) + b_\mu + n_{cd}^\mu + 1} \\
& \cdot \prod_{j=1}^{J}\left(\prod_{k'=1}^{K}\frac{\Gamma(b_{\alpha_j} + n_{jck'}^\alpha + a_{ijk'})}{\Gamma(b_{\alpha_j} + n_{jck'}^\alpha)}\right) \frac{\Gamma\left(\sum_{k'=1}^{K}\left(b_{\alpha_j} + n_{jck'}^\alpha\right)\right)}{\Gamma\left(\sum_{k'=1}^{K}\left(b_{\alpha_j} + n_{jck'}^\alpha + a_{ijk'}\right)\right)} \\
& \cdot \prod_{f=1}^{F}\frac{\Gamma(b_\phi + n_{df}^\phi + x_{if})}{\Gamma(b_\phi + n_{df}^\phi)} \frac{\Gamma\left(\sum_{f=1}^{F}\left(b_\phi + n_{jck'}^\phi\right)\right)}{\Gamma\left(\sum_{f=1}^{F}\left(b_\phi + n_{jck'}^\phi + x_{if}\right)\right)}
\end{aligned}
$$

Let $x^{(n)} \stackrel{def}{=} x(x+1)\ldots(x+n-1)$ be the rising factorial. Note that $\frac{\Gamma(x+n)}{\Gamma(x)} = x^{(n)}$. After re-ordering the numerators and denominators to identify rising factorials and noting that $x^{(1)} = x$, we are left with:

$$
\begin{aligned}
[Y_i = c, M_i = d] \propto & (b_\theta + n_c^\theta) \frac{1}{\prod_{k'=1}^{K}\Gamma(b_\mu + n_{ck'}^\mu)} (b_\mu + n_{cd}^\mu) \frac{1}{\sum_{k'=1}^{K}(b_\mu + n_{ck'}^\mu)} \\
& \cdot \prod_{j=1}^{J} \frac{1}{\left(\sum_{k'=1}^{K} b_{\alpha_j} + n_{jck'}^\alpha\right)^{(\sum_{k'=1}^{K} a_{ijk'})}} \prod_{k'=1}^{K}\left(b_{\alpha_j} + n_{jck'}^\alpha\right)^{(\sum_{k'=1}^{K} a_{ijk'})} \\
& \cdot \frac{1}{\left(\sum_{f=1}^{F}\left(b_\phi + n_{df}^\phi\right)\right)^{(\sum_{f=1}^{F} x_{if})}} \prod_{f=1}^{F}\left(b_\phi + n_{df}^\phi\right)^{(\sum_{f=1}^{F} x_{if})}
\end{aligned}
$$

These are the complete conditionals as presented in equation 10.1.

References

[1] Naoki Abe and Hiroshi Mamitsuka. Query learning strategies using boosting and bagging. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 1–9. Morgan Kaufmann Publishers Inc., 1998.

[2] Brigham Anderson and Andrew Moore. Active learning for hidden Markov models: Objective functions and algorithms. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 9–16, 2005.

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

[4] Shilpa Arora and Eric Nyberg. Assessing benefit from feature feedback in active learning for text classification. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 106–114. Association for Computational Linguistics, 2011.

[5] Shilpa Arora, Eric Nyberg, and Carolyn P. Rosé. Estimating annotation cost for active learning in a multi-annotator environment. In *Proceedings of the HLT-NAACL 2009 Workshop on Active Learning for Natural Language Processing*, pages 18–26, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[6] Jason Baldridge and Miles Osborne. Active learning and the total cost of annotation. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004.

[7] Jason Baldridge and Alexis Palmer. How well does active learning actually work?: Time-based evaluation of cost-reduction strategies for language documentation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 296–305. Association for Computational Linguistics, 2009.

[8] Markus Becker, Ben Hachey, Beatrice Alex, and Claire Grover. Optimising selective sampling for bootstrapping named entity recognition. In *Proceedings of the the International Conference on Machine Learning Workshop on Learning with Multiple Views*, 2005.

[9] Thorsten Brants. TnT: A statistical part-of-speech tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231. Association for Computational Linguistics, 2000.

[10] Eric Brill and Jun Wu. Classifier combination for improved lexical disambiguation. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 1, pages 191–195. Association for Computational Linguistics, 1998.

