Exploring Weakly Labeled Data Across the Noise-Bias Spectrum

Robert W. H. Fisher

April 2016
CMU-ML-16-101

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Reid Simmons, Chair
Geoffrey Gordon
Carolyn Penstein Rosé
Dieter Fox, University of Washington

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Copyright © 2016 Robert W. H. Fisher

This research was sponsored by the National Science Foundation grant numbers DGE0750271, DGE1252522, EEC0540865, and IIS1450543.

Keywords: Weakly labeled data, spectral methods, latent variable models

Thanks to all my friends and family for their support, particularly my wife Katherine, without whom none of this would be possible.

Abstract

As the availability of unstructured data on the web continues to increase, it is becoming increasingly necessary to develop machine learning methods that rely less on human annotated training data. In this thesis, we present methods for learning from weakly labeled data. We present a unifying framework to understand weakly labeled data in terms of bias and noise and identify methods that are well suited to learning from certain types of weak labels. To compensate for the tremendous sizes of weakly labeled datasets, we leverage computationally efficient and statistically consistent spectral methods. Using these methods, we present results from four diverse, real-world applications coupled with a unifying simulation environment. This allows us to make general observations that would not be apparent when examining any one application on its own. These contributions allow us to significantly improve prediction when labeled data is available, and they also make learning tractable when the cost of acquiring annotated data is prohibitively high.

Acknowledgments

I would chiefly like to acknowledge my advisor Reid Simmons, and the rest of my thesis committee, Geoff Gordon, Carolyn Rosé, and Dieter Fox. Thanks also to the many colleagues who have provided valuable contributions to this work. In particular Tom Kollar, Ankhur Parikh, Maxim Makatchev, Eleanor Avrunin, Max Tomlinson, Miriam Robinson, Jayant Krishnamurthy, and the entire staff of the Human Engineering Research Laboratory. Finally thanks to the National Science Foundation Graduate Fellowship program, the Quality of Life Technology Center, and the American Rewards for College Scientists program for their generous financial support.

Contents

1 Introduction **1**

2 Related Work **5**

2.1 Background 5

2.1.1 Latent Variable Methods 5

2.1.2 Spectral Hidden Markov Model 6

2.1.3 Discourse Analysis 9

2.2 Related Work 11

3 Methodology **15**

3.1 Acquiring Annotated Data 16

3.1.1 Crowdsourcing 16

3.1.2 Expert Domains 18

3.1.3 Personalized Models 19

3.2 Weakly Labeled Data 19

3.3 Simulation Environment for Weakly Labeled and Unlabeled Data 23

3.4 Using Bootstrapping and Cross-training for Unlabeled Data 24

3.5 Improving the Quality of Weakly Labeled Data 27

3.5.1 Internal Error Correction for Noisy Data 27

3.5.2 Active Training for Weakly Labeled Data 29

3.5.3 Learning Weights for Weakly Labeled Data 32

3.5.4 A Combined Approach to Improving Data Quality 34

3.6 Overview of Applications 37

4 InContext Smartphone Application **43**

4.1 Introduction 43

4.2 Dataset 45

4.3 Weakly Labeled Data 47

4.4 System Overview 49

4.4.1 Phone Posture Recognition 49

4.4.2 Voice Activity Detection 50

4.4.3 Smoothing of Information Extraction Output 52

4.4.4 Preference Classification 52

4.4.5 Active Learning 54

4.5 Results 56
4.6 Conclusions 58

5 The Virtual Seating Coach System **59**
5.1 Introduction 59
5.2 The Virtual Seating Coach System 60
5.3 Weakly Labeled Data 62
5.4 Spectral Hidden Markov Models 64
5.5 Decision Trees 66
5.6 Results 67
5.7 Conclusions 69

6 Discourse Analysis **71**
6.1 Introduction 71
6.2 Problem Definition and Dataset 72
6.3 Approach 75
6.3.1 Spectral Learning 76
6.4 Weakly Labeled Data 77
6.5 Results 80
6.6 Conclusions 81

7 MOOC Thread Structure Identification **83**
7.1 Introduction 83
7.2 Weakly Labeled Dataset 85
7.3 Method 87
7.4 Results 89
7.5 Conclusion 90

8 Conclusions and Future Work **93**

Bibliography **97**

List of Figures

2.1 Graphical Dependencies of a Model Without Latent Variables (Left) and With (Right) 5
2.2 Running time of EM compared to spectral method as function of the size of training set (left [64]) and the number of hidden states (right [14]). The Y axis obeys a logarithmic scale in both figures. 7
2.3 Discourse Relations in a Two Party Conversation 10

3.1 Some examples of weakly labeled datasets 22
3.2 Simulated HMM Results in the Noise-Bias Spectrum 25
3.3 Predictive Accuracy of HMM with Bootstrapping and Cross Training with a CRF 27
3.4 Effects of Internal Error Correction 29
3.5 Effects of Training Weights on Predictive Performance 33
3.6 Combined Data Quality Improvements for Biased Dataset (Left) and Noisy Dataset (Right) 35
3.7 Combined Data Quality Improvements for Dataset with Noise and Bias 36
3.8 Empirical Noise and Bias Across Four Applications 38

4.1 The InContext framework 45
4.2 User interface for data collection 46
4.3 InContext Data in the Noise-Bias Spectrum 48
4.4 Comparison of preference classifiers 56
4.5 Active learning curve 57

5.1 A researcher demonstrating the use of the Virtual Coach system 61
5.2 Weakly Labeled Virtual Coach Data in the Noise Bias Spectrum 64
5.3 A Decision Tree Showing the Compliance Predictions of a User 67
5.4 Empirical Training Curve 68

6.1 Overview of all relations occurring in the Penn Discourse Treebank 74
6.2 An example of the latent variable discourse parsing model taken from the Penn Discourse Treebank Dataset. The relation here is an example of a cause attribution relation. 75
6.3 Empirical Arrangement of Datasources in the Noise-Bias Spectrum 77
6.4 Empirical results for labeling of implicit relations. 80

7.1 Noise-Bias Spectrum of Weakly Labeled MOOC 86

7.2 Empirical Prediction Results Using Coursera Data 89

List of Tables

3.1 Types of Datasets Created for Simulation Results 34

4.1 The representation of a user's context 47
4.2 Voice activity detection features 51
4.3 Voice activity detection accuracy 51
4.4 Effects of voice activity detection and phone posture recognition (Information extraction) 56

5.1 The features computed using the Virtual Coach sensors 62
5.2 Effects of Data Improvement Methods for the Virtual Coach 69

7.1 MOOC Datasets 85
7.2 Predictive Precision Results 90

Chapter 1

Introduction

The previous thirty years have represented the most staggering exponential growth of recorded information in human history. In 1984, only 1,000 computer hosts were connected to the internet. By 2012 this number had risen to 2.1 billion. In 2004, Facebook was launched in a college dorm-room. By 2010, 3.5 billion pieces of content were being shared on Facebook in a single week—by 2011 this number had doubled again to 7 billion pieces of content per week. Data has become the most valuable commodity of the information age—data in such great volume that no team of human analysts could ever hope to learn anything significant without the use of automated data analysis methods. However, the vast majority of data on the web is unstructured and unlabeled, whereas many state of the art machine learning techniques continue to rely on human annotated datasets. This leads to unrealized potential, wherein learning algorithms are training on datasets that account for tiny fractions of the data that is available. To give a concrete example, we may consider a recently released dataset containing human annotations denoting the activity of tigers or dogs in high definition video [18]. This dataset contains annotated video, indicating when the animals are performing specific actions, such as walking or jumping. The total length of video content in this dataset is two hours, meanwhile, five hours of video footage is uploaded to YouTube *every second.* If we could extract a training signal from even one in every half billion videos on Youtube, we could build a training set that would dwarf the size of a hand-labeled dataset.

In this thesis, we explore a variety of techniques for reducing the need for human supervision in machine learning, ranging across a variety of real world applications. Fully labeled data has been shown to be very useful for many learning applications, but labeled examples generally only represent a tiny fraction of the available volume of data. Unlabeled datasets, on the other hand, are widely available but are much less useful for most learning tasks. We explore a middle ground between these two extremes, *weakly labeled* data, which can be labeled using automated heuristic methods and subsequently used for training. We introduce the Noise-Bias spectrum as a framework for formalizing the quality of weakly labeled data. In this space, we identify the noise of the training labels and the bias of the feature space distribution of the training data as properties of a weakly labeled dataset. Noise is defined simply using the probability that training labels differ from the ground truth, and bias is measured using the KullbackLeibler divergence in the feature space between the weakly labeled data and the testing environment. We can compensate for noise with the use of Internal Error Correction (IEC), which allows a model to reclassify points in the training set that are believed to be erroneously labeled. To combat bias, we can employ density sampling, which will allow us to select training data that most closely reflects the data the model will be asked to make predictions upon.

We also discuss the use of unlabeled data in learning *latent variable models* over high dimensions. When using unlabeled data, we investigate some of the classical approaches of bootstrapping and cross-training. Unfortunately in many instances these approaches simply allow a model to self-reinforce its existing beliefs, limiting the effectiveness of these approaches. However, unlabeled data can still be quite useful. Often when learning latent variable models, empirical n-grams must be computed using training data, and using unlabeled data is an extremely effective way to reduce the sparsity of these matrices.

The applications that we explore are quite computationally difficult due to the high volume of training data afforded by the use of weakly labeled data, coupled with the propensity towards local optima in optimization created by the latent variable approach. Fortunately, a class of algorithms known as *spectral methods* are uniquely equipped to deal with both of these problems. Spectral methods are statistically consistent, meaning they are not susceptible to local optima, and in practice they are orders of magnitude faster than other comparable methods. Using spectral methods, coupled with weakly labeled and unlabeled data, we will present results from a variety of applications.

In the personalized healthcare space, we discuss the *virtual seating coach*, a platform that monitors the pressure relief habits of power wheelchair users. We present a personalized learning algorithm that uses GPS, accelerometer, temperature, time of day, and day of the week to identify contexts in which a user is most likely to conduct pressure relief, allowing us to intelligently remind users to perform exercises in situations that are most convenient.

In the natural language processing domain, we consider textual discourse parsing. Discourse parsing is the process of discovering inter-sentential relations of a document, for instance when one unit of text provides evidence for a claim made elsewhere. For this task, we leverage large unlabeled and weakly labeled datasets, allowing us to significantly outperform traditional fully supervised parsers.

Related to the discourse parsing application, we also consider learning response relations of posts between students participating in a Massively Online Open Course (MOOC). As the number of students in a MOOC grows, it becomes increasingly difficult for faculty and paid instructors to provide all of the academic support needed by the students. This means it is vital to encourage students to support one another. By applying our discourse parsing methods to conversations between students, we are able to identify the structure of forum threads, which could be used to identify productive conversations or potentially helpful student mentors.

In each of these applications, the use of weakly labeled and unlabeled data allows us to achieve significant performance gains. In some of these applications, such as textual discourse parsing, large labeled datasets exist, but our approaches significantly outperform fully supervised methods. In other applications, such as learning response relations in a MOOC, labeled datasets do not exist—all but necessitating the use of weakly labeled data. We identify the position in the Noise-Bias spectrum for each source of weakly labeled training data across each application. Coupled with experiments using simulated training data, we are able to make several generalizable observations regarding the efficacy of using weakly labeled training data in various points of the Noise-Bias spectrum and the best methods to use for each region. Taken together, these results represent a significant contribution towards reducing the need for human annotation, in an age when unlabeled data is becoming ever more prevalent.

Chapter 2

Related Work

2.1 Background

2.1.1 Latent Variable Methods

Latent variable models, such as Hidden Markov Models and Conditional Random Fields, have long been used successfully in many difficult prediction and modeling tasks, particularly when observations appear in continuous sequences. For example, we may have a sequence of observed symptoms from a patient in the hospital, while the underlaying condition of the patient's health remains unobserved. By explicitly assuming the existence of these hidden states, we can improve predictive capability without needing to statistically model the joint distribution of the entire sequence of observations.

Figure 2.1.1 provides an illustrative example of latent variable models, using the symptoms

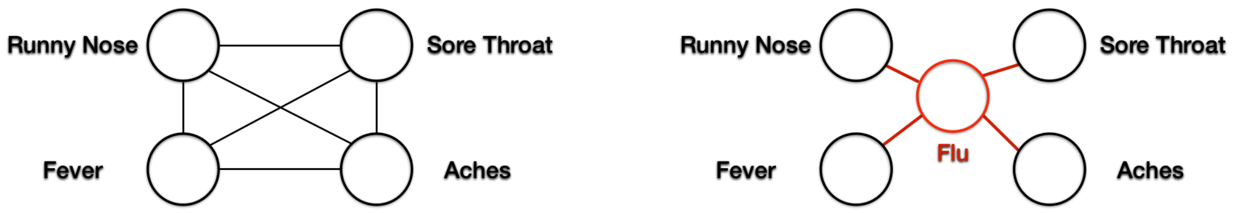

Figure 2.1: Graphical Dependencies of a Model Without Latent Variables (Left) and With (Right)

of a patient in a hospital as observed variables. Without latent variable models, the 4 symptom variables we observe are dependent on one another, so we must model the joint distribution $P(\text{Runny Nose, Sore Throat, Fever, Aches})$. However, if we assume the existence of a latent variable that indicates if the patient has the flu, these variables become conditionally independent given the flu variable. This simplifies the problem computationally when compared to working with the joint distribution, but this model is also significantly more expressive than one that assumes each variable is strictly independent.

Unfortunately, introducing latent variables to our representation significantly complicates the optimization problem entailed in learning from data. Due to the many possible ways we may credit empirical observations to unobserved latent variables, this optimization space include many local optima that can make traditional approaches like Expectation-Maximization ineffective. Fortunately, spectral methods, which are not prone to local optima, have been developed in recent years for many different latent variable models.

2.1.2 Spectral Hidden Markov Model

The addition of weakly labeled and unlabeled data can result in an exponential increase in the size of our training sets. This can lead to intractable computational requirements for large scale problems, such as mining text data from the web. This necessitates the use of more computationally efficient optimization procedures. There has recently been growing interest in a breed of algorithms based on spectral decomposition, which are well suited to training with unlabeled data. Spectral algorithms utilize matrix factorization algorithms such as Singular Value Decomposition and rank factorization to discover *low-rank decompositions* of matrices or tensors of empirical moments. Spectral algorithms tend to be much faster—sometimes orders of magnitude faster—than competing approaches, which makes them ideal for tackling large datasets. Figure 2.1.2 compares the running time of a spectral method and an Expectation Maximization (EM) approach. The left figure shows that when training a Hidden Markov Model (HMM), EM

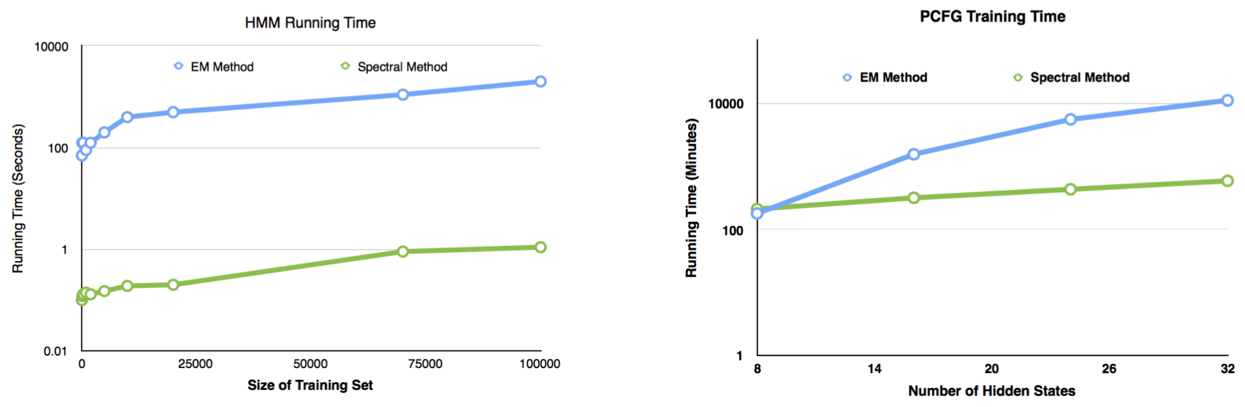

Figure 2.2: Running time of EM compared to spectral method as function of the size of training set (left [64]) and the number of hidden states (right [14]). The Y axis obeys a logarithmic scale in both figures.

runs in exponential time as a function of training set size, while the spectral method is roughly linear with regards to the number of examples in the dataset [64]. This results in a maximal running time of 33 minutes for the EM algorithm, and just 1 second for the spectral method. Similarly, the right figure shows that EM runs exponentially in the number of hidden variable states when training a Probabilistic Context-Free Grammar (PCFG) [14]. The worst case running time discrepancy in this case is 187 hours with EM and 9 hours with a spectral method.

This tremendous discrepancy in running time compared to traditional methods allows us to overcome the increase in training set size required to include unlabeled and weakly labeled data in our training sets. Spectral methods also have an additional benefit of statistical consistency, meaning that they avoid local optima during training. This often results in a slight increase in classification accuracy, on top of the exponential decrease in running time [14].

Spectral methods have been developed for a variety of tasks, including topic modeling, Probabilistic Context Free Grammars (PCFG), data clustering, and Hidden Markov Models (HMMs). In this section, we will describe the spectral learning process for a Hidden Markov Model, which is an approach we will use for many sequential learning problems encountered in this thesis. One of the first algorithms for spectral learning of HMMs was introduced in [32], and this is the approach described in this section.

Our specified model requires learning three parameters: the initial state distribution $\vec{\pi}$, the state transition matrix T, and the observation matrix, O_{ij}, for observation i and latent state j. These parameter matrices define the model completely, but are difficult to learn directly. Instead, the spectral formulation of the model has us learning the *observable operators* of the model in a reduced dimensionality subspace. If we define the matrix A_x as

$$A_x = T\text{diag}(O_{x,1}...O_{x,m})$$

Then the following equality holds:

$$Pr[x_1...x_t] = \vec{1}_m^T A_{x_t}...A_{x_1}\vec{\pi}$$

Using this formulation of the model, we need to learn the matrix A_x and the vector $\vec{\pi}$. To compute these parameters, we use the unigram, bigram, and trigram probability matrices. We denote the unigram matrix as P_1, the bigram matrix as $P_{2,1}$, and the trigram matrix as $P_{3,x,1}$, with one trigram matrix for each value of x. Define these matrices as follows:

$$[P_1]_i = Pr[x_1 = i]$$

$$[P_{2,1}]_{ij} = Pr[x_2 = i, x_1 = j]$$

$$[P_{3,x,1}]_{ij} = Pr[x_3 = i, x_2 = x, x_1 = j]\ \forall x$$

Spectral latent variable models utilize subspace identification to learn the model dynamics in a reduced dimensionality space. If we assume that the dimensionality of this subspace is no less than the rank of the parameter matrices O and T, then the model learned in the subspace is equivalent to the model from the original feature space. There are different methods of projecting the data into this subspace, but one convenient approach is to take the left Singular Values of the bigram matrix $P_{2,1}$. We denote this subspace transformation matrix as $U \in \mathbb{R}^{n \times m}$. If we denote the subspace model parameters as π_U, T_U, and A_U, these parameters can be easily learned using factorization over the fully observable matrices defined above. Proofs of all equalities given in this section are available in [32].

$$\hat{\pi}_U = U^T P_1$$

$$\hat{A}_U = U^T P_{3,x,1} (U^T P_{2,1})^+ \; \forall x$$

We see here that computing the parameters of the spectral HMM only requires a series of matrix operations and the computation of the singular values of the bigram matrix—meaning the parameters can be computed extremely quickly, particularly on parallel or distributed hardware. However, the empirical n-gram matrices can be very sparse for high-dimensional problems, even in the reduced dimensionality subspace. And yet, if class labels are not included as part of the observation space, then unlabeled data can be used to compute the empirical unigram, bigram, and trigrams. For applications with large volumes of unlabeled data, such as text parsing applications, this approach can yield stable and accurate estimates of the empirical n-grams.

2.1.3 Discourse Analysis

Discourse parsing is a fundamental task in natural language processing that entails the discovery of the latent relational structure in a multi-sentence piece of text. Unlike semantic and syntactic parsing, which are used for single sentence parsing, discourse parsing is used to discover inter-sentential relations in longer pieces of text. Without discourse, parsing methods can only be used to understand documents as sequences of unrelated sentences. Discourse relations may include explanatory, contradictory, or elaboration relations. Figure 2.1.3 shows some possible discourse relations between utterances occurring in a conversation between a student and a teacher.

Discourse parsing can be reduced to three separate tasks. First, the text must be decomposed

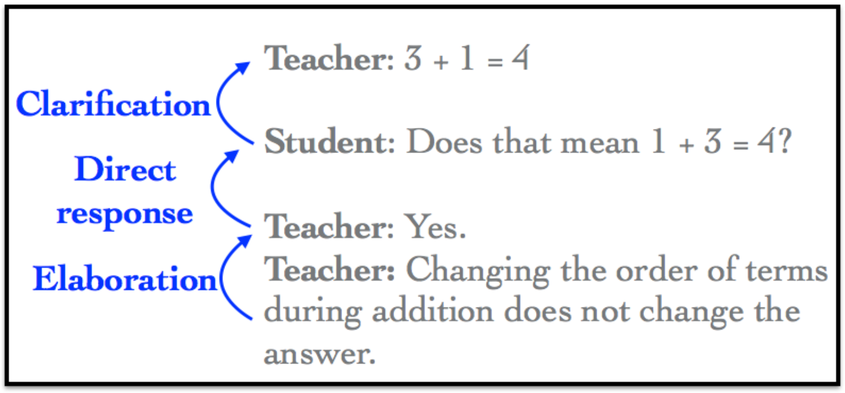
Clarification
Direct response
Elaboration
Teacher: 3 + 1 = 4
Student: Does that mean 1 + 3 = 4?
Teacher: Yes.
Teacher: Changing the order of terms during addition does not change the answer.

Figure 2.3: Discourse Relations in a Two Party Conversation

into *elementary discourse units*, or *EDUs*, which may or may not coincide with sentence boundaries. The EDUs are often independent clauses that may be connected with conjunctions. After the text has been partitioned into EDUs, the discourse structure must be identified. This requires us to identify all pairs of EDUs that will be connected with *some* discourse relation. These relational links induce the skeletal structure of the discourse parse tree. Finally, each connection identified in the previous step must be labeled using a known set of relations. Examples of these discourse relations include concession, causal, and instantiation relations. In some frameworks, such as the Penn Discourse Treebank (PDTB), only adjacent discourse units are connected with a discourse relation, yielding parse sequences rather than parse trees. In other approaches, such as Rhetorical Structure Theory (RST) more complex discourse structures are permitted.

Some relations are induced by specific connective words in the text. For example, a contrast relation may be explicitly revealed by the conjunction *but*. Sentence (1) demonstrates an explicit relation with two EDUs, connected by a coordinating connective.

(1) "The rapid expansion of the Bitcoin market initially led to huge gains, **but** recent electronic attacks of prolific banking services have resulted in a catastrophic free-fall of the digital currency's value".

Simple classifiers using only the text of the discourse connective with POS tags can find explicit relations with high accuracy [43]. For comparison, sentence (2) shows an example of the more difficult implicit relation. In this sentence, two EDUs are connected with an explanatory relation, shown in bold, although the connective word does not occur in the text.

> (2) “But a few funds have taken other defensive steps. Some have raised their cash positions to record levels. **[BECAUSE]** High cash positions help buffer a fund when the market falls.”

Many relations can be expressed in either implicit or explicit forms, however some relations, such as response relations, only occur implicitly. Most discourse parsing methods are primarily devoted to the more difficult problem of learning implicit relations. However, explicit relations identified with a simple classifier make for a high volume dataset with very little noise, although this type of training data will only represent a biased region of the feature space due to the exclusion of implicit relations.

2.2 Related Work

Researchers have been working with weakly labeled data for decades, although the notion of weakness is often confined to noise in the training data labels. The generalized expectation criteria was introduced as one of the early efforts to exploit the specific structure of weakly labeled data using expectation constraints [46]. This work demonstrated that weakly labeled data can be made much more useful when it is not simply used as labeled data. Similar approaches using weakly labeled data have been successfully employed for tasks ranging from visual object classification [6] to logical relation extraction from unstructured text [51]. An increasingly popular source of weakly labeled data is crowdsourcing, in which paid annotators remotely perform labeling micro-tasks for clients. Research has shown that the effectiveness of these services is extremely sensitive to the design of the micro-task being performed and the incentive levels offered to annotators [38]. These labels are also inherently noisy, with some researchers suggesting that getting a higher volume of noisier labels from annotators is more desirable for all parties involved [39], because of validation tools such as inter-annotator agreement. Unfortunately, this kind of approach is non-viable for tasks that require a high level of annotator expertise.

In this thesis, we will be examining four applications in detail. The first application is determining smartphone interruptibility. Researchers have been studying interruptibility in the field of human-computer interaction for many years. This research was initially in the domain of desktop computers, when multi-tasking applications began introducing irritating interruptions to users while they worked. Early research focused on using only information about the state of the user's software to determine interruptibility [60], but more recent work has instead been using sensors to perceive the user's environment in order to achieve context-aware interruptibility [26, 30]. The growing popularity of mobile devices has reinvigorated interest in determining interruptibility. Many users have a consistent internal model of when and why they want their mobile device turned on [68], but many of us often forget to change the device's settings at the appropriate times. For many users, autonomously learning their preferences requires the ability to sense factors in the user's current environment. To achieve this, many previous context-aware systems have required users to place wearable sensors around their body [29, 65]. However, modern smartphones, like the Apple iPhone or Google's Android platform, feature an array of useful sensors that can allow us to circumvent the need for specialized hardware. Only a small number of systems have been designed to leverage the power of these new hardware platforms when predicting interruptibility [57].

In the next application, we extend the ideas from smartphone interruptibility to build a system to help prevent pressure ulcers for power-wheelchair users. There has been some previous work utilizing machine learning to predict patient behavior in healthcare applications, but rarely with the fine level of granularity of an interruptibility system. For instance, one author used statistical machine learning techniques with patient data to predict which subjects suffering from coronary artery disease would be likely to comply with pharmaceutical guidelines for managing cholesterol levels [20]. These predictions were based primarily on demographic data, and did not give indications as to which contexts and circumstances would lead to non-compliance for a subject. There has also been quite a bit of work using machine learning to predict health care outcomes and complications, but this work has not been focused on understanding or altering patient behavior to improve outcomes [16, 66].

We then turn to the task of discourse parsing with the Penn Discourse Treebank (PDTB). There has been quite a bit of work concerning fully supervised relation classification with the PDTB [22, 44, 69]. Semi-supervised relation classification is much less common however. One recent example of an attempt to leverage unlabeled data appears in [28], which showed that moderate classification accuracy can be achieved with very small labeled datasets. However, this approach is not competitive with fully supervised classifiers when more training data is available. Recently there has also been some work to use Conditional Random Fields (CRFs) to represent the global properties of a parse sequence [23, 34], though this work has focused on the RST-DT corpus, rather than the PDTB. In addition to requiring a fully supervised training set, most existing discourse parsers use non-spectral optimization that is often slow and inexact. However, there has been some work in other parsing tasks to employ spectral methods in both supervised and semi-supervised settings [15, 52]. Spectral methods have also been applied very successfully in many non-linguistic domains [8, 25, 32].

Finally, we extend the discourse parsing framework to analyze discussions between students in an online course. Education research has demonstrated that characteristics of dialogue and discourse between students can be used to evaluate the effectiveness of the discussion and predict student outcomes [1, 48, 58]. However, these approaches often either rely on human supplied dialogue tags, or fully supervised discourse parsers that require large, labeled datasets. As such, these systems show great promise on the domains in which they are deployed, but they are often not generalizable.

Throughout this work, we leverage spectral methods to combat our large, weakly labeled datasets. Spectral methods have become a very popular topic in machine learning research. Based on eigenvector decomposition, such as is used in Singular Value Decomposition (SVD) and Principle Component Analysis (PCA), spectral methods can give optimal results for many optimization tasks using a small fraction of the computation required by comparable algorithms.

Spectral algorithms exist for learning latent-variable PCFG's [13], dynamical systems [9], and Hidden Markov Models [32]. From an applications standpoint, spectral methods have been used to estimate a student's aptitude for key tasks in a classroom setting [21], for dependency parsing of natural language text [19], and for image segmentation and classification tasks [50].

Chapter 3

Methodology

In this chapter, we introduce the broad principles that motivate this thesis. We begin by discussing the most common current approach to machine learning, learning with fully labeled data. We will briefly consider some of the ways that fully labeled data can be acquired, and discuss some circumstances in which human annotated data is either undesirable or completely inviable. Next we will discuss the most prominent alternative to fully labeled data considered in this work, *weakly labeled data*. Weakly labeled data consists of examples that have somehow been labeled or organized in a fashion that does not meet the gold standards of human annotated data. These datasets can often be acquired through autonomous heuristics and have the potential to be much larger than a hand labeled dataset. We leverage the notions of *bias* and *noise* as metrics to measure the quality of weakly labeled data and present a simulated framework with which to explore the noise-bias spectrum. We then present introductory, simulated results to motivate the ideas described in the remainder of the thesis. Throughout this work, we utilize spectral latent variable models, a class of algorithms that are well positioned to deal with the unique challenges inherent in working with large volumes of weakly labeled and unlabeled data.

3.1 Acquiring Annotated Data

Human annotated data continues to be the gold standard in quality for prediction, structure-learning, and language annotation tasks. However, when dealing with large datasets, annotation becomes the most significant bottleneck in an analysis pipeline. Hiring a team of devoted human annotators can yield datasets containing millions of labeled examples—for instance with part of speech tagging in the Penn Treebank [54]. Unfortunately the time and financial investments required to create such a dataset are prohibitive for most applications, and even the largest such datasets pale in comparison to the overwhelming volume of unlabeled data freely available online. In recent years, crowdsourcing techniques have become increasingly popular methods to acquire annotated data. By utilizing large, distributed networks of annotators, these approaches can produce labeled datasets much larger than what is capable with a small, professional team of experts, potentially at the cost of label quality. This places crowdsourced data firmly in the realm of weakly labeled data. Noise in label noise could be very common, depending on the difficulty of the task, and bias may be introduced if annotators are able to select only a subset of tasks to complete. Throughout this section, we will identify some of the challenges of acquiring labels from human annotators, in particular for some applications in which crowdsourcing is not viable.

3.1.1 Crowdsourcing

Crowdsourcing represents perhaps the most promising method of labeling and organizing unstructured data for large-scale use in machine learning analysis. These systems can be partitioned into service-driven and community driven platforms. Each with its own advantages and disadvantages.

With a service driven platform, such as Amazon's Mechanical Turk, a client contracts human workers to label, segment, or otherwise annotate units of data. This approach has been used successfully to create datasets for machine translation, identify spatial usage of social network, and to simply encourage anonymous volunteers to go out and perform a good deed. In general, the more expertise a task requires of the annotator the more expensive an individual annotation will become and fewer annotators will become available. Additionally, many clients of service driven crowdsourcing platforms complain that their annotators prefer to produce annotations as quickly as possible, leading to compromised work quality compared to in-house professional annotators—which generally requires the client to request multiple labels for each datapoint in order to measure inter-annotator agreement.

For applications in which significant resources can be devoted to annotation tasks that require little expertise, service driven crowd-funding can be an excellent way to acquire small-to-medium sized labeled datasets quickly. However, both of these constraints can prove restrictive in many applications. An alternative is community driven crowdsourcing, in which volunteers from across the web can volunteer their time, expertise, or personal data. Wikipedia is perhaps the most well-known such platform, and it has been shown that Wikipedia articles contain fewer errata on average than the Encyclopedia Brittanica (however, those errors in Wikipedia tend to much more egregious and are occasionally committed maliciously) [27]. Community driven crowdsourcing has also been used in more specialized applications, such as the Tiramisu platform which tracks public buses in realtime by monitoring location data from users of the application [72]. Similarly, Google leverages location data from users of its Android mobile to improve predictions of traffic conditions on the road.

The disadvantages of community driven crowdsourcing are twofold. First, we face significant privacy concerns when a user's data is being collected and analyzed without their knowledge. Although consent is generally collected through End User License Agreements, research has shown that only a small fraction of users read through the details of these agreements. The second major concern with community driven crowd-funding is the need for a large community—which introduces a quandary when a platform requires user data to provide a valuable service, but requires a valuable service to attract users. Companies with large install bases, such as Google, Facebook, and Apple view their user data as one of their most valuable assets, an asset that is unavailable to smaller operations. Also, while services such as Wikipedia are a testament to the potential of crowdsourcing, history has shown that the vast majority of crowdsourcing platforms invariably fail to establish a significant user base.

There are also some crowdsourcing services that exist somewhere between service driven and community driven, such as Duolingo, which allows its users to access foreign language educational material in exchange for selling the translations they produce to outside sources. These platforms can operate by providing a service to users, which they may be willing to pay for with their efforts or personal data. A platform like this requires non-monetary incentive for users to participate, which restricts this approach to a relatively small set of applications.

With the right conditions, crowd-sourced methods can produce tremendously sized datasets with varying levels of structural annotation. However, significant resources or an established user-base are required to successfully crowd-source a dataset. This represents a significant limitation for the portability of these platforms, leaving many applications to rely on in-house annotators.

3.1.2 Expert Domains

For some applications, the expertise required to supply annotations greatly restricts the volume of labeled training data that may be acquired. For example, if we wish to determine whether a time series of blood pressure readings for a congestive heart patient is abnormal, we would need a cardiac physician or other qualified clinician. This makes crowdsourcing services infeasible, and greatly adds to the cost of in-house annotation. There have been some recorded instances in which crowdsourcing difficult problems with relatively uninformed participants can lead to viable prediction. Examples include estimating the commercial viability of a film based on its theatrical trailer [7], predicting odds of presidential candidates winning elections using a Las Vegas style betting system [11], and determining protein folding sequences [36]. In hese instances, many thousands of non-expert volunteers must participate to produce a prediction for a single datapoint, which makes this approach nonviable in high volume applications, such as healthcare or education.

3.1.3 Personalized Models

Another class of applications in which annotated data is difficult to acquire are personalized models, such as recommendation systems or personalized healthcare support. In the most extreme case, a separate model may need to be trained for each user, which implies that we must rely on a single—potentially uncooperative—annotator. Approaches such as collaborative filtering seek to leverage users with similar interests to train a model for an individual. However, these models assume that the user is willing to provide some information about their preferences, such as ratings for products or services. Many users may be unwilling to provide this information, leaving us only with weakly labeled and unlabeled signals derived from observations of their activity—such as viewing habits through a streaming service, or usage patterns for a smartphone.

3.2 Weakly Labeled Data

In many applications, we may wish to consider data that lies somewhere between data that has been fully annotated by a human and data that has no annotations or demarcated structure. In particular, data that has been partially or fully labeled using an intermediate, automated heuristic can benefit from the volume of unlabeled data and the usefulness of fully labeled data. In this section, we will consider *weakly labeled* data, meaning data that has been annotated in some way, but does not meet the gold standard of human annotated data.

For example, imagine that we wish to train an image processing algorithm to detect the presence of various mammals in photographs. We could scrape all images from a cat enthusiast website and declare that all of these images contain the presence of a cat. Alternatively, we could use a separately trained model to predict the presence of animals in a random set of training images. We would not expect either of these sources of training data to work particularly well in real world conditions, but next we will consider the specific characteristics that make them undesirable for training.

To speak more precisely about weakly labeled data, we propose two simple metrics to measure the quality of a training set. The first, and most obvious, is label noise, referring to the percentage of time the weak label assigned to a training point correctly reflects the true label—according to a human expert. Returning to the examples given in the previous paragraph, the cat images would likely have very little labeling noise if the assumption holds that most images on a cat enthusiast website contain cats. In the second scenario, in which predicted labels from another algorithm are used, the amount of noise would be proportional to the predictive accuracy of the model used, which presumably would be quite high in a difficult task such as visual entity recognition. If we denote a dataset, X, with a set of weak labels, $\hat{Y}$, and denote Y as the set of ground truth labels for this data, our measure of noise is simply the empirical mean accuracy of the labels in the dataset.

$$\text{Noise}(X) = \frac{1}{n}\sum_{i=1}^{n} I(\hat{y}_i = y_i)$$

The second metric of data weakness is *bias*, that is, how accurately the distribution of the training data over the feature space reflects the distribution we will be evaluating against. By measuring bias in the feature space, we are also able to detect bias in the label space—assuming that classes are cleanly separated in the feature space. Returning to the previous examples, the cat images would be extremely biased, because they only reflect a single predicted class taken from a very small subset of the set of all possible images. The bootstrapped images could have very low bias in the feature space, but may show bias in the labels assigned to them. Under this formulation, unlabeled data becomes weakly labeled when bootstrapping is applied to it. To formalize the notion of bias, we refer to the empirical Kullback-Leibler divergence (KL divergence) of a training data set compared to the testing set or equivalent deployment environment. For training set X and testing set X', we can define the empirical KL-divergence as follows:

$$D_{KL}(X||X') = \sum_{x_i \in X'} P([x \in X] = x_i) \log \frac{P([x \in X] = x_i)}{P([x' \in X'] = x_i)}$$

It should be noted that in sparse, high-dimensional spaces, the preceding estimate will suggest an inaccurately high level of divergence. To compensate, we may conduct the KL divergence estimate in a subspace generated by Singular Value Decomposition (SVD) or Principle Component Analysis (PCA). Alternatively, we may wish to create a continuous estimate of the feature space distributions using a non-parametric Kernel Density Estimate (KDE), which would allow us to employ the continuous definition of the Kullback-Leibler divergence. This continuous estimate could be computed in the original space, or in a reduced dimensional subspace.

In using these two metrics to measure bias we seek to maintain a balance between expressivity and simplicity in our definition of weak data. We shall see that this spectrum captures a very wide variety of weakness in data, although one could imagine many different types of problematic datasets that do not conform to these definitions. For example, we may have noise in the feature space, the labels of the training data may be noiseless indications of an attribute that is related—but not equivalent to—the label we are interested in, or we may see a significant volume of missing attributes in the training data. However, in this work we will restrict ourselves to the definition of noise and bias given above, in the hopes of presenting the simplest framework that is able to describe a wide variety of weak datasets.

For any given training set, we can plot that set in a two dimensional space, where one axis represents the percentage of labels in the training set that are incorrect, and the other axis represents the KL divergence of the training set, measuring bias. Figure 3.2 shows what we might expect to see if we use this space to plot the two training sets described in the image mammal detection task. We also denote the ideal training set, with no noise and no bias, in the upper right of the plot.

The framework presented here is quite simple, which also allows it to be quite general. We could cast most conceivable training sets into this space, either empirically or hypothetically.

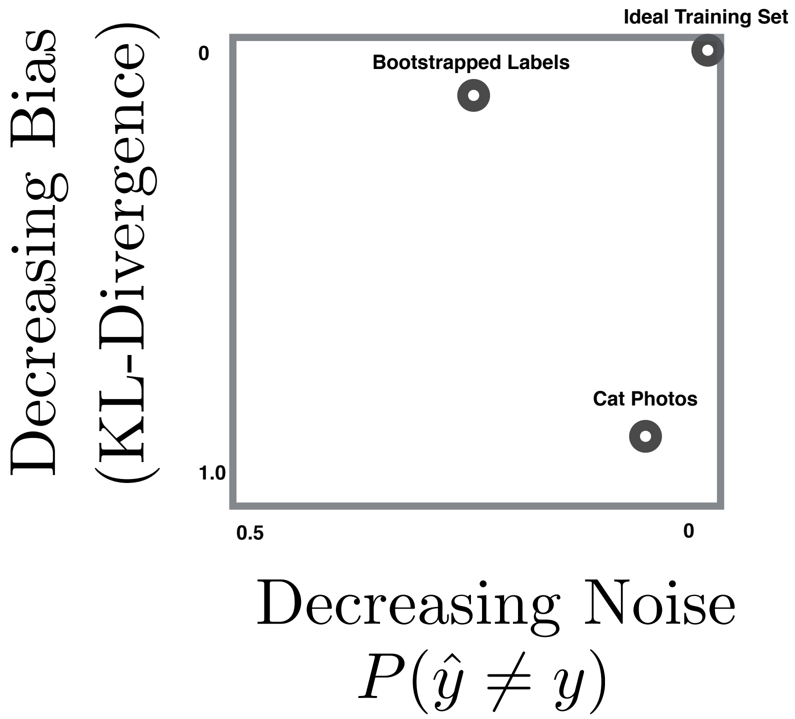

Figure 3.1: Some examples of weakly labeled datasets

This framing device also raises some interesting questions regarding weakly labeled data. Obviously some portions of this space are more desirable than others, but at which point does learning become intractable? Are there specific techniques we can employ with certain types of weak data that can alleviate some of the problems with the training data? Do different training models perform better or worse with weak data?