[11] George Busby and Eric K. Ringger. An exact, efficient algorithm for computing the conditional label sequence entropy. Technical Report NLP-TR5, Computer Science Department, Brigham Young University, 2007. URL `http://nlp.cs.byu.edu/doc/techreports/TR5/BYUNLP-TR5.pdf`.

[12] Bob Carpenter. Multilevel bayesian models of categorical data annotation. *Unpublished manuscript*, 2008.

[13] James L. Carroll. *A Bayesian decision theoretical approach to supervised learning, selective sampling, and empirical function optimization*. PhD thesis, Brigham Young University, 2010.

[14] James L Carroll, Robbie Haertel, Peter McClanahan, Eric Ringger, and Kevin Seppi. Modeling the annotation process for ancient corpus creation. In Petr Zemánek, Jost Gippert, Hans-Christian Luschützky, and Petr Vavroušek, editors, *Chatreššar 2007: Proceedings of the International Conference of Electronic Corpora of Ancient Languages*, pages 25–42. Charles University, 2007.

[15] James L. Carroll, Neil Toronto, Kevin D. Seppi, and Robbie A. Haertel. Explicit utility in supervised learning. Poster at *Neural Information Processing Systems 2008 Workshop on Cost Sensitive Learning*, 2008.

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

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

[18] Aron Culotta and Andrew McCallum. Reducing labeling effort for structured prediction tasks. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 746, 2005.

[19] Ido Dagan and Sean P. Engelson. Committee-based sampling for training probabilistic classifiers. In *Proceedings of the International Conference on Machine Learning*, page 150–157, 1995.

[20] Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. *Learning Theory*, pages 889–905, 2005.

[21] Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. *The Journal of Machine Learning Research*, 10:281–299, 2009.

[22] A.P. Dawid and A.M. Skene. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied Statistics*, pages 20–28, 1979.

[23] David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 348–355. Association for Computational Linguistics, 1997.

[24] Pinar Donmez and Jaime G. Carbonell. Proactive learning: Cost-sensitive active learning with multiple imperfect oracles. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 619–628. ACM, 2008.

[25] Pinar Donmez, Jaime G. Carbonell, and Jeff Schneider. Efficiently learning the accuracy of labeling sources for selective sampling. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268. ACM, 2009.

[26] Pinar Donmez, Jaime Carbonell, and Jeff Schneider. A probabilistic framework to learn from multiple annotators with time-varying accuracy. In *SIAM International Conference on Data Mining (SDM)*, pages 826–837, 2010.

[27] Sean P. Engelson and Ido Dagan. Minimizing manual annotation cost in supervised training from corpora. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, pages 319–326, 1996.

[28] Paul Felt, Eric Ringger, evin. Seppi, Kristian Heal, Robbie Haertel, and Deryle Lonsdale. First results in a study evaluating pre-labeling and correction propagation for machine-assisted syriac morphological analysis. *Proceedings of the International Conference on Language Resources and Evaluation*, 2012.

[29] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28:133–168, 1997.

[30] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian data analysis*. Chapman & Hall/CRC, 2004.

[31] Geoffrey Godbert and Jay Ramsay. *For Now*. Diamond, 1991.

[32] Gahgene Gweon, Carolyn Ros, Joerg Wittwer, and Matthias Nueckles. Supporting efficient and reliable content analysis using automatic text processing technology. In Maria Costabile and Fabio Patern, editors, *Human-Computer Interaction - INTERACT 2005*, volume 3585 of *Lecture Notes in Computer Science*, pages 1112–1115. Springer Berlin / Heidelberg, 2005.

[33] Ben Hachey, Beatrice Alex, and Markus Becker. Investigating the effects of selective sampling on the annotation task. In *Proceedings of Computational Natural Language Learning*, 2005.

[34] Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. Assessing the costs of sampling methods in active learning for annotation. In *Proceedings of the 46th Annual Meeting of the Association of Computational Linguistics*, Columbus, OH, USA, June 2008.

[35] Robbie Haertel, Paul Felt, Eric Ringger, and Kevin Seppi. Parallel active learning: Eliminating wait time with minimal staleness. In *Proceedings of the HLT-NAACL 2010 Workshop on Active Learning for Natural Language Processing*, pages 33–41. Association for Computational Linguistics, 2010.

[36] Robbie A. Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. Return on investment for active learning. In *Proceedings of the Neural Information Processing Systems Workshop on Cost Sensitive Learning*, 2008.