In the following chapters we will address some of these questions through work with synthetic experiments and several real world applications. First, we will introduce some of the tools at our disposal to deal with the challenges inherent in these learning tasks. We will first consider one of the practical side effects of incorporating weakly labeled and unlabeled data into our training sets: the significant increase in the size of our training sets. In the next section, we introduce *spectral methods*, a class of algorithms that are unusually well suited to dealing with large, partially labeled training sets.

3.3 Simulation Environment for Weakly Labeled and Unlabeled Data

To further explore the effects of bias and noise in weakly labeled data, we have created a simulation that allows us to control these parameters and determine the best methods for a given learning environment. In this section, we present the basic parameters used to generate sequential training data with latent behavior.

We will begin with a simple binary prediction task with a linear model. In particular, we will have d total features, $x_i ... x_d$. Each feature also has a corresponding coefficient c_i, which is unobserved in the training data. The label for a given training point is induced by the following summation:

$$y_i = I\left(0 \leq \sum_{i=1}^{d} c_i x_i\right)$$

Each $x_i \sim N(\mu_i, 1)$, where each μ_i and c_i are themselves drawn from $N(0, 1)$. We now considering adding noise and bias to an artificially generated training set. A given noise level $p = P(\hat{y} = y)$ can be easily created by independently changing the labels in the training set according to a coin of bias p. To achieve a desired KL divergence, we can add noise $\epsilon_i \sim (0, \sigma^2)$ to each mean μ_i after generating the testing set, but before generating the training set. In this case, the KL divergence between the testing and training set will be proportional to σ^2.

We now wish to extend this model to incorporate sequential, latent information, For each datapoint in a sequence, we include a hidden state $h_i \in [1, N]$. We now have N parameters for each feature, meaning $x_i \sim \mu_i^j$ for the i^{th} feature, whenever the hidden state has value j. Additionally, rather than using 0 as the summation threshold when determining the label, we use $\delta^j \sim N(0, 1)$ whenever the hidden state has value j. The hidden state is assigned according to a randomly chosen initial state distribution π and transition probability matrix T. Just as with the non-sequential model, we can add noise or bias to the training data in this setting by randomly adjusting labels or adding noise to the training data generation parameters. We can also modify the model so that each latent state has a different noise rate p_i. Experiments with simulated data indicate that this does not dramatically impact results, compared to using a global noise rate p. Unless otherwise noted, each sequence generated is of length 20.

Using data generated from this model, we can see initial findings on the effects of weakly labeled data in the noise-bias spectrum. Figure 3.2 shows the predictive accuracy of an HMM trained with data from the sequential simulation model, trained over 10,000 iterations varying the noise, $P(\hat{Y} = Y)$, and the KL divergence between the testing set and training set. The training in these experiments included a 200 dimensional feature space, with 1 million training examples.

Unsurprisingly, extremely weak data, *i.e.* $P(\hat{Y} = Y) < 0.6$ or KL divergence above 0.8, results in a model that underperforms compared to a naive baseline that randomly assigns labels. With no noise or bias, the model attains a maximum classification accuracy above 0.90. We see here that label noise appears to have a much more significant detrimental impact on performance than feature space bias. Without noise, a KL divergence of 0.8 leads to 0.756 prediction accuracy. On the other hand, a model with no bias and 0.60 noise rate yields a model with predictive accuracy 0.61.

We will continue to refer to this simulation model as we explore methods to reduce the impact of noise and bias in our weakly labeled data. Next we will consider some of the classical methods of turning unlabeled data into weakly labeled data, bootstrapping and cross-training.

3.4 Using Bootstrapping and Cross-training for Unlabeled Data

The polar opposite of fully labeled data is fully unlabeled data. The web is rife with unstructured text, image, and video data. Unfortunately, this data also tends to be the least informative. Bootstrapping is one of the oldest and most well-known methods for trying to utilize unlabeled data. In its most basic form, bootstrapping entails training a learning algorithm using a small set of labeled data, and then using the algorithm's predictions to assign labels to an unlabeled set. In general, labels should only be assigned when a model shows sufficient certainty with regards to

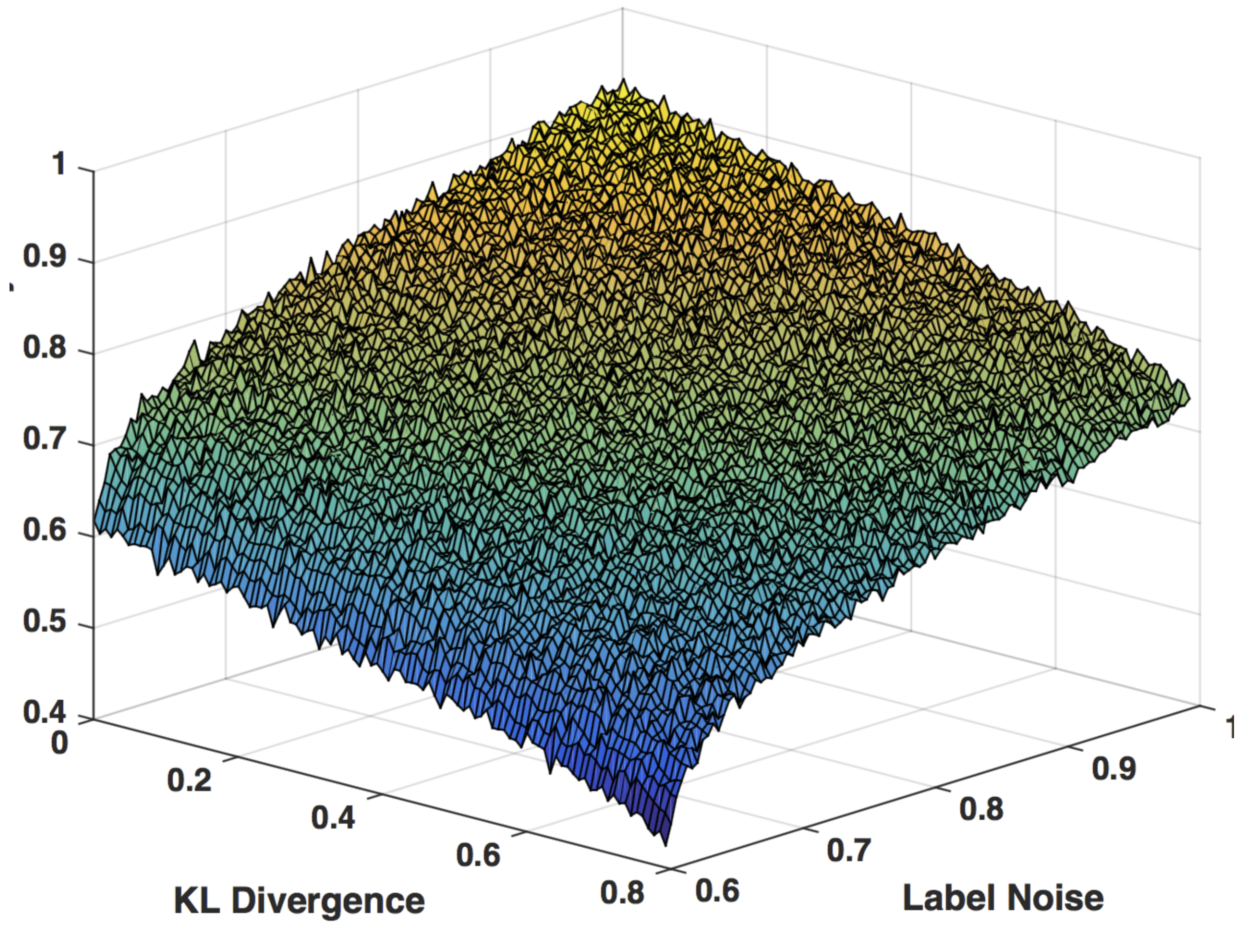

Figure 3.2: Simulated HMM Results in the Noise-Bias Spectrum

an example's label. Information theoretic entropy is a common and useful metric for measuring certainty. A classifier with a low entropy distribution over the predicted classes can be thought of as more certain about its prediction compared to a classifier with high entropy. The definition of entropy for discrete prediction is given below.

$$H(X) = -P(Y=1)\log_2[P(Y=1)] - P(Y=0)\log_2[P(Y=0)]$$

After a model is trained on labeled data, any examples in which the model's entropy falls below a given threshold, ψ, are labeled using the model's prediction, and that example is added to the training pool. This process is then repeated over several iterations with the newly labeled data added to the training set after each iteration. A common criticism regarding bootstrapping says that it is difficult for the model to generalize, as this process may simply reinforce existing errors committed by the model. A slightly alternative approach is that of cross-training. In this approach, two competing models are each trained separately on a shared set of labeled training data. The models then predict labels of unlabeled data, as in bootstrapping, but then the predicted labels from model A are fed to model B, and vice versa. The argument in favor of cross-training suggests that if we use two models that make independent errors, we can benefit from the best of both methods and avoid the self-reinforcement sometimes seen with bootstrapping. Figure 3.3 shows simulated results with bootstrapping and cross-training with data generated by the sequential simulation model described in section 3.3. In these tests the HMM is cross-trained with a Conditional Random Field (CRF). These tests were conducted with 200 dimensional feature space and 1,500 labeled examples without noise or bias. Figure 3.3 shows the effect of adding up to 1 million unlabeled training examples. We see that both bootstrapping and cross-training produce increased accuracy, though only by magnitudes of 0.01 and 0.02 respectively. As such, the performance of bootstrapping and cross-training are underwhelming compared to tests using weakly labeled datasets of similar sizes.

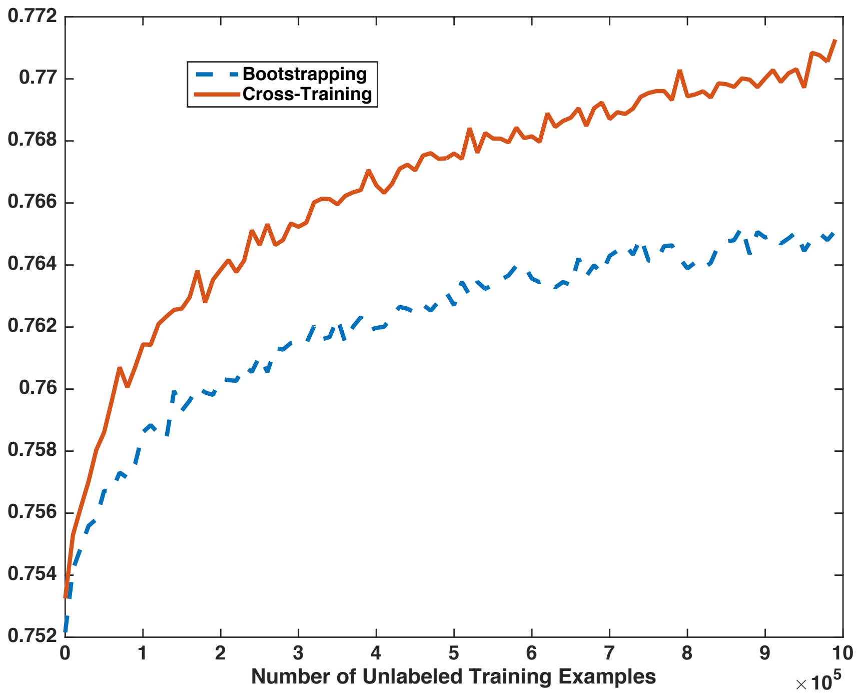

Figure 3.3: Predictive Accuracy of HMM with Bootstrapping and Cross Training with a CRF

3.5 Improving the Quality of Weakly Labeled Data

Fundamentally, weakly labeled data affords us the opportunity to access large volumes of training data, at the cost of increased noise and bias. In this section, we will discuss methods to minimize the negative impact that noise and bias cause during training, allowing us to move closer towards high volume, high quality datasets.

3.5.1 Internal Error Correction for Noisy Data

As we have seen, bootstrapping can have limited effectiveness for prediction in high dimensional space. In this section, we present a related concept that is uniquely effective for dealing with certain types of weakly labeled data. When using bootstrapping with unlabeled data, achieving improvements in predictive capability requires the model to generalize based on prior observations. This is possible in some circumstances, but often the model simply reinforcements what it already believes. Instead, we may present the model with two seemingly contradictory labeled data-points and ask which point we would relabel given the rest of the data. In this case, we are not asking the model to generalize, but instead to detect anomalies based on prior experiences, which is a more tractable problem.

In this section we introduce the notion of Internal Error Correction (IEC), in which a model may choose to relabel samples in a training set, if the model believes with sufficiently high certainty that a label is incorrect. This approach is particularly well suited to dealing with weakly labeled data that is noisy. Similar to bootstrapping, information theoretic entropy of the label prediction distribution is a convenient metric for labeling.

To conduct Internal Error Correction, we propose a procedure similar to cross validation. We begin by training the classifier on all labeled and weakly labeled data except for one held out weakly labeled training datapoint. We then compute the entropy of the classifier's predictions for the one left out example. If that entropy is below a threshold, ψ, and the predicted label disagrees with the label presented in the dataset, we modify the label of that example for future training. We repeat this procedure until all training data has been evaluated, and then retrain the classifier one last time with the corrected training data.

To evaluate this method in an ideal environment, we conducted tests with the simulated data described in section 3.3. We conducted these tests using the simulation model, and trained a Hidden Markov Model. The predictive accuracy with and without IEC is shown as a function of noise rate in figure 3.5.1. We observe that the beneficial effects of IEC are most significant when the rate of noisy labels in the training data is high. By the time the ratio of correct labels surpasses 90%, we no longer see statistically significant results. Internal Error Correction appears to be best suited to instances in which the noise rate is between 60% and 75%. In this range, the classifier without IEC initially fails to even surpass a random baseline. In the 70%-80% noise range, we see an improvement of predictive accuracy of 4-5 percentage points.

Similar to bootstrapping, we can also consider a cross-training style approach, in which a second classifier is used to conduct correction of the labels. We see a modest improvement in a dataset in which 65% of the labels are correct by training a CRF to conduct error correction. In

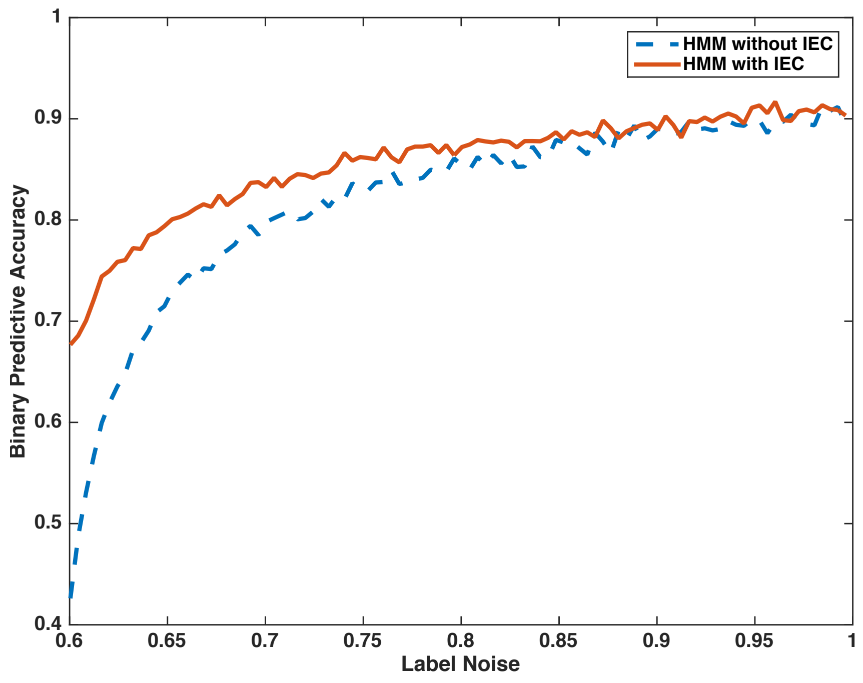

Figure 3.4: Effects of Internal Error Correction

this instance, accuracy increases from 77.5% to 78.7%, which is a significantly smaller increase in performance than we see when cross-training with bootstrapping. These results indicate that internal error correction is much less sensitive to data that is labeled in correspondence with the model's existing beliefs. This further suggests that the advantage of error correction is a reduction in contradictory training data, rather than any sort of novel discovery about the learning task.

3.5.2 Active Training for Weakly Labeled Data

Active learning is a paradigm of machine learning in which training data is actively chosen from a given source of data based upon various metrics of desirability, as opposed to traditional learning environments in which data is utilized in a static ordering [63]. In some formulations of active learning, we are given access to a *labeling oracle*, which will supply us with a noise-free label for any datapoint that is selected. This scenario can be used to represent access to a human annotator, for instance. However, in this section we will be considering the use of active learning to determine which subset of a weakly labeled dataset is suitable for training, and which subsets should be discarded or reclassified using Internal Error Correction. For weakly labeled datasets of limited size, this process can be thought of us partitioning a dataset according to quality thresholds. For larger sources of weakly labeled data, such as natural language on the web, we can instead treat the data source as if it were a continuous stream, in which each datapoint or sequence is evaluated separately.

Throughout the literature, uncertainty sampling is the most common metric for conducting active learning. In this framework, datapoints that the model has the most difficulty classifying are selecting for training, most often using entropy over the predicted class distribution to measure uncertainty. Intuitively, this allows the model to select training examples that will be the most informative. If we are dealing with a discrete classification problem with a datapoint X with classes $Y \in [1, c]$ and $\hat{p}(Y_i)$ represents the model's predicted probability that Y_i is the true label for X, then the uncertainty metric for this datapoint is as follows:

$$H(X) = -\sum_{i=1}^{c} \hat{p}(Y_i) \log \hat{p}(Y_i)$$

As a result of selecting training data with maximum uncertainty, this method will choose datapoints for training that will have the largest impact on the model's learned parameters. If the incoming data is weakly labeled, however, this can present complications. If the data source is highly noisy, an incorrect label on an uncertain datapoint may cause significant damage to predictive accuracy by pushing the learned parameters away from optimal values. If the data source is biased, the datapoint may positively improve performance, but that performance gain may not be reflected in the deployment environment. To improve the usefulness of active learning, we will consider other metrics for data selection found in the literature.

Density sampling is an alternate metric that selects training data based on its probability of occurrence in a target distribution. This approach allows us to exclude training data that would be unlikely to occur in a testing environment. This is particularly useful in instances in which the training and testing sets come from differing distributions, such as biased, weakly labeled data sources. This method requires an estimate for the distribution of the feature space in the deployment environment. If we can access a small to medium size dataset taken from the same source as our testing data, we can easily compute a non-parametric *Kernel Density Estimate* (KDE) of the distribution this data was sampled from. Kernel Density Estimation constructs the estimated distribution as a mixture of functions, such as Gaussians. The KDE is built using an estimation set and each datapoint in the estimation set contributes to the kernel function. The estimation set does not require labels, and if the estimation set is labeled, using it to build a KDE does not preclude us from also using this data during training or parameter evaluation. The accuracy of a KDE is very sensitive to a bandwidth parameter, h, that determines how much each estimation point affects the kernel function. For a Gaussian kernel, heuristic estimates of h depending on the empirical variance of the estimation set have proven to be very effective. Once a sufficient KDE has been established for the deployment environment, we may evaluate the probability of observing a weakly labeled datapoint in deployment. If the estimation set consists of datapoints $x_1 ... x_n$, and $K_h()$ represents the kernel function with bandwidth h, then the KDE probability estimate for a new datapoint X is defined as:

$$\hat{f}(X) = \frac{1}{n} \sum_{i=1}^{n} K_h(X - x_i)$$

Selecting training data that maximizes this function allows us to combat bias by discarding training data that is unlikely to be observed during evaluation. A probability threshold can be selected proportional to the size of the weakly labeled datasets. Given a stream of heavily biased data with little noise, this method can yield very large, high quality training sets.

These two metrics each have certain advantages and disadvantages. Uncertainty sampling can have a large impact on a model's parameters, but the effects may be dampened during testing if the training data is highly biased. Density sampling ensures that the training data is representative of the testing data, but the datapoints selected may not present the model with new information.

We can combine the benefits of both approaches by using a hybrid metric, *Density-Weighted Uncertainty Sampling* (DWU). For a datapoint X, the Density Weighted Uncertainty measure is defined as follows:

$$DWU(X) = \hat{f}(X) + bH(X)$$

In this equation b is a real valued parameter that allows us to control our preference for density weighting versus uncertainty sampling. The optimal value for b will be sensitive to the noise and bias of the weakly labeled training set. In particular for noisy training sets, uncertainty sampling has the potential to diminish the performance of the model, so we may wish to use a smaller value for b.

Another closely related metric is density-weighted certainty sampling (DCS), in which we use the reciprocal of the certainty term. We may wish to use this modified form if we are working with weak labels that are themselves the product of a classifier, and we would like to select those datapoints that the model is most certain about. In this special case, we have a direct estimate of the likelihood that a datapoint is noisy. In other instances, we prefer the more general form of density-weighted uncertainty sampling.

3.5.3 Learning Weights for Weakly Labeled Data

Compared to traditional supervised learning, weakly labeled data provides us with much larger training datasets, at the cost of data quality. In many instances, we will wish to use a mixture of weakly labeled datasets from multiple sources with a smaller hand labeled training set. In these instances, the volume of weakly labeled data can overwhelm the labeled data. A training procedure that does not give special consideration to the effects of weakly labeled data will learn a model that favors the larger weakly labeled datasets over the smaller labeled dataset, which can result in lower performance than a model trained using only the fully labeled dataset. To ensure that the labeled data is fully utilized during training, we introduce training weights for

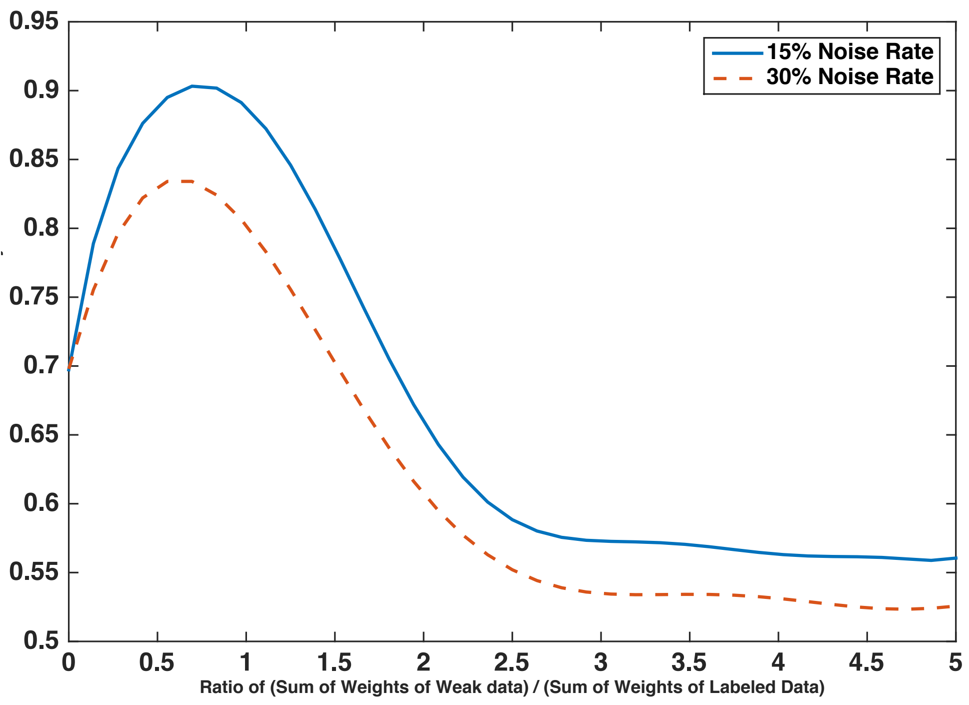

Figure 3.5: Effects of Training Weights on Predictive Performance

each datapoint. For weakly labeled data, the optimal weight of the datapoints may depend on the noise and bias of the weak data source, as well as the total volume of weak data. A convenient method of controlling for dataset size is to constrain the total sum of weakly labeled weight compared to labeled weight. In particular, we will select a value W_r to maximize predictive accuracy on a labeled parameter validation dataset.

$$W_r = \frac{\sum_{x_i \in \text{Labeled}} w_i}{\sum_{x_j \in \text{Weak}} w_j}$$

This formula allows us to assign different weights to each point in a dataset, but it is often sufficient to assign uniform weights to all datapoints for a given source.

Figure 3.5 demonstrates the effect of weight parameter selection on predictive accuracy using the simulation environment introduced in section 3.3. These results were obtained using a spectral HMM trained with 1,000 labeled examples and 10,000 weak examples using uniform

Dataset	**Noise Rate**	**KL-Divergence**	**Size**
Labeled Training	0	0	500
Weak (Biased)	0	0.4	2,000
Weak (Noisey)	.3	0	2,000
Weak (Noisy and Biased)	0.3	0.4	2,000
Parameter Validation	0	0	150
Testing	0	0	2,000

Table 3.1: Types of Datasets Created for Simulation Results

label noise. Internal Error Correction and density-weighted uncertainty sampling have both been applied. Predictive accuracy was then computed with 1,000 randomly generated datasets for each value of W_r. Figure 3.5 shows similar behavior for datasets with differing noise rates. In particular, we see peak accuracy in the range $W_r \in [0.5, 1]$, with accuracy dropping precipitously before settling asymptotically at a lower performance value than we began with. The asymptotic point in these curves roughly reflects the model's performance when trained using only weakly labeled data. Fortunately, in the region of the parameter space where the optimal value of W_r is likely to lie, these curves are convex. This allows us to use simple methods of selecting W_r such as binary search or gradient descent. Models trained with weak datasets including bias, rather than noise, produce similarly shaped training curves, with differing global optima.

Through these experimental results, we indeed observe that the optimal value of W_r for a given datasource depends on the noise, bias, and volume of the training set. If we assume a uniform or unknown distribution of noise and bias in a weak dataset, we have no reason to assign different weights to different datapoints from a given source. This means that each datapoint from a source of size d will have weight $\frac{W_r}{d}$.

3.5.4 A Combined Approach to Improving Data Quality

We now present the effects of using every method for leveraging weakly labeled data described in this section with experiments conducted in the simulation environment introduced in section 3.3.

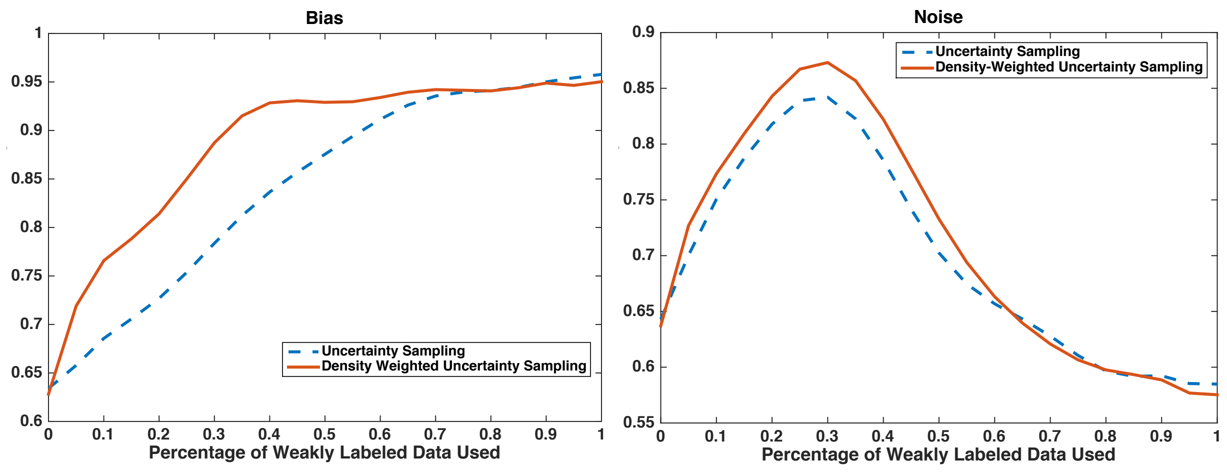

Figure 3.6: Combined Data Quality Improvements for Biased Dataset (Left) and Noisy Dataset (Right)

For each experiment, a training set, parameter validation set, and testing set were created, along with one of three types of weakly labeled data: 1) noisy data, 2) biased data, or 3) noisy & biased data. Table 3.1 shows the parameters used to generate each of these types of datasets. For each of the three types of weakly labeled data, 1,000 experiments were conducted, and results were averaged across all trials. Figures 3.6 and 3.7 present the results of these experiments.

Each experiment begins with a spectral HMM trained using the labeled dataset. Internal Error Correction is then performed on the weakly labeled dataset. The data is then sorted according to either the uncertainty sampling active learning metric or the density-weighted uncertainty sampling metric, and the top data points are used for training with percentiles ranging from 0 to 100. After training using the weakly labeled data, Internal Error Correction is applied once more, and the process repeats for a second iteration. The parameters b and W_r are selected using the parameter validation set, and the density weighting KDE is also constructed using the validation set.

There are several observations that can be made with regards to these empirical results. The parameter, b, which controls the preference for uncertainty sampling in density-weighted uncertainty sampling, decreases monotonically with added noise and bias. Density weighting dampens the effects of bias in a data source, so it is sensible that smaller values of b would perform best

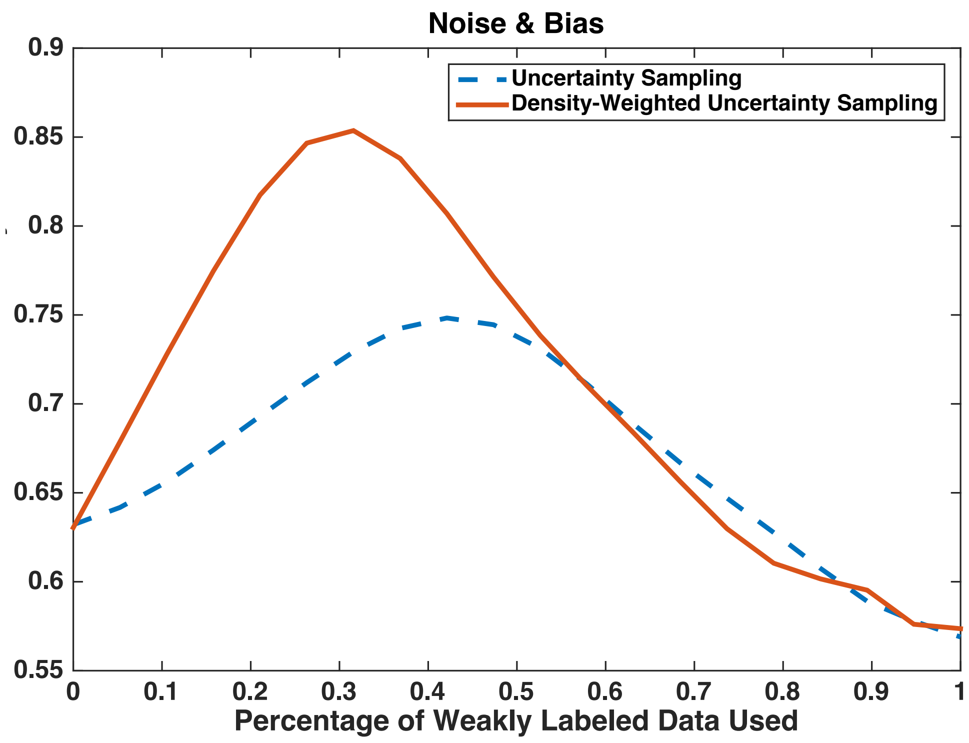

Figure 3.7: Combined Data Quality Improvements for Dataset with Noise and Bias

in those instances. With noisy data, uncertainty sampling may introduce unnecessary error, also causing smaller values for b to be selected. The parameter W_d similarly decreases monotonically with noise and bias, as the quality of the weakly labeled dataset worsens. We also observe that when noise is present in the dataset, using the entire weakly labeled dataset actually reduces performance. In figure 3.7, the data is being selected according to the active learning metric, meaning that as we go from using 80% of data to 100% of data, we are the 20% of weakest data according to our data selection metrics. This effect does not happen with data that is only biased, as the most biased data does not improve prediction, but also does not significantly hinder prediction either.

It is also worth noting that predictive accuracy is closely tied to the correctness of the learned latent space. In the experiments shown in figure 3.7, when the model's predictive accuracy is at its lowest (below 65%) the number of learned latent states matches the correct number of latent states in only 72% of experiments. On the other hand, in experiments in which the predictive accuracy exceeds 80%, the correct number of latents states is learned 91% of the time.

Figures 3.6 and 3.7 show direct comparisons between standard uncertainty sampling and density-weighted uncertainty sampling. Using density-weighting always improves performance on average, but those effects are magnified when dealing with significant bias in the training set. We can also observe that when a dataset is biased, but contains no noise, prediction accuracy increases monotonically with additional weakly labeled data, although active learning allows a model to learn with less data in these instances. However, when noise is introduced, we see that performance eventually dips below that of the fully supervised classifier. Figure 3.7 shows the most difficult training environment, when the weakly labeled dataset is both noisy and biased. Looking at these results, along with Figure 3.5, we see that some combination of weighting and active learning is required to outperform the fully supervised classifier. We also see that additionally incorporating Internal Error Correction allows us to achieve a 21 percentage point improvement over the fully supervised baseline, using only an additional 650 datapoints.

Throughout these simulated experiments, we observe that there is no single solution for weakly labeled data. Weakly labeled datasets in the real world often contain a mixture of noise and bias, requiring us to employ a mixture of different techniques to better leverage these datasets. In the subsequent chapters, we will present a variety of challenging applications using weakly labeled data from across the noise-bias spectrum. As we will see, the fundamental tools presented in this chapter will allow us to significantly outperform fully supervised models using little to no expensive labeled training data.

3.6 Overview of Applications

In subsequent chapters we will highlight four applications in which we are able to extensively leverage weakly labeled data. In this section we will briefly introduce each application, and describe how its data sources fit into the noise-bias spectrum. To give some insight for the reader into each problem, we used the labeled testing data for each application to empirically determine

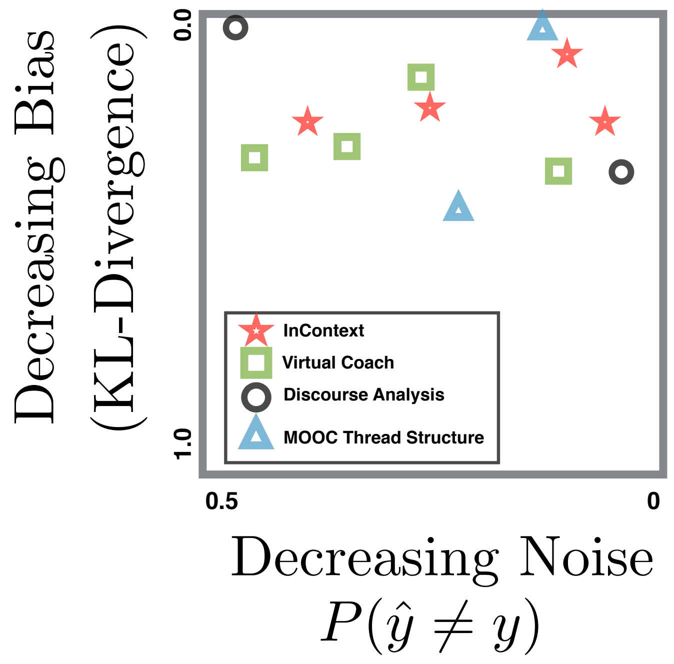

Figure 3.8: Empirical Noise and Bias Across Four Applications

the KL-divergence and noise rate of each source of weakly labeled data, though this information was not used at any point during model training. Figure 3.8 gives a high-level overview of where each application fits in the spectrum.

The first application is the *InContext* system for mobile devices [24]. This system monitors contextual features from a mobile device, including GPS location, time of day, and day of week, as well as information taken from accelerometer and audio signals. This information is then combined with an individual's phone usage history to develop a predictive model to determine whether the individual would prefer to have their phone ringer sounded or silenced in the current context. For evaluation purposes, a prototype application was deployed amongst several volunteers that periodically provided ground truth responses for their preference in various contexts. The primary source of weakly labeled data in this application is the hardware ringer switch present on the phone. Unlike many types of weakly labeled data, this data source actually represents a *curve* through the noise-bias spectrum rather than a single point. This curve is a function of t, the time since the user changed the setting of the hardware ringer switch. As the hardware setting becomes more stale, the quality of labels inferred from the switch become weaker. The noise and bias of the hardware switch as a label proxy was computed using the user's ground truth labels. It is quite intuitive that the noise of these labels would increase with time, but stale hardware settings also tend to occur often while the user is sleeping, which is a context not represented in the labeled data, leading to increased bias as well. Figure 3.8 shows three average noise-bias values for varying values of t in minutes, specifically $t < 10$, $10 \leq t < 150$, and $150 \leq t$, represented by the leftmost three stars in figure 3.8. Empirically, we see a significant increase in noise, and a slight increase in bias as a function of t. A second source of weakly labeled data was the user's responses to incoming phone calls, represented by the lower right star in the figure. These labels were relatively noise free, but were biased by the distribution of incoming phone calls received by the user, which did not cover all contexts the user was likely to visit. The datasets involved in this application are significantly smaller than the other applications, reducing the need for spectral methods. Instead, a relatively simple nearest neighbors approach is combined with internal error correction to yield an average accuracy above 95%.

The second application is the *Virtual Coach* smart wheelchair platform [25]. Similar to the InContext system, the goal in this application is to determine a user's preferences given contextual information. In this case, we wish to determine if a user of a power wheelchair would be willing to perform a pressure relief exercise in a given context. Pressure relief exercises can take several minutes to complete, and require the user to remain stationary on even ground—making them potentially very disruptive for the user. However, these exercises must be completed quite regularly to prevent potentially fatal pressure ulcers. Users of the Virtual Coach periodically receive prompts from the system to perform pressure relief, and the system monitors if a user complies with the prompt or not. These responses are taken as ground truth, and thus do not exhibit noise or bias. Furthermore, the user is able to set their notification preferences at any time, which provides us with a weak signal very similar to the hardware ringer switch seen with the InContext application. Similarly, the quality of this signal degrades with time since the user set the software preferences, and the leftmost three squares in figure 3.8 represent this curve. In general users of the Virtual Coach were less likely to set their preferences than users of InContext, leading to a significant increase in noise, and a slight increase in bias. Users were also periodically prompted to indicate to the system if they would be willing to perform a pressure relief exercise in their current context. These answers are shown as the rightmost box in figure 3.8. This labels demonstrate some noise, and the user's responses did not alway correspond to their actual reactions to pressure relief prompts, but also shows significant bias because many of the periodic prompts were ignored by the users—leaving only a subset of contexts represented by the user responses. The dataset for this application is significantly larger and noisier than InContext, allowing us to leverage a spectral, latent-variable model. Combining this model with internal error correction allows us to achieve a predictive accuracy of 94%.

The third application consists of labeling implicit discourse relations as described in section 2.1.3. In this application, the Penn Discourse Treebank was used to provide a labeled set of testing and training data from across the Wall Street Journal [54]. In addition, implicit relations are identified from unlabeled data across the web. Implicit relations can be identified quite accurately using very simple methods. These relations are represented by the rightmost circle in figure 3.8. The implicit relations are almost completely noise free, but suffer from significant bias, as not all relations have implicit forms, and those that do may have significant lexical differences between their implicit and explicit forms. As such, we see that this data source is the least noisy, but the second most biased of all data sources across all applications. In addition, bootstrapped labels taken from unlabeled text were added to the training set, represented by the left-most circle. These labels were taken from Wall Street Journal articles, and showed very little bias, however, this was the noisiest data source of all applications, due to the intrinsic difficulty of discourse relation classification. The enormous size of this dataset all but requires a spectral approach. Additionally, the large, biased implicit relations dataset lends itself very well to active sampling, resulting in an average predictive accuracy of 49%.