[37] Jiawei Han. Mining heterogeneous information networks by exploring the power of links. In *Discovery Science*, pages 13–30. Springer, 2009.

[38] Diego Hernando, Valentino Crespi, and George Cybenko. Efficient computation of the hidden Markov model entropy for a given observation sequence. *IEEE Transactions on Information Theory*, 51(7):2681–2685, 2005. ISSN 0018-9448.

[39] Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence Journal*, 126:159–96, 2001.

[40] Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. Learning whom to trust with mace. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1120–1130, 2013.

[41] Ted Hughes. *Selected Poems: 1957-1981*. Faber & Faber, 1982.

[42] Rebecca Hwa. Sample selection for statistical grammar induction. In *The Conference on Empirical Methods in Natural Language Processing*, pages 45–52, 2000.

[43] Rebecca Hwa. Sample selection for statistical parsing. *Computational Linguistics*, 30:253–276, 2004.

[44] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 143–151. Morgan Kaufmann Publishers Inc., 1997.

[45] Rosie Jones, Rayid Ghani, Tom Mitchell, and Ellen Riloff. Active learning for information extraction with multiple view feature sets. In *Proceedings of the Adaptive Text Extraction and Mining Workshop (ATEM03)*, pages 26–34, 2003.

[46] Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2007.

[47] Ross D. King, Kenneth E. Whelan, Ffion M. Jones, Phillip G. K. Reiser, Chrisopher H. Bryant, Stephen H. Muggleton, Douglas B. Kell, and Stephen G. Oliver. Functional genomic hypothesis generation and experimentation by a robot scientist. *Nature*, 427(6971):247–252, 2004.

[48] George A. Kiraz. Automatic concordance generation of Syriac texts. *Orientalia Christiana analecta*, pages 461–475, 1994.

[49] Julian Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer Speech & Language*, 6(3):225–242, 1992.

[50] David D. Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In William W. Cohen and Haym Hirsh, editors, *Proceedings of the International Conference on Machine Learning-94, 11th International Conference on Machine Learning*, page 148156, New Brunswick, US, 1994. Morgan Kaufmann Publishers, San Francisco, US.

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

[52] Pery Liang, Michael I. Jordan, and Dan Klein. Learning from measurements in exponential families. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 641–648. ACM, 2009.

[53] Christopher H. Lin, Mausam, and Daniel S. Weld. Dynamically switching between synergistic workflows for crowdsourcing. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[54] Jun S. Liu. The collapsed gibbs sampler in bayesian computations with applications to a gene regulation problem. *Journal of the American Statistical Association*, 89(427):pp. 958–966, 1994. ISSN 01621459. URL `http://www.jstor.org/stable/2290921`.

[55] Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 109–112. Association for Computational Linguistics, 2007.

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

[57] Mitchell P. Marcus, Beatrice Santorini, Mary Ann Marcinkiewicz, and Ann Taylor. Treebank–3. *Philadelphia: Linguistic Data Consortium*, 1999.

[58] Dragos D. Margineantu. Active cost-sensitive learning. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, volume 19, page 1622, 2005.

[59] Andrew Kachites McCallum and Kamal Nigam. Employing EM and pool-based active learning for text classification. In *Proceedings of the International Conference on Machine Learning*, pages 350–358, 1998.

[60] Marina Meilă and David Heckerman. An experimental comparison of model-based clustering methods. *Machine Learning*, 42(1-2):9–29, 2001.

[61] Grace Ngai and David Yarowsky. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of ACL*, pages 117–125, 2000.

[62] Fredrik Olsson. A literature survey of active machine learning in the context of natural language processing. Technical Report T2009:06, Swedish Institute of Computer Science, 2009.

[63] Miles Osborne and Jason Baldridge. Ensemble-based active learning for parse selection. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 89–96, 2004.

[64] Alexis Palmer, Taesun Moon, and Jason Baldridge. Evaluating automation strategies in language documentation. In *Proceedings of the HLT-NAACL 2009 Workshop on Active*

Learning for Natural Language Processing, pages 36–44. Association for Computational Linguistics, 2009.

[65] Jeff Pasternack and Dan Roth. Knowing what to believe (when you already know something). In *COLING*, Beijing, China, 8 2010. URL `http://cogcomp.cs.illinois.edu/papers/PasternackRo10.pdf`.