The final application discussed is determining the response structure of threads seen in forums taken from Massively Open Online Courseware (MOOCs). The goal here is to determine which pairs of posts in a thread share a response relation, wherein the second post in the pair represents a direct response to the first. This application uses two sources of weakly labeled data. The first is based on structured comments, wherein a user can post a response directly to another post in a thread. This source of data is represented by the leftmost triangle in figure 3.8, which shows that this is amongst the weakest sources of data across all applications. The comment pairs were highly biased because they only represent positive examples of response relations, and the lexical structure of comments is often quite different from unstructured posts. The comment structures were also highly noisy, as users would often use the comment system to make general observations unrelated to the post upon which they were commenting, making this data source one of the most ideal use cases for internal error correction. The second data source for this application is based upon the observation that the second post in a thread is nearly always a direct response to the first. This data source is represented by the rightmost triangle, which we see suffers from very little bias and noise. By combing our tools for improving weakly labeled data with the spectral, latent variable method we achieve a maximum predictive accuracy of 92%.

Taken together, these applications incorporate weakly labeled from all across the noise-bias spectrum. As we will see, in several instances we can use tools to offset the negative impact of weakly labeled data, allowing us to significantly outperform fully supervised classifiers that require costly hand-labeled data.

Chapter 4

InContext Smartphone Application

4.1 Introduction

With the proliferation of mobile devices, interruptibility has become a defining problem. Users often forget to change the settings on their mobile devices throughout the day, which results in inappropriate interruptions or important notifications being missed [49]. However, modern mobile devices are being outfitted with broad sensing suites and relatively powerful computational capabilities, giving those devices the ability to monitor and adapt to changing social contexts. In this chapter, we describe the *InContext* smartphone application, which uses a combination of weakly labeled data, signal processing, active learning, and supervised machine learning to create a personalized policy for changing a user's ringtone autonomously. This application leverages a smartphone's GPS, accelerometer, microphone, proximity sensor, and computing power to identify similar contexts and act according to the user's observed historical preferences. The techniques being used in this application could be applied in any setting in which we wish to personalize an instrumented system—as we shall see with the Virtual Coach smart wheelchair system in the subsequent chapter.

There are several practical and theoretical challenges involved in building such a system. On a practical level, the system should be able to operate using only those sensors found in a standard smartphone, without requiring the user to wear additional instrumentation. Furthermore, the power consumption of the system must be such that the user can continue to use his or her phone throughout the day, which limits the computational complexity of the methods we may use. On a theoretical level, a variety of latent variables, which the onboard sensors cannot observe, may factor into users' preferences in different contexts. Also, because the system is being designed to reduce the intrusiveness of the device, unnecessary or inappropriate queries of the user should be avoided.

An overview of the InContext system is shown in Figure 4.1. We use information extraction algorithms on the phone's sensor data to build a representation of the user's current context. In particular, we use a voice activity detection algorithm on audio data and a phone posture recognition algorithm on accelerometer and proximity sensor data, as well as features that do not require pre-processing such as GPS data and timestamps. Given a representation of the current context, we passively monitor the user's changes to their hardware ringer setting, as well as their responses to incoming calls. These signals represent this application's weakly labeled data sources, because users may have forgotten to change their ringer setting, or they may be basing their decision to accept a call on latent factors. For users who consistently change their ringer setting according to their preferences, the model trained on the weakly labeled data is sufficient. However, we also allow the system to use an active learning framework to select contexts in which to query the user about their true preferences in cases where the system has significant uncertainty about the correct setting in the current context. Unlike other applications in which we use active learning to select the most useful subset of a weakly labeled dataset, in this case we are using these techniques to minimize the invasiveness of the system.

Most previous attempts to determine user interruptibility, in mobile as well as desktop applications, have relied on active user input to determine their preferences [10, 31, 67]. Some of this work has explicitly considered the user's current interruptibility when deciding whether to issue prompts for input[57], but all of these systems perform poorly with users who are often unwilling or unable to respond to queries.

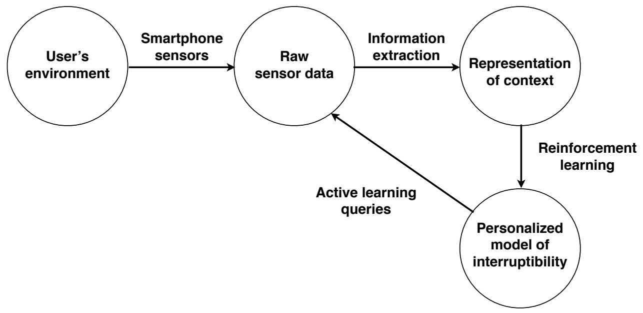

Figure 4.1: The InContext framework

Our system expands on previous work in two central ways. First, we incorporate weakly labeled data by allowing the system to learn about users by passively observing their day-to-day behavior with their phone, such as when they change the ringer setting or respond to incoming calls. This allows us to learn an effective model using either a small number of questions or no questions at all. Secondly, we leverage a new metric for actively learning, one not previously used in the application of interruptibility. While most existing systems have issued questions to the user based on uncertainty sampling [41], we propose the use of density-weighted uncertainty sampling [62], as described in section 3.5.2, which considers how representative the current context is of other contexts in the user's data-set, in addition to the system's uncertainty about the current user preference. Unlike the other applications described in this thesis, the datasets for this application were relatively small, reducing the need for spectral methods, allowing us to achieve positive results using simpler classifiers. We discovered that this approach allows us to attain an aggregate classifications accuracy of 96%, while requiring fewer queries of the user than previous approaches.

4.2 Dataset

Data was collected over a seven-day period from five volunteers using iPhone brand smartphones. The data collected included readings from the phone's 3-axis accelerometer, GPS unit, microphone, proximity sensor, timestamp records as well as user activity and responses to incoming

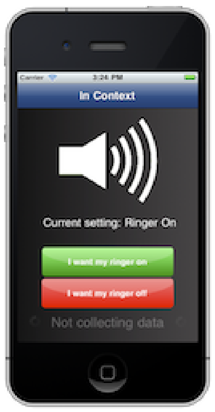

Figure 4.2: User interface for data collection

phone calls. The state of the user's hardware ringer switch (on or off) was also collected with every sample. Data was read from the sensors for only a ten-second period once every ten minutes, in order to preserve the phone's battery life. Under these conditions, we estimated that our system is able to run for 23 hours continuously on an iPhone 4 handset, or 19 hours continuously on an iPhone 3GS. For purposes of system evaluation, each user was also queried approximately once every two hours to provide their true preference for the ringer setting in the current context. In addition, each user was also permitted to provide their current preference to the system at any time, which would postpone the next prompt for user input by two hours. The graphical user interface is shown in Figure 4.2.

After the raw data was collected, it was passed through information extraction algorithms on board the smartphone, and the output of those algorithms was stored. The details of the information extraction are provided in section 4.4. In particular, we represented a user's context using seven core pieces of information, described in Table 4.1. We have done our best to minimize the invasiveness of the system on the user's privacy by encrypting or deleting data as much as possible. Although some private information is collected, previous work suggests that most users are willing to divulge some private information in return for services with high utility [67].

There are other modes of data which could be collected on a smartphone but were not used in this study. For instance, only one of our users reported keeping their smartphone calendar up-to-date, so calendar events were not collected in our dataset. Additionally, we did not record

Context feature	**Details**
Phone posture	A number in the set $\{0, 1, 2\}$ indicating if the phone is 0: Resting on table 1: In user's hand 2: In pocket or bag
Voice Activity	A bit indicated the presence of human speech
Sound level	A number in the set $\{0, 1, 2\}$ indicating if the sound level is quiet, average, or high.
Hour	An integer in 0-23 indicating the hour of the day
Weekday	An integer in 1-7 indicating the day of the week
Location	The latitude and longitude of the current location. These numbers are hashed using a secret key before being recorded to preserve privacy.
Ringer switch	The current setting of the hardware ringer switch.

Table 4.1: The representation of a user's context

the identity of incoming callers at the request of several of our study participants.

4.3 Weakly Labeled Data

Figure 4.3 indicates where the weakly labeled data for this application fall in the noise-bias spectrum. The two sources of weakly labeled data are taken from the phone's hardware ringer switch, and the user's responses to incoming calls.

As discussed in section 3.6, the hardware ringer switch does not represent a single point in the noise-bias spectrum, but rather a curve. This signal becomes weaker as the user goes longer without changing their hardware ringer switch. Figure 4.3 shows this curve represented at three different intervals of time, t, in minutes. This source of weakly labeled data shows very little bias, what little bias is present is likely accounted for by data collected at night when user preferences are not collected. However, the noise of this signal increases significantly with t. It is also worth noting that the weakness of the hardware switch labels varied significantly between users. In one user, the switch nearly represented ground truth, while another user hardly ever changed the ringer setting, despite indicating varying preferences though the labeled data. For users with

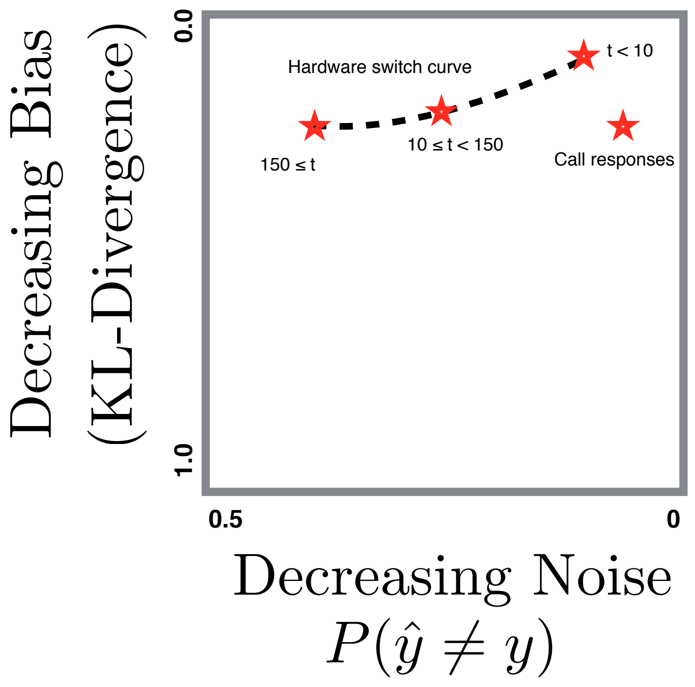

Figure 4.3: InContext Data in the Noise-Bias Spectrum

particularly noisy ringer settings, using some labeled data to train a model allows us to leverage internal error correction as described in section 3.5.1, which significantly reduces the confusion caused by the noisy data. For some users we see that even when $t = 0$, the hardware switch may be not reflect the user's true preferences. This may be due to the switch being accidentally flipped while in a pocket or purse, but we also observed several instances of the switch being rapidly toggled on and off, likely due to idle handling of the phone by the user.

The second source of data is collected from user's responses to incoming phone calls. When a call comes in, we collect all relevant contextual data and monitor whether the user accepts the call or not. This signal shows very little noise, though there were some instances in which the user's response did not coincide with their stated preference. An explanation for these discrepancies may be the identity of the caller, which the system was unaware of. This data source does also demonstrate some bias, as many users received phone calls only during relatively narrow windows of time through the day.

Because these data sources show relatively little bias, we use active learning to minimize the invasiveness of the system when collecting labeled data, rather than filtering irrelevant biased data. For the user that showed significant noise in the hardware switch signal, internal error correction is an ideal tool for offsetting this noise.

4.4 System Overview

This section describes the primary components of the InContext system. The first subsections present the information extraction algorithms for phone posture recognition and voice activity detection, as well as the smoothing routine applied to the output of both. Next, we describe the techniques we evaluated for predicting a user's preferences using only the weakly labeled data. Finally, we describe our use of density-weighted uncertainty sampling to select the contexts in which we wish to issue active queries for the user's preferences.

4.4.1 Phone Posture Recognition

Previous work has shown that having knowledge of the user's physical activities can be used to help determine interruptability [29]. However, accurately classifying a user's activity generally requires one or more accelerometers placed at known locations around a user's body. With a mobile phone, a user may carry the phone in their pocket, purse, or on their belt, so we do not have a known reference point from which to conduct activity recognition. Instead, we simplify the problem to trying to estimate the current physical posture of the device itself. In this task, we wish to determine if the phone is resting on a flat surface, if it is being actively held by the user, or if it is placed in a pocket, purse, backpack, etc. To address this problem, we collected labeled data from these three classes, using the 3-axis accelerometer and the proximity sensor of the phone. The data was divided into overlapping half-second frames, with the sample mean and variance of the accelerometer axes recorded for each frame. The number of times that the proximity sensor was triggered over the half-second was also recorded. A linear support vector machine was then trained to differentiate these classes, attaining 91.4% accuracy over 89 test samples.

4.4.2 Voice Activity Detection

Audio data was collected from the smartphone devices at a sample rate of 8192Hz. Ten seconds of audio was recorded, and this signal was broken into 20 half-second samples. For each of these samples, a Fast Fourier Transform is used to extract 16 features, presented in Table 4.2. We empirically compared multiple classifiers for use in the voice activity detection task. A support vector machine with a linear kernel and a Gaussian mixture model were both trained on labeled audio samples to differentiate audio samples containing human speech from samples that do not contain speech. Although previous work has shown this approach to be effective at the voice activity detection task [37, 55], there is one complication that arises in a mobile devices application: the device may be in a user's pocket or handbag when the sample is collected, resulting in a significantly dampened signal and many false-negative predictions by the classifier. Because we are able to detect when the phone is in a pocket using the accelerometers and proximity sensor, we train a second speech detection classifier for this scenario. A linear support vector machine and Gaussian mixture model were also trained in this instance, with a new set of trained weights to account for the dampened signal.

The performance of the classifiers with the phone in and out of a pocket is shown in Table 4.3. The testing set included many noisy audio samples without voice activity, such as music and sounds of car traffic. Based on these results, the linear support vector machine was selected for deployment in the InContext application.

#	Feature	Description
1	Fourier mean	The sample mean of the magnitudes of all Fourier coefficients in the sample.
2	Fourier variance	The sample variance of the magnitudes of all the Fourier coefficients.
3	Total signal power	The sum of the squared magnitudes of all the Fourier coefficients
4	Mid-range power	The sum of the squared magnitudes of the Fourier coefficients in the 250-600Hz range of the spectrum.
5	Ratio	The ratio of the mid-range power over the total signal power.
6	Zero crossings	The number of zero crossings in the Linear PCM encoding of the audio signal
7-16	Band power	9 features representing the signal power in 100Hz bands from 1 to 1000Hz. The bands are 1-100Hz, 101-200Hz...901-1000Hz.

Table 4.2: Voice activity detection features

	GMM	**SVM**
In pocket	86.7%	91.3%
Out of pocket	90.5%	95.1%

Table 4.3: Voice activity detection accuracy

4.4.3 Smoothing of Information Extraction Output

Because several seconds of data is collected whenever we wish to sense the current context, both of the information extraction algorithms actually produce a sequence of predictions as their output. Rather than simply using the mode label of the sequence, we instead consider the certainty of the classifiers over the time interval. For phone posture recognition, we select the label with the most total certainty over the sequence. However, for the voice activity detection, we recognize that pauses in a conversation may result in many elements of the label sequence receiving a negative label, even though there is actually a conversation present in the environment. Therefore when smoothing the predictions of the voice activity detection algorithm, rather than selecting the most confident label, we instead require that the total confidence surpass a certain threshold. This is formalized in equation 4.1, in which the smoothed label Y_{VAD} is based on a sequence of data-points $\mathbf{x_1} ... \mathbf{x_T}$. For each element in the sequence, the distance to the decision boundary of the SVM is given by $\mathbf{w} \cdot \mathbf{x_t}$. The threshold, C, was selected to maximize classification accuracy on a labeled parameter validation set.

$$Y_{VAD} = I\left(\left[\sum_{t=1}^{T} \mathbf{w} \cdot \mathbf{x_t}\right] \geq C\right) \tag{4.1}$$

4.4.4 Preference Classification

This section addresses the problem of trying to predict a user's preferences in a given context, given their preferences in previous contexts. Unlike the other applications described in this thesis, we will be dealing with small datasets that do not require spectral methods to minimize computational complexity. Instead, we employ a variant of the nearest-neighbors algorithm for the task of selecting ringer preferences in different contexts. We compared this algorithm to a variety of other simple baseline classification algorithms, including a support vector machine, decision tree, and naive Bayes algorithms, and found that the nearest-neighbor algorithms outperformed these alternatives.

For the purposes of classification, we use the first six variables presented in table 4.1 as features, and we use the weakly labeled data sources to provide training labels. As noted previously, we believe that at the moment a user changes their ringer setting, this setting is most likely correct for their current context (or near future contexts). Therefore, the weakness of this data source will increase with time. Unlike other weakly labeled data sources in which we may use a static weight value in training, in this instance we wish to decrease the weight of the hardware switch data source as a function of t. To capture this behavior, equation 4.2 shows the exponential decay function we use to weight the samples. In this equation, the weight W_i of datapoint i is based on the hardware ringer switch, Y_i, of this sample, which adopts value 1 if the switch is on and -1 if the switch is off. The exponential decay parameter λ is selected using the validation set. It was empirically determined that small changes to these parameters do not have a significant impact on classifier performance, so it is not necessary to learn them for each user. The variable h denotes the number of hours since this setting was selected, rounded to the nearest whole number. The weight function has an additional benefit as well. When this system is deployed on a user's phone, if the preference classifier is working correctly, we envision the users no longer needing to change their ringer setting. By ignoring the ringer setting if it has been a long time since the user set it, we allow the system to take over completely when the user is satisfied with the system.

$$W_i = \begin{cases} Y_i e^{-h/\lambda} & : h \leq 12 \\ 0 & : h > 12 \end{cases} \tag{4.2}$$

The distance function for the nearest neighbor classifier is given in equation 4.3. This function describes the distance between two recorded context C^i and C^j. For each feature d in contexts i and j, we consider the difference $|f_d^i - f_d^j|$. For the phone posture, voice activity, weekday, and sound level features, this is simply the Hamming distance. For the hour feature, the difference is $min(|h_i - h_j|, 4)$. For the location feature, the difference is the indicator function, denoting whether these two locations are within 150 meters of one another. For each feature

k, we have a distance parameter d_k, which was selected on a validation set taken from a single user's data.

$$D(C^i, C^j) = \sum_{k=1}^{6} d_k |f_d^i - f_d^j| \tag{4.3}$$

Using the distance function given by equation 4.3, we have the decision policy given in equation 4.4. If we wish to predict the ringer setting for a context C, we take a weighted summation over the k contexts in the user's history with the smallest distance to the current context. If this summation is non-negative, the algorithm predicts that the user would like the ringer turned on. Otherwise the ringer is turned off. It is worth noting that this prediction function gives us an obvious confidence measure, namely the weighted summation of distances to the nearest contexts. The larger the magnitude of this summation, the more confident the algorithm is of the prediction. Empirical results for this algorithm are given in section 4.5.

$$Pred(C) = I\left(\left[\sum_{i=1}^{k} \frac{W_i}{D(C, C^i)^2}\right] \geq 0\right) \tag{4.4}$$

4.4.5 Active Learning

For this application, we will leverage active learning as described in section 3.5.2. However, rather than using it to filter weakly labeled data, we will use it as a traditional tool to query a labeling oracle. In the context of interruptibility, the user acts as the oracle, and these queries are presented *in situ* so as to benefit from the increased accuracy of experience sampling [17]. In scenarios in which a labeling oracle is available, active learning has been shown to greatly increase classification accuracy[61].

As previously described, we will use entropy $H(X)$ as our measure of uncertainty.

$$H(X) = -\sum_{l=0}^{1} P(Y(X) = l|X) \log[P(Y(X) = l|X)] \tag{4.5}$$

In our mobile phone application we are likely to see many samples densely packed around a small number of contexts (e.g. the user is at work or at home), plus a small number of previously unknown contexts (such as when the user tries a new restaurant). This makes the density sampling component described in section 3.5.2 particularly important. However, rather than building a kernel density estimate for the feature space, we will use a more computationally efficient function that can be easily computed on a phone. The metric function for density-weighted uncertainty (DW) sampling is given in equation 4.6. In this equation, $sim(X, Y)$, is a function representing how similar two points X and Y are. For our similarity function, we use the squared reciprocal of the distance function from equation 4.3.

$$DW(X) = H(X) \sum_{i=1}^{n} sim(X, X^i) \tag{4.6}$$

With density-weighted uncertainty sampling, we wish to query the label of the sample, X, that maximizes $DW(X)$. This will be a sample that the algorithm is uncertain about labeling, but which is also representative of several other data-points in our dataset. We also use an additional heuristic, in which the algorithm will not request the label of a point if a similar data-point has already been labeled.