[66] Jeff Pasternack and Dan Roth. Comprehensive trust metrics for information networks. In *ASC*, Orlando, Florida, 12 2010. URL `http://cogcomp.cs.illinois.edu/papers/PasternackRo10b.pdf`.

[67] Jeff Pasternack and Dan Roth. Latent credibility analysis. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1009–1020, 2013.

[68] Lawrence Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[69] Howard Raiffa and Robert Schlaifer. *Applied Statistical Decision Theory*. Division of Research, Graduate School of Business Adminitration, Harvard University, 1961.

[70] Craig Raine. *Rich*. Faber & Faber, 1984.

[71] Adwait Ratnaparkhi. A maximum entropy part-of-speech tagger. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996:133–142, 1996.

[72] Vikas C. Raykar and Shipeng Yu. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *The Journal of Machine Learning Research*, 13:491–518, 2012.

[73] Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin. Seppi, and Deryle Lonsdale. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of Linguistic Annotation Workshop*, pages 101–108, 2007.

[74] Eric Ringger, Marc Carmen, Robbie Haertel, Kevin Seppi, Deryle Londsale, Peter McClanahan, James Carroll, and Noel Ellison. Assessing the costs of machine-assisted corpus annotation through a user study. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2008.

[75] Nicholas Roy and Andrew McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 441–448, 2001.

[76] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden Markov models for information extraction. *Advances in Intelligent Data Analysis*, pages 309–318, 2001.

[77] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proceedings of the International Conference on Machine Learning*, pages 839–846, 2000.

[78] Burr Settles. *Curious machines: Active learning with structured instances*. PhD thesis, University of Wisconsin-Madison, 2008.

[79] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[80] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[81] Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the Neural Information Processing Systems Workshop on Cost-Sensitive Learning*, pages 1069–1078, 2008.

[82] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 287–294, New York, NY, USA, 1992. ACM.

[83] Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 614–622. ACM, 2008.

[84] Padhraic Smyth, Usama Fayyad, Michael Burl, Pietro Perona, and Pierre Baldi. Inferring ground truth from subjective labelling of venus images. *Advances in Neural Information Processing Systems*, pages 1085–1092, 1995.

[85] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to Statistical Relational Learning*, 93:142–146, 2007.

[86] Min Tang, Xiaoqiang Luo, and Salim Roukos. Active learning for statistical natural language parsing. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 120–127. Association for Computational Linguistics, 2002.

[87] Sebastian Thrun and Knut Möller. Active exploration in dynamic environments. In *Advances in Neural Information Processing Systems*, volume 4, pages 531–538, 1991.

[88] Katrin Tomanek and Udo Hahn. Semi-supervised active learning for sequence labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1039–1047. Association for Computational Linguistics, 2009.

[89] Katrin Tomanek and Udo Hahn. A comparison of models for cost-sensitive active learning. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1247–1255. Association for Computational Linguistics, 2010.

[90] Katrin Tomanek and Fredrik Olsson. A web survey on the use of active learning to support annotation of text data. In *Proceedings of the Human Language Technology Conference of the HLT-NAACL 2009 Workshop on Active Learning for Natural Language Processing*, pages 45–48. Association for Computational Linguistics, 2009.

[91] Katrin Tomanek, Joachim Wermter, and Udo Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 486–495, 2007.

[92] Kristina Toutanova and Chrisopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 63–70, 2000.

[93] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Edmonton, Canada, 2003. Association for Computational Linguistics.

[94] Daniel David Walker and Eric K. Ringger. Model-based document clustering with a collapsed gibbs sampler. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 704–712. ACM, 2008.

[95] Daniel S. Weld, Mausam, and Peng Dai. Human intelligence needs artificial intelligence. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[96] Jacob Whitehill, Paul Ruvolo, Tingfan Wu, Jacob Bergsma, and Javier Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *Advances in Neural Information Processing Systems*, 22(2035-2043):7–13, 2009.

[97] Wikipedia. Standing on the shoulders of giants — Wikipedia, The Free Encyclopedia, 2013. URL http://en.wikipedia.org/w/index.php?title=Standing_on_the_shoulders_of_giants&oldid=559172313. [Online; accessed 16-June-2013].

[98] Tong Zhang and Frank J. Oles. The value of unlabeled data for classification problems. In *Proceedings of the Seventeenth International Conference on Machine Learning, (Langley, P., ed.)*, pages 1191–1198. Citeseer, 2000.