We collected 50-100 labeled data-points for each volunteer using the InContext system. These data-points were collected using experience sampling, according to a uniform query schedule. Of these labeled data-points, 50% were set aside for testing for each user. The remaining labeled data-points were used to evaluate the benefit of allowing the system to actively request labels. In the next section, we compare density-weighted uncertainty sampling to standard uncertainty sampling, which is the technique used in most previous interruptibility prediction systems[29, 35].

	Features	**RBF SVM accuracy**
With Information Extraction	6	95.3%
Without Information Extraction	33	59.3%

Table 4.4: Effects of voice activity detection and phone posture recognition (Information extraction)

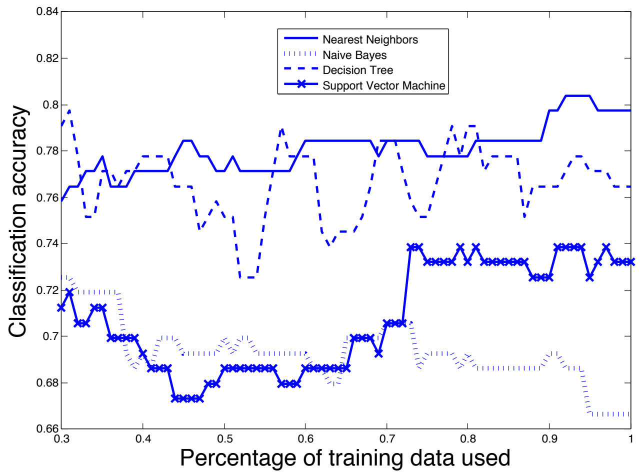

Figure 4.4: Comparison of preference classifiers

4.5 Results

Figure 4.4 shows a comparison of four different classifiers for predicting ringer preferences. We evaluated the nearest-neighbors algorithm described in section 4.4.4, a support vector machine with an RBF kernel, Naive Bayes, and a decision tree using the information-gain metric. Additionally, the support vector machine was evaluated on a single user when given the raw features used for information extraction, rather than the output of the information extraction algorithms (voice activity detection and phone posture recognition). From the summary of experimental results, shown in table 4.4, we see that that these two information extraction algorithms have a significant positive impact on classification accuracy.

The effects of active learning queries on classification accuracy are shown in figure 4.5. In this experiment, we compared the performance of standard uncertainty sampling against density-weighted uncertainty sampling. The classification accuracy is averaged across all five users. We see that density-weighted uncertainty sampling consistently outperforms uncertainty sampling. When given the maximum number of active labels (15), the classifier with density-weighted

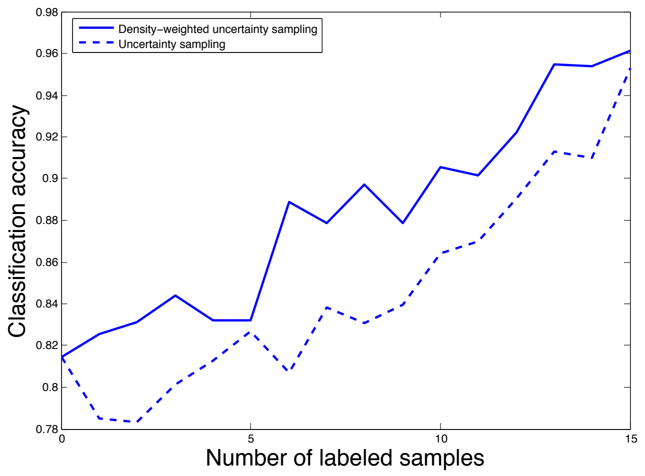

Figure 4.5: Active learning curve

uncertainty sampling attained an average classification accuracy of $96.12\% \pm 3.37\%$. This equates to approximately two queries per day. When the number of queries is dropped to one per day, the classification accuracy is $87.86\% \pm 5.68\%$. With no active queries at all, the classification accuracy is $81.46\% \pm 6.20\%$, indicating that the weakly labeled data in this application is quite useful on its own. For all but one user, using weak data alone leads to an average accuracy of 87.82%—indicating very low noise rates for those users. The remaining user very rarely set the hardware switch, so training this user using only weakly labeled data attains only 52.0% accuracy. By training using 7 active queries, and then conducting internal error correction as described in section 3.5.1, we are able to improve this number to 83.52% for this user, and 90.12% averaged across all users.

Hardware ringer switches have not previously been used to train interruptibility classifiers, presumably because the noise was thought to be too great. We see that, in general, this is not the case, with four of our five users attaining a classification accuracy above 80% with no active labels. One user attained 96.32% accuracy using no active labels. In addition, internal error correction allows us to attain positive results even in instances in which the noise rate would otherwise make training intractable. By starting with a much higher baseline, we need fewer queries to users to push classification accuracy above 95%. Compared to a system that relies only on user queries[57], we are able to produce comparable accuracy with many fewer queries, and much greater accuracy when the user is willing to answer only a small number of queries. We additionally see that the density-weighted uncertainty sampling provides increased accuracy compared to regular uncertainty sampling. Furthermore, we conjecture that the density term will prevent the system from issuing queries every time the user travels to a new context.

4.6 Conclusions

The InContext application is the simplest task described in this document, allowing us to use simple classification methods to great effect. However, it succinctly demonstrates the effectiveness of using weakly labeled data to minimize the need for hand-labeled training data. In this chapter, we have seen that we can achieve very accurate prediction using little to no supervision. The sources of weakly labeled data in this application showed fairly little bias and noise, with the exception of one user. For the anomalous user, internal error correction allowed us to achieve desirable results using only 7 labeled examples. We have also seen that active learning can make an excellent tool for carefully selecting labeled examples. In subsequent chapters, we will also see how active learning can combat biased weakly labeled data.

In the following chapter, we will turn to the Virtual Seating coach, a natural extension of the InContext application. With the Virtual Seating coach, we will be dealing with a larger, weaker training dataset, requiring us to leverage more advanced techniques, in particular a latent variable, spectral method.

Chapter 5

The Virtual Seating Coach System

5.1 Introduction

We now turn to an application that is conceptually very similar to InContext, but with with much higher stakes depending on the system's performance. More than 2.5 million Americans experience pressure ulcers every year [56]. One particularly high risk group of individuals are those with severe physical disabilities that utilize power wheelchairs as their primary means of mobility. Power wheelchairs can very effectively relieve pressure by raising the user to a reclined, titled position for several seconds. However, research indicates that less than 40% of power wheelchair users correctly use their power seat functions to relieve pressure and prevent deadly ulcers [40]. This leads many power wheelchair users to be exposed to preventable pressure ulcer formation, which can often lead to complicating infections or even death.

The Virtual Coach smart power wheelchair system was designed at the Human Engineering Research Laboratory [45]. It is built to track the power wheelchair usage of users in order to help them better conform to the pressure relief guidelines set forth for them by clinicians, as well as improve their overall posture. By monitoring encoders in the power wheelchair's joints (chair tilt, leg rest elevation, seat elevation, etc), the system is able to determine when a user has successfully performed the repositioning exercise prescribed by the user's healthcare provider.

Users are reminded on a periodic basis to conduct pressure relief, and the system tracks whether the user complied with the reminder or not. Initial results suggest that users that receive regular reminders from the system have higher rates of compliance with their exercise regime than those users not receiving instruction [45]. However, there is a risk of the system becoming intrusive and annoying if users are given too many undesired reminders.

Instead of simply reminding users with a static periodicity, we would like to expand upon the InContext framework to deliver context-aware, personalized exercise prompts. In this section, we present the results of applying our methods to predict whether a user is likely to comply with a reminder given information about the user's current context collected by several sensors onboard the Virtual Coach system. We show that when averaged across all users, we can attain a predictive accuracy of 92%. Compared to the InContext system, we are dealing with weaker, higher dimensional data—with regards to both noise and bias. In addition to leveraging active sampling and internal error correction, we also utilize the spectral, latent variable method described in section 2.1.2. The spectral HMM provides us with the more accurate predictive model, but some clinicians have also requested a more interpretable model to help them understand why patients do or do not comply with exercise prompts. To address this need, we also provide results with a computationally lightweight decision tree model that creates an output which is easy for clinicians and users to understand. The goal for this work is to create an intelligent, decision theoretic reminding system that would select the best moments to issue reminders to users given the likelihood of compliance in the current context, the time since the last pressure relief was completed, and the recommendations given by the overseeing clinician.

5.2 The Virtual Seating Coach System

The Virtual Seating Coach (VC) system is a smart power wheelchair outfitted with a variety of sensors and an onboard computer. The sensors installed on these chairs include encoders in each of the chair's joints and wheels, accelerometers in the base of the seat, a seat occupancy sensor,

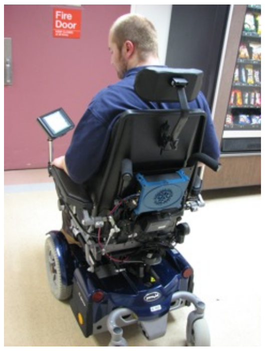

Figure 5.1: A researcher demonstrating the use of the Virtual Coach system

a thermometer, and a light sensor. Additionally, a subset of users in the clinical trails used chairs that were outfitted with GPS chips and microphones. A complete list of the sensors and the machine learning features that were computed is shown in Table 5.1.

A tablet computer is attached to the arm of the chair, as seen in Figure 5.1. The software on this tablet is used to issue reminders to users, and to provide feedback while an exercise is being conducted. Additionally, users can set preferences that dictates if reminders will be given using audio queues, silent text, or if the reminders should be disabled completely. When a reminder is given, a user is given the choice of snoozing the reminder, which will result in another prompt being given in 5 minutes, or the reminder can be dismissed for 1 hour. If the user does not comply with the reminder, it is automatically snoozed for 5 minutes. If the user successfully completes the exercise, the reminding system will be reset for a duration determined by a clinician. Additionally users were queried at longer random intervals to indicate if they would be willing to perform a pressure relief exercise in the current environment. The user questionnaire responses are used to supplement the training data, but the user responses to exercise prompts are treated as the golden standard for determining interruptibility.

Data Source	**Features**
Chair angle encoders	Mean and variance of encoder values over previous 10 seconds
Wheel Encoders	Mean and variance of rotational velocity over previous 10 seconds
Accelerometers	Average Fourier power in 500 Hertz bands over 0-4000 Hz

Thermometer	Average temperature over previous 30 seconds
Seat Occupancy	A 0/1 value indicating state of pressure sensor in seat
Clock	The current day of the week and hour of the day
Recent Behavior	Time since the last reminder
Recent Behavior	Outcome of previous reminder (Snooze, dismiss, compliance)
GPS Data[1]	The current encrypted longitude and latitude coordinates
Audio Data[1]	Average Fourier power in 100 Hertz bands over 0-1000 Hz

Table 5.1: The features computed using the Virtual Coach sensors

The Human Engineering Research Laboratory has built several prototype versions of the system that have been undergoing clinical trials, and more than 20 volunteers have participated in the trials using the VC system over a period of 6 weeks per user [45]. Volunteers were either assigned to a control group, being given a sensor equipped chair without a reminding system, or they were given the full VC system with tablet computer. The system stores clinician pressure relief guidelines for the individual user. For instance, the recommendation may be for the user to receive 30 seconds of pressure relief once every 60 minutes. During clinical trials the system issued reminders on a fixed basis according to the prescription, and the user's response to the reminder was recorded—indicating if the user snoozed the request, ignored it, or complied with the prompt by completing an exercise.

5.3 Weakly Labeled Data

Figure 5.2 shows the empirical noise bias values for the weakly labeled data sources available with the Virtual Coach. We see a very similar arrangement in the noise-bias spectrum when compared to the InContext application. In particular, we have two principal sources of weakly

[1]Data only available for subset of users

labeled data.

For the first data source, users are able to provide reminding preferences through the Virtual Coach software, allowing them to select various reminding modalities (textual, audio, vibration) or to disable reminders all together. For the purposes of determining interruptibility, we cast these preferences into a binary class indicating if the user has disabled reminders or not. Similar to InContext, this data source represents a curve through the noise bias spectrum, as a function of time since the reminder was set. The KL-Divergence with regards to this testing set is computed using the features and labels of these datapoints, yielding an empirical estimate for bias. We use the recorded responses of users to exercise prompts to determine the noise of the labels in the software preferences as a function of preference staleness. We see that this data source is weaker than the hardware switch in the InContext application, both in terms of noise and bias. With the InContext application, user preferences are controlled by a simple, pre-existing hardware switch on a phone the users are already familiar with. With the Virtual Coach, user preferences are recorded via software in a sub-menu of a system that users are unfamiliar with. This lead to higher average staleness of preferences and more noise even for fresh preference labels.

The second source of weakly labeled data are periodic software questionares issued to users asking if the current context would be appropriate for interruption. We see a much lower noise rate for this datasource (0.0921), but increased bias. In particular, we see that users are significantly more likely to respond to these queries in instances in which they would be willing to comply with an exercise prompt, leading to an increased KL-Divergence with respect to the testing data.

Both of these datasets show high noise-rates, which makes Internal Error Correction an effective approach. To achieve this, we train a spectral, latent variable model using the labeled data, and relabeled all training points below entropy threshold ($\psi = 0.15$).

The user questionnaire responses are fairly low in volume for this application, making the density-weighted threshold less useful. As such, only the lowest 10 percentile of these labels are removed from the training set. On the other hand the continuously observed data labeled using

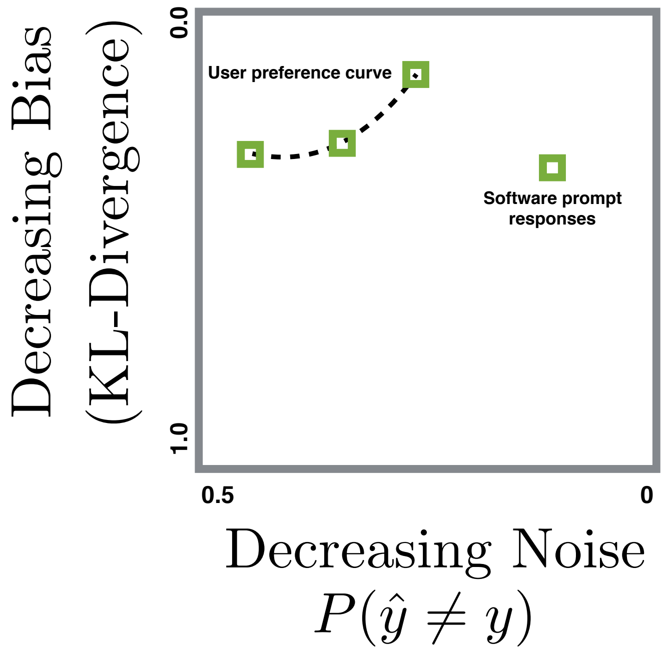

Figure 5.2: Weakly Labeled Virtual Coach Data in the Noise Bias Spectrum

the software preference setting has much greater volume, although most of these datapoints are redundant. As such, we only retain the top 30 percent of the original data according to density sampling, with the caveat that all datapoints in the training set have a maximum cosine similarity of 0.8. This step prevents too many redundant datapoints from being used in training.

Additionally, after Internal Error Correction and density thresholding are conducted, these datasources are re-weighted according to the procedure described in section 3.5.3. Using $W_r = 1$ for the exercise responses, we arrive at $W_r = .82$ for the software responses and $W_r = 0.525$ for the passively recorded preference data.

5.4 Spectral Hidden Markov Models

We see from figure 5.2 that we are dealing with weaker data in this application compared to InContext. In addition to leveraging our methods for improving weakly labeled data, we can improve predictive performance by explicitly modeling the sequential nature of the task at hand.

A person's daily routine is inherently sequential, as we move from one context to another, thus using a model with latent states allows us to compensate for the more difficult dataset induced by this application. The spectral, latent variable methods described in section 2.1.2 provide us with a powerful framework that models sequential dependencies using Markovian latent variables.

For this application, we have access to a relatively low volume of unique labeled datapoints. However, the environmental sensors were running almost continuously for all subjects, giving us a fairly complete representation of transitions between contexts. One of the most computationally expensive steps in building a spectral, latent variable model consists of computing empirical n-grams of the observation space, defined as follows:

$$[P_1]_i = Pr[x_1 = i]$$

$$[P_{2,1}]_{ij} = Pr[x_2 = i, x_1 = j]$$

$$[P_{3,x,1}]_{ij} = Pr[x_3 = i, x_2 = x, x_1 = j] \ \forall x$$

These n-grams can also be very sparse when the observation space is large, or training data is limited. The weakly labeled continuous data is of limited use for training, but is ideal for computing these empirical n-grams. By improving the accuracy of these n-gram matrices, we also derive improved estimates for the subspace spectral paramters $\hat{\pi}_U$ and $\hat{A}_U$.

The model is initially trained using a dataset that has undergone Internal Error Correction, density thresholding, and datasource re-weighting. After the model is trained using all these datasources, internal error correction is applied a second time, and the final model is trained using this refined dataset.

5.5 Decision Trees

Power wheelchairs are a principle component in the life of many people living with a physical disability, and research suggests that chair users prefer to remain in full control of their chair's functionality [5]. In light of this, we may wish to create a reminding system that is maximally transparent, so that a user or a clinician could better understand how the intelligent system is deciding when to issue prompts. The spectral HMM, while very powerful, relies on a latent variable distribution that often lacks a simple, relatable explanation.

In an effort to explore a model that would be easier for end users to comprehend, we turned to one of the oldest and most popular machine learning classifiers: decision trees. Decision trees conduct classification through a series of logical binary operations applied to the learning features. *Information gain* is most often used to determine which features to place near the root of the tree. Information gain is defined as the reduction in statistical entropy gained by learning the state of a random variable. If we denote $H(Y)$ to be the statistical entropy of a random variable, the information gain of X with regards to label Y is $H(Y) - H(Y|X = a)$.

Decision trees have been shown to be high bias classifiers, leading to a great deal of overfitting [70]. To combat this, decision trees are generally pruned near the leaves of the tree to reduce bias and improve the tree's capacity for generalization. Smaller trees will also be easier for users to view and comprehend.

An example of a heavily pruned decision tree trained on one user's Virtual Coach data is shown in Figure 5.3. In this figure, a lead node of 1 indicates that the user is expected to comply with a reminder, 0 indicates predicted non-compliance. The tree in this figure was trimmed to depth 3, leaving only encrypted location and audio features remaining. It is worth noting that the audio feature with the highest information gain is the Fourier signal power in the 300 to 400Hz range of the spectrum. This represents the lower end of the audio spectrum that the human voice inhabits, so the decision tree seems to indicate that this given user would prefer not to be reminded if there is evidence of a speech signal present in the environment. If a Virtual Coach

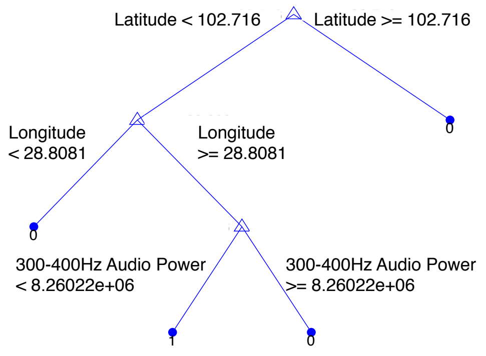

Figure 5.3: A Decision Tree Showing the Compliance Predictions of a User

user were to elect to use a decision tree to determine when the intelligent system should issue reminders, the user would be able to verify, and possibly modify, the behavior of the system. This would help to curb any confusion a user may have as to why the system is behaving in the way that it is. As before, the initial training data has already undergone Internal Error Correction, density thresholding, and re-weighting.

5.6 Results

When collecting empirical results, we used 15 fold cross validation to determine prediction accuracy of our models by randomly selecting one week of data to hold out for validation. Figure 5.4 shows the average prediction accuracy of the spectral HMM and decision tree models as a function of training data volume, including the usage of Internal Error Correction, density thresholding, and re-weighting. We also considered several other classifiers, including Support Vector Machines, Nearest Neighbors, Naive Bayes' algorithm, and Conditional Random Fields. None of these other approaches performed as well as the spectral HMM, or produced output that was as interpretable as the decision trees.

We see from these results that the spectral HMM performs particularly well with small amounts of data, with a very steep peak in the training curve after being supplied with only

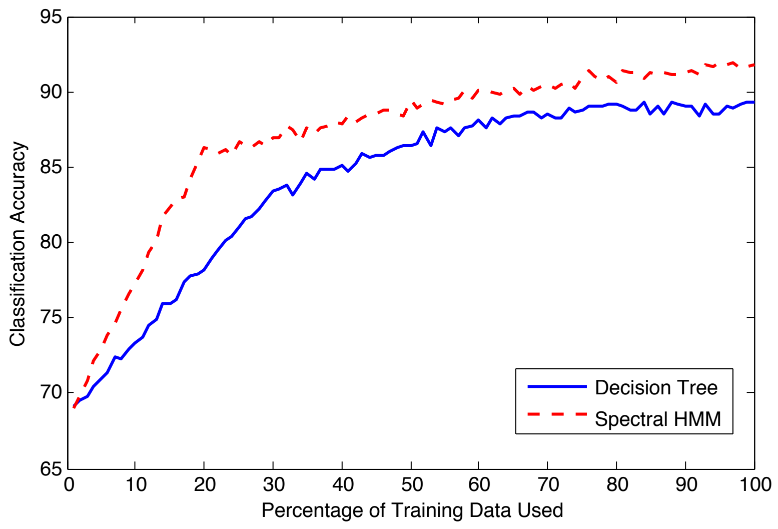

Figure 5.4: Empirical Training Curve

20% of the data (roughly one week's worth of data). This result fits with intuition, because we expect accuracy to plateau after seeing one full instance of a user's weekly schedule. The spectral HMM peaks at around 92% classification accuracy, while the decision tree model peaks at 88%. It is not surprising that the spectral HMM performs the best, because the HMM reflects recent observations through the latent state distribution, while the decision tree performs classification using only the current sensor values. For comparison, a naive model that always predicts the most likely prior attains 69% classification accuracy.

Table 5.2 shows the effects of Internal Error Correction (IEC), density thresholding, and data-source re-weighting. Similar to simulated results in this region of the noise-bias spectrum, we see a fairly significant improvement from Internal Error Correction. Density thresholding, however, provides a much more modest improvement. Due to the low volume of unique labels, we were unable to aggressively prune biased data away, which explains the incremental improvements this approach produces.

Re-weighting produces the most dramatic effects. This is likely due to the fact that the largest source of labels (continuously collected data labeled with software preferences) is also the largest—particularly when density thresholding is not deployed. This leads to an over-emphasis

IEC	Density Threshold	Re-weighting	HMM	DT
Y	Y	Y	.92	.88
N	Y	Y	.87	.82
N	N	Y	.85	.81
N	N	N	.76	.71

Table 5.2: Effects of Data Improvement Methods for the Virtual Coach

on noisy, biased data, which—in the worst case—leads to a model that barely outperforms the naive baseline.

We determined that both models performed best when supplied with training and testing data from only a single user. Models trained with data from multiple users resulted in many contradictory training examples, which is intuitive, because we would expect different users to react differently to a reminder under the same circumstances.

Prediction accuracy with users for whom audio and GPS data were available was higher than the rest of the population, showing 94% average accuracy with the HMM, and 92% average accuracy with the decision tree. We see a smaller differential between the two models in this case due to the fact that the GPS data shows very high information gain, often placing it near the root of the decision trees. For these users we can attain a predictive accuracy surpassing 85% using trees of only depth four that were built only using time of day, GPS coordinates, and the outcome of the most recent reminder. This suggests that simpler models are almost as effective for users that are willing to share personal information, such as location data, with the Virtual Coaching system.

5.7 Conclusions

Initial results show that we can accurately predict repositioning exercise compliance given a set of fairly common sensors embedded in a power wheelchair. The spectral Hidden Markov Model leverages Markovian information about the past in a framework that has many desirable theoretical and practical qualities. On the other hand, the decision tree model is computationally very lightweight, and presents an easily understandable graphical representation of the model. It could also be possible to combine these approaches, using a Hidden Markov Decision Tree [33] to build a model with maximum predictive performance that is still interpretable. Regardless of the model used, we have seen that by leveraging weakly labeled datasets, and a variety of tools for improving data quality, we are able to achieve high predictive accuracy in a task that includes very few labeled training examples.

By using our methods to monitor and alter users' pressure relief habits, we have the potential to prevent thousands of occurrences of pressure ulcers every year. The cost saving nature of this technology will also make it desirable for insurance companies and healthcare organizations, which could lead to faster adoption of the technology and improved outcomes for the millions of power wheelchair users in the world.

In subsequent chapters we will turn away from sensor based, context-aware applications, and towards large scale natural language tasks. We will encounter many of the same types of weakly labeled datasets, however, as the size of these datasets increases, the value of active density sampling will increase significantly.

Chapter 6

Discourse Analysis

6.1 Introduction

We now consider our first application in the natural language domain, discourse parsing. Compared to applications in the previous chapters, the size of available datasets derived from unlabeled text on the web is nearly infinite, which makes density thresholding more valuable and requires us to leverage fast, efficient spectral methods to train with the most data possible. Discourse parsing is a fundamental task in natural language processing that entails the discovery of the latent relational structure in a multi-sentence piece of text. Unlike semantic and syntactic parsing, which are used for single sentence parsing, discourse parsing is used to discover inter-sentential relations in longer pieces of text. Without discourse, parsing methods can only be used to understand documents as sequences of unrelated sentences.

Unfortunately, manual annotation of discourse structure in text is costly and time consuming. Multiple expert annotators are required for each relation to estimate inter-annotator agreement. The Penn Discourse Treebank (PDTB) [54] is one of the largest annotated discourse parsing datasets, with 16,224 implicit relations. However, this pales in comparison to unlabeled datasets that can include millions of sentences of text. By augmenting a labeled dataset with weakly labeled data, we can offset the limitations of a small hand-labeled dataset.

As a consequence of collecting weakly labeled data from the web, we are left with enormous datasets consisting of tens of millions of relation pairs, with hundreds of features within each pair. As such, the primary constraint on training set size depends on computation rather than availability of training data. As we have previously seen in Figure 2.1.2, spectral methods can sometimes be orders of magnitude faster than similar approaches such as EM. This increase in efficiency directly affects the volume of data that can be used for training in a given time, leading to a significant increase in predictive accuracy, in addition to the benefits of statistical consistency entailed in spectral methods. As such, discourse parsing is the first application described in this document in which we can begin to realize the full potential of spectral methods, while other competing methods become intractable.

This chapter presents a spectral model for a sequential relation labeling task for discourse parsing using weakly labeled data. Besides the theoretically desirable properties mentioned above, we also demonstrate the practical advantages of the model with an empirical evaluation on the Penn Discourse Treebank (PDTB) [54] dataset, which yields an F_1 score of 0.485. This accuracy shows a 7-9 percentage point improvement over previous approaches that do not utilize weakly labeled training data with spectral methods.

6.2 Problem Definition and Dataset

This section defines the discourse parsing problem and discusses the characteristics of the PDTB. The PDTB consists of annotated articles from the Wall Street Journal and is used in our empirical evaluations. This is combined with the New York Times Annotated Corpus [59], which includes 1.8 million New York Times articles printed between 1987 and 2007.

Discourse parsing can be reduced to three separate tasks. First, the text must be decomposed into *elementary discourse units* (EDUs), which may or may not coincide with sentence boundaries. The EDUs are often independent clauses that may be connected with conjunctions. After the text has been partitioned into EDUs, the discourse structure must be identified. This requires us to identify all pairs of EDUs that will be connected with *some* discourse relation. These relational links induce the skeletal structure of the discourse parse tree. Finally, each connection identified in the previous step must be labeled using a known set of relations [47]. Examples of these discourse relations include concession, causal, and instantiation relations. In the PDTB, only adjacent discourse units are connected with a discourse relation, so with this dataset we are considering parse sequences rather than parse trees.

In this work, we focus on the relation labeling task, as fairly simple methods perform quite well at the other two tasks [69]. We use the ground truth parse structures provided by the PDTB dataset, so as to isolate the error introduced by relation labeling in our results, but in practice a greedy structure learning algorithm can be used if the parse structures are not known *a priori*.

Some of the relations in the dataset are induced by specific connective words in the text. For example, a contrast relation may be explicitly revealed by the conjunction *but*. Simple classifiers using only the text of the discourse connective with POS tags can find explicit relations with high accuracy [44]. The following sentence shows an example of a more difficult implicit relation. In this sentence, two EDUs are connected with an explanatory relation, shown in bold, although the connective word does not occur in the text.

> "But a few funds have taken other defensive steps. Some have raised their cash positions to record levels. **[BECAUSE]** High cash positions help buffer a fund when the market falls."

We focus on the more difficult implicit relations that are not induced by coordinating connectives in the text. The implicit relations have been shown to require more sophisticated feature sets including syntactic and linguistic information [42]. The PDTB dataset includes 16,224 examples of implicit relations.

A full list of the PDTB relations is shown in figure 6.1. The relations are organized hierarchically into top level, types, and sub-types. Our experiments focus on learning only up to level 2, as the level 3 (sub-type) relations are too specific and show only 80% inter-annotator agree-

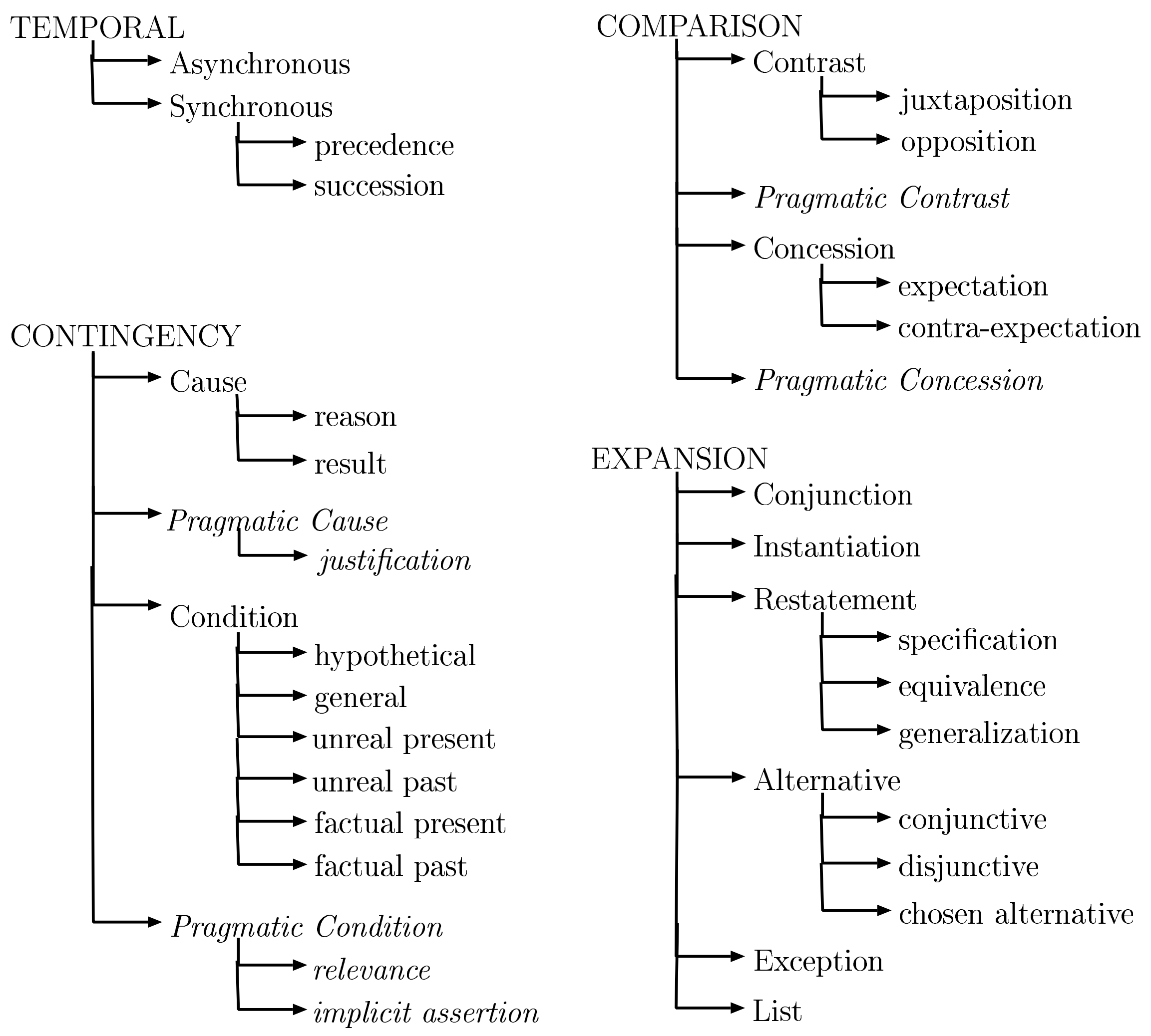

Figure 6.1: Overview of all relations occurring in the Penn Discourse Treebank

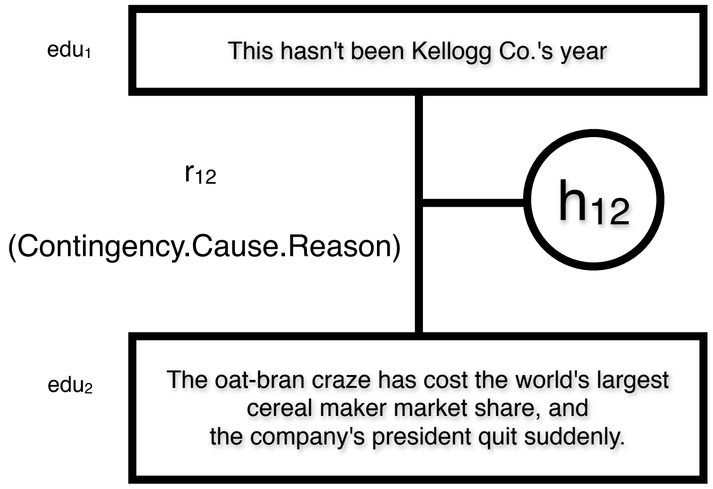
edu$_1$
This hasn't been Kellogg Co.'s year
r$_{12}$
(Contingency.Cause.Reason)
h$_{12}$
edu$_2$
The oat-bran craze has cost the world's largest cereal maker market share, and the company's president quit suddenly.

Figure 6.2: An example of the latent variable discourse parsing model taken from the Penn Discourse Treebank Dataset. The relation here is an example of a cause attribution relation.

ment. There are 16 level 2 relations in the PDTB, but the 5 least common relations only appear a handful of times in the dataset and are omitted from our tests, yielding 11 possible classes.

6.3 Approach

We incorporate weakly labeled data into our spectral discourse parsing model using a bootstrapping framework. The model is trained over several iterations, and the most useful weakly labeled sequences are added as labeled training data after each iteration. Our method also utilizes Markovian latent states to compactly capture global information about a parse sequence, with one latent variable for each relation in the discourse parsing sequence. Most discourse parsing frameworks will label relations independently of the rest of the accompanying parse sequence, but this model allows for information about the global structure of the discourse parse to be used when labeling a relation.

Specifically, each potential relation r_{ij} between elementary discourse units e_i and e_j is accompanied by a corresponding latent variable as h_{ij}. A graphical representation of one link in the parsing model is shown in Figure 6.2. According to the model assumptions, the following equality holds:

$$P(r_{ij} = r | r_{1,2}, r_{2,3}...r_{n+1,n}) = P(r_{ij} = r | h_{ij})$$

To maintain notational consistency with other latent variable models, we will denote these relation variables as $x_1...x_n$, keeping in mind that there is one possible relation for each adjacent pair of elementary discourse units.

For the Penn Discourse Treebank Dataset, the discourse parses behave like sequence of random variables representing the relations, which allows us to use an HMM-like latent variable model based on the framework presented in [32]. If the discourse parses were instead trees, such as those seen in Rhetorical Structure Theory (RST) datasets, we can modify the standard model to include separate parameters for left and right children, as demonstrated in [19].

6.3.1 Spectral Learning

For this application, we will once again be using the spectral latent-variable method described in section 2.1.2. For our original feature space, we use the rich linguistic discourse parsing features defined in [23], which includes syntactic and linguistic features taken from dependency parsing, POS tagging, and semantic similarity measures. We augment this feature space with a vector space representation of semantics. A term-document co-occurrence matrix is computed using all of Wikipedia and Latent Dirichlet Analysis was performed using this matrix. The top 200 concepts from the vector space representation for each pair of EDUs in the dataset are included in the feature space, with a concept regularization parameter of 0.01.

Similar to the Virtual Coach, abundant weakly labeled and unlabeled data are particularly useful for computing the empirical n-gram matrices, P_1, $P_{2,1}$, and $P_{3,x,1}$. Given the very high-dimensional, sparse feature space, computing these empirical matrices using only the labeled PDTB dataset yields unacceptably sparse results—even in the reduced dimensional subspace.

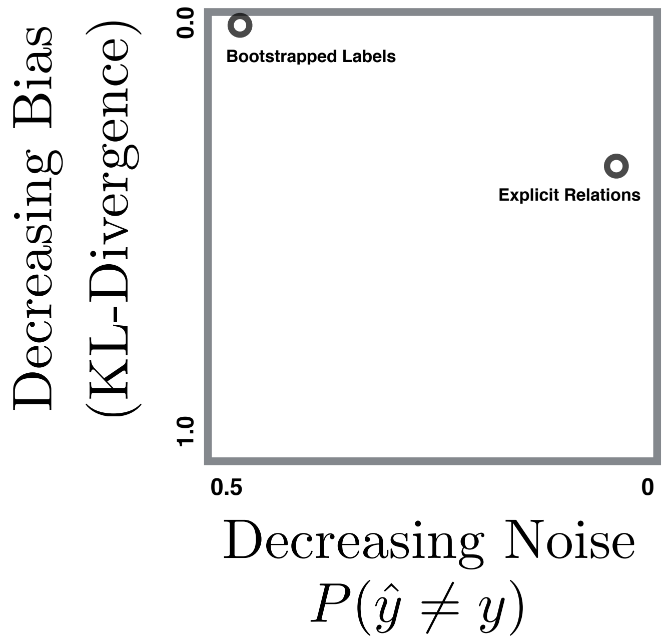

Figure 6.3: Empirical Arrangement of Datasources in the Noise-Bias Spectrum

6.4 Weakly Labeled Data

For this application, we will be dealing with two weakly labeled datasources, which are almost diametrically opposed to one another in the noise-bias feature space.

In the first datasource, we have bootstrapped labels sampled from unlabeled Wall Street Journal newsfeed articles. Because the source of this data is the same as the testing set, the bias measured by KL-divergence is very low (0.049). However, the difficulty of the prediction task leads to very high noise rates, even when using aggressive certainty thresholds. Additionally, as discussed in Chapter 3, this noise is strongly correlated with the existing beliefs of the classifier, making internal error correction much less useful. If we were given no other options, cross-training would be on possible method to improve the quality of these labels, but fortunately we have a second, extremely useful source of weakly labeled data.

The second source of weakly labeled data are *explicit relations, i.e.* those relations that have a known lexical connecter that induces the relation. These types of relations can be trivially identified using simple methods such as a linear SVM with an F_1 score of .87 or above [44]. This provides us with a nearly endless set of low-noise weak labels. Unfortunately, not all relations have explicit forms, and those that do can have very different lexical properties when taken in explicit form. To make things more concrete, we can consider an example where the explicit and implicit forms of a relation are quite different. For example, the temporal precedence relation indicates that the events of the head EDU take place before the tail. In explicit form, this can be indicated by a simple connective argument.

(1) "The Professor gave the lecture, **and then** he went home".

(2) "The Professor gave the lecture. It was later in the day that he arrived at home".

In this instance, we see significant lexical differences in the implicit and explicit forms of the relation. However, other relations may retain the same meaning by simply removing the explicit connective, such as this example of a causal relation.

(1) "The President's favorability is dropping, **because** people just don't think the economy is on track".

(2) "The President's favorability is dropping; people just don't think the economy is on track".

Using density sampling allows us to greatly prefer datapoints more akin to the second example. This makes explicit relations one of the most ideal scenarios for using active density-weighted certainty sampling. We begin with unlabeled Wall Street Journal and New York Times news feed text, and use the EDU segmenter described in [23] and the explicit relation classifier described in [44] to produce a weakly labeled training set. The most informative sequences in the weakly labeled training set are added to the labeled training set as labeled examples. Unlike most weakly labeled datasets, the confidence of the SVM predictions gives us an explicit measure of how likely a label is to be noisy. We capture this information using density-weighted certainty sampling (DCS). Specifically for a sequence of relations $r_1 ... r_n$ taken from a document, d, we use the following formula:

$$DCS(d) = \frac{1}{n}\sum_{i=1}^{n}\frac{\hat{p}(r_i)}{U(r_i)}$$

In this equation, $U(r_i)$ represents the distance from the decision boundary supplied by the explicit relation SVM. Density is denoted $\hat{p}(r_i)$, and this quantity measures the extent to which the text corresponding to this relation is representative of the labeled corpus. To compute this measure, we create a Kernel Density Estimate (KDE) over a 100 dimensional LDA vector space representation of all EDU's in the labeled corpus. We then compute the density of the KDE for the text associated with relation r_i, which gives us $\hat{p}(r_i)$. All sequences of relations in the weakly labeled dataset are ranked according to their average density-weighted certainty score, and all sequences scoring above a parameter ψ are added to the training set. The model is then retrained, the weakly labeled data re-scored, and the process is repeated for several iterations. In iteration i, the labeled data in the training set is weighted w_i^l, and the weakly labeled data is weighted w_i^u, with the weakly labeled data receiving higher weight in subsequent iterations. The KDE kernel bandwidth and the parameters ψ, w_i^l, w_i^u, and the number of hidden states are chosen in experiments using 10-fold cross validation on the labeled training set, coupled with a subset of the weakly labeled data.

We see an empirical noise rate of roughly $.05$ when selecting the top 40% of training data according to DCS. Further examination indicates that this noise is generally not attributed to misclassification by the SVM, but rather explicit relations becoming ambiguous when the explicit connector is removed.

(1) “The President’s favorability is dropping, **however** people don’t understand his vision for the country”.

(2) “The President’s favorability is dropping; people don’t understand his vision for the country”.

In the first example, we see a comparison-concession relation, where the author seems to be indicating that the President is a sort of misunderstood visionary. The second example appears

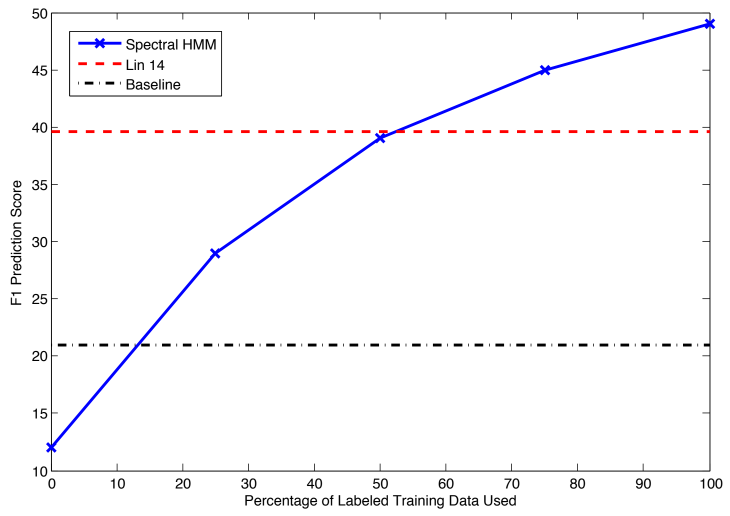

Figure 6.4: Empirical results for labeling of implicit relations.

to be more of an explanatory relation, where the second clause is providing evidence to support the first clause. Empirically, these ambiguities occur at a fairly low rate [12], but these instances are a larger source of error than explicit relation misclassification.

6.5 Results

Figure 6.4 shows the F_1 scores of the model using various sizes of labeled training sets. In all cases, the entirety of the weakly labeled data is made available, and 7 rounds of bootstrapping is conducted. Sections 2-22 of the PDTB are used for training, with section 23 being withheld for testing, as recommended by the dataset guidelines [54]. The results are compared against those reported in [44], as well as a simple baseline classifier that labels all relations with the most common class, $EntRel$. Compared to the semi-supervised method described in [28], we show significant gains in accuracy using similarly sized labeled datasets, although the weakly labeled dataset used in our experiments is much larger.

When the spectral HMM is trained using only the labeled dataset, with no weakly labeled data, it produces an F_1 score of 41.1%, which is comparable to the results reported in [44]. By comparison, the classifier using weakly labeled data is able to obtain similar accuracy when using approximately 50% of the labeled training data. When given access to the full labeled dataset, we see an improvement in the F_1 score of 7-9 percentage points. Recent work has shown promising results using CRFs for discourse parsing [23, 34], but the results reported in this work were taken from the RST-DT corpus and are not directly comparable. However, supervised CRFs and HMMs show similar accuracy in other language tasks [4, 53].

6.6 Conclusions

In this chapter, we have shown that we are able to outperform fully-supervised relation classifiers by augmenting the training data with weakly labeled text. The spectral optimization used in this approach makes computation tractable even when using over one million documents. In the subsequent chapter, we will take this parsing framework and apply it to a real world dataset taken from Massively Online Open Courses operated by Coursera. we will see that by leveraging weakly labeled data, we are able to learn an accurate response relation classifier using no hand labeled data for training.

Chapter 7

MOOC Thread Structure Identification

7.1 Introduction

We now turn to a practical application of the discourse parsing framework introduced in the previous chapter, concerning the analysis of conversations between students participating in a Massive Online Open Course (MOOC). These types of online courses offer tremendous potential for expanding access to quality education. However, as the number of students vastly outweigh the number of administrators and educators, it is vital that productive collaboration between students is fostered. Given the sheer volume of forum activity in a large MOOC, there is an opportunity to develop software techniques that can automatically help to analyze natural language discussions between students [2, 3] . This analysis can be used to clean and curate course forums, as well as deploy automated interventions to aid struggling students. We may, for instance, identify which students are most actively engaged in productive conversations with their fellow students, and these peer mentors could be leveraged when another student is found to be struggling.

Shallow, sentence level language analysis has proven useful in a variety of MOOC applications, however many of these methods ignore the higher level inter-sentential structure of online discussions. There is an opportunity to better understand conversations between students by identifying which posts within a thread are related to one another, which induces a directed relationship graph for the thread. Conversational structure, such as dialogue act sequences, can be very useful when assessing the quality of unstructured discourse between students [1]. Unfortunately, most of this work has relied on manually annotated conversational structure. In this chapter, we present a method to automatically discover forum thread structure in a weakly supervised manner that does not require costly, manual annotation of data. Instead, we rely on weakly labeled data inferred from a thread's metadata, or supplied directly by the students during normal forum usage.

Discourse parsing offers a useful framework for analyzing the high-level structure of a conversation or a block of text. In most discourse parsing work, semantic relationships are identified between *elementary discourse units* such as sentences or sentence fragments. This framework generally requires very large manually annotated datasets, such as the Penn Discourse Treebank [54]. Compiling this sort of annotated corpus is difficult for MOOCs, where supervised classifiers may not generalize across different course topics. Instead, we consider a simpler problem in which we seek to identify pairs of forum posts in which the second post is a direct response to the first. This allows us to leverage structured forum responses built into popular online platforms such as Coursera. Coupling these structured responses with a semi-supervised training framework allows us to train a discourse parser that does not require manually supplied training data.

In this chapter we leverage the spectral, hidden variable discourse parsing algorithm for learning the relational structure of conversations between students participating in a MOOC. We describe the specifics of the Coursera data, consisting of forum activity taken from two courses, and we provide empirical results on a human annotated testing set compiled from the data. The model is able to achieve predictive accuracy of .921 and .903 on the two course datasets, which is nearing the estimated human inter-annotator agreement of .939.

	Python	**Psychology**
Total Threads	3,234	1,488
Total Posts	24,963	5,244
Mean Posts/Thread	7.7	3.5
Total Comments	8,421	2,747
Labeled Threads	54	34
Labeled Pairs	1,120	916

Table 7.1: MOOC Datasets

7.2 Weakly Labeled Dataset

The dataset for our experiments consists of forum posts from two Coursera courses. Some statistics describing these datasets are shown in table 7.1. The first course, 'Learn to Program: The Fundamentals', focused on teaching the Python programming language to students with little computer science experience. The second dataset, based on a psychology course, was used in the shared task for the Modeling Large Scale Social Interaction in Massively Open Online Courses Workshop [71]. Posts are organized into threads, and each post is accompanied by a set of metadata variables, including the author of the post, the title of the thread, the timestamp, the number of votes given to the post by other students, as well as a tag indicating if the thread the post is contained in has been identified as resolved by the thread's creator. Additionally, each post is designated either as an unstructured member of the forum thread, or as a *comment* which is a structured method for forum participants to respond to another specific post.

A subset of the dataset was manually annotated for use in testing the model's predictive accuracy, and the sizes of these testing sets are also shown in table 7.1. These labels were generated by selected a random subset of forum threads, and then each unique pair of posts in the thread were labeled. It should be noted again that this labeled data is only used for evaluation, and no manually annotated data is required for training. Each pair of posts in this set was evaluated to determine if the second post in the pair was a direct response to the first, and a label was given for every ordered pair of posts taken from the threads selected for annotation. Five threads from the psychology dataset were labeled by two different annotators, and within those threads we

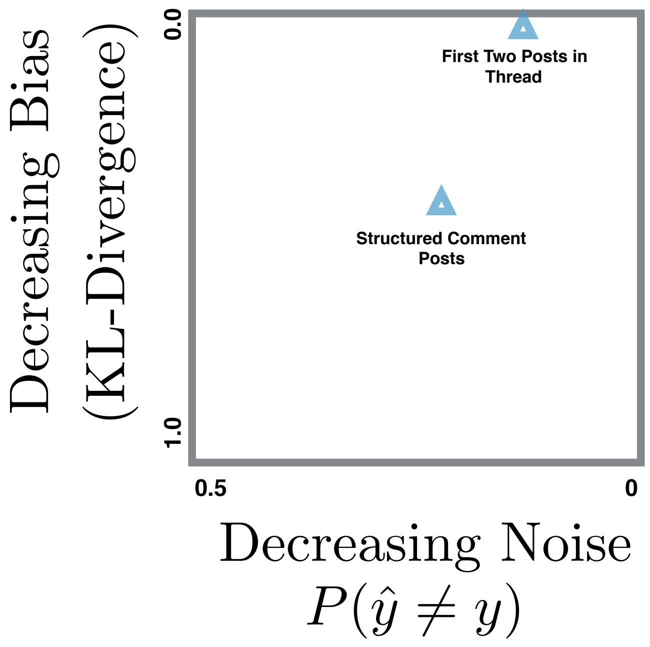

Figure 7.1: Noise-Bias Spectrum of Weakly Labeled MOOC

observed an inter-annotator agreement rate of .939. Of all possible pairs of posts in the testing sets, 24.7% of them were observed to share a response relation in the Python course, and 19.1% were related in the psychology course.

Figure 7.1 shows the noise-bias spectrum of the weakly labeled data used in this application. Structured comments were seen to be related to their parent post roughly 60% of the time, while the second post in a thread was observed to be a direct response to the first post 94% of the time. For the purposes of evaluation, only unstructured pairs of posts were used for testing. Structured comment posts and the first two posts in every thread were excluded. Surprisingly, the comment posts also demonstrated significant bias, as the lexical form of many of these posts differed significantly from general forum posts. Collectively, these two sources of weakly labeled data constitute one of the weakest and one of the strongest heuristically labeled datasets considered in all applications of this thesis.

7.3 Method

We once again use use Markovian latent states to compactly capture global information about a parse sequence, with one latent variable for each relation in the discourse parsing sequence. Most other discourse parsing frameworks label each relation independently, but our model allows information about the global structure of the discourse parse to be used when identifying an individual relation.

Our model considers each post to be one elementary discourse unit. Each thread of length t produces $t - 1$ sequences of posts. Specifically, we first consider all pairs of posts in the thread of distance 1 in chronological order, *i.e.* all adjacent posts. We then consider all pairs of post of distance 2, etc. This allows us to identify all responses in a sequence, not just relations between adjacent discourse units as seen in datasets such at the Penn Discourse Treebank. This is especially important for forum posts which are asynchronous, which may often result in relations between non-adjacent posts in a thread.

According to our assumption of Markovian structure, each potential relation x_t between elementary discourse units *edu*$_t$ and *edu*$_{t+1}$ is accompanied by a corresponding latent variable h_t, and conditioning on h_t makes x_t independent from $x_{1...t-1}$ and $x_{t+1...n}$.

For the feature space, we once again use the discourse parsing features defined in [22], which include syntactic and linguistic features taken from dependency parsing, POS tagging, and semantic similarity measures. The various metadata for each post is also included as features, as is the time between publication for each pair of posts, and the distance between their positions in the forum thread. We also include a feature indicating if the name of the author from the first post appears in the body of text in the second post.

In addition, a Latent Dirichlet Allocation (LDA) model is trained using 15 million English language tweets made to Twitter, and this model is used to compute a vector space representation of the semantics of each post. In the previous chapter, Wikipedia was used as the basis of the semantic vector space. However, with the forum posts we see much more casual language, including many misspellings, acronyms and textual emoticons. By using Twitter, another casual forum for language, we are able to capture some of the most common informal language practices. From the LDA model, the top 400 concepts for each post are included in the feature space, as is the cosine similarity of the vector space representations of the two posts. URLs, HTML tags, and Python code are removed from posts before the vector space representation is computed. LDA concepts related to math, science, and technology were more common in the Python dataset, while the psychology dataset included more concepts related to the social sciences and popular culture.

We use one variable in the observation space to denote the response relationship for a pair of posts, indicating whether there should be an edge between these posts in the discourse tree. If the second post in the pair is a comment directed toward the first post, we consider this a responsive pair and set the response variable to 1. Initially, all structured comments are labeled as related to their parent post, as are the first two posts in every thread. Together, these two sets of datapoints are used to train an initial model. For all other pairs in the dataset, the response variable is set according to the model's estimated probability that the pair constitutes a response relation given the other observed variables, and the latent state distribution. Initially this probability is set to the naive prior, but the estimates for the unlabeled data change every time the parameters are recomputed. Unlabeled pairs with a predicted relation probability surpassing a threshold ϕ have their labels set to 1. Because of the speed of the spectral method, we are able to construct estimates for the response variables and recompute the model parameters several times.

After five iterations of bootstrapping, we conduct internal error correction. Those comments with a predicted confidence below a threshold, ψ, become negative training samples, and the model is trained one more time. This step accounts for the fact that many students use the comment feature in unintended ways, which results in many false positive data points being added to the training set. A small parameter validation set was held out separately from the training set and used to select the threshold parameters ψ and ϕ.

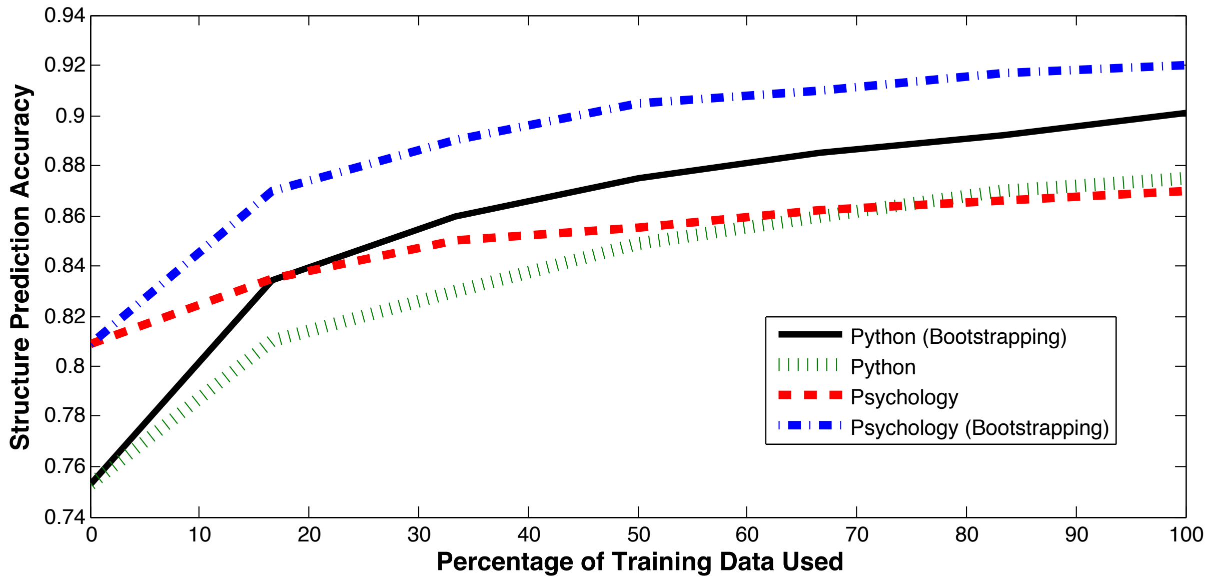

Figure 7.2: Empirical Prediction Results Using Coursera Data

7.4 Results

Figure 7.2 shows empirical results using the spectral HMM to predict response relations in the Coursera dataset as a function of the percentage of training data used. The training curves denoted with (Bootstrapping) include the stages in which the unlabeled data is incorporated into the training set and internal error correction is conducted. The other training curves include neither unlabeled data nor label correction.

We see a maximum classification accuracy of 0.921, and a maximum precision of 0.782. This predictive accuracy is nearing the estimated, human inter-annotator agreement of .939, which can be thought of as an upper-limit on predictive performance at this task. Without bootstrapping, when the model has only used structured comments and the first two posts of every thread for training, the maximum precision drops to 0.689. This indicates that leveraging unlabeled data and relabeling incorrect comments results in a 10 percentage point increase in prediction precision. For comparison, a naive classifier that predicts that no pairs of posts share a relation would produce an accuracy of 0.638 with 0 precision. We also note that the psychology dataset yields

Dataset	Precision
Psychology (Bootstrapping)	0.749
Psychology	0.675
Python (Bootstrapping)	0.782
Python	0.689

Table 7.2: Predictive Precision Results

a higher overall accuracy but a lower precision. The decrease in precision is likely due to the smaller size of the training set, while the increase in accuracy may simply be a consequence of the distribution being more skewed towards negative examples, leading to a higher baseline prior.

In these experiments, the HMM parameters were computed and used to produce estimates for the relation probability of all pairs in the training data that did not have a comment relationship. These estimates were then used to recompute the HMM parameters, and this process was repeated five times. The labels of the structured comments were then predicted, and all comments with a relation probability below ψ became negative training results, while unlabeled pairs with a confidence above ϕ became positive training examples.

Without using internal error correction on the comment structures, the precision in both datasets drops by approximately 4 percentage points. Using the value of ψ chosen with the validation set, we observe that roughly 30% of the structured comments have their labeled switched to negative. Figure 7.2 does not include results with comment prediction, but separate tests suggest the predictive recall for comments is nearly 88%. This indicates that the corrective step of the algorithm could be reducing the percent of false positive training samples from 40% to as low as 15%.

7.5 Conclusion

In this chapter we have leveraged the previously introduced parsing framework for learning the inter-post structure of online forum threads. Previous research has shown many applications in which discourse information is useful such as dialogue evaluation, thread resolution prediction, and thread curation. There are also other applications that have not been widely studied. For instance, the response relations between posts tell us explicitly which students are speaking to one another in the forums. This could help us identify potential peer-mentors that commonly answer questions for their fellow students, and these mentors could be leveraged by an automated intervention to minimize the workload for paid teaching assistants.

An opportunity for future work is to create new weakly supervised methods to learn multiple semantic relations between posts. While fully unsupervised methods might not reliably learn the most useful types of relations, semi-supervised methods or techniques that leverage existing structured data show promise. Regardless of the methods used, the sheer scale of MOOCs presents a fantastic opportunity for data-mining in a variety of tasks, but many existing language analysis methods are often bottlenecked by the need for annotated data. By better utilizing vast stores of weakly labeled data, we can build tools to benefit educators and students alike.

Chapter 8

Conclusions and Future Work

In this thesis, we have examined four very different applications in which availability of hand labeled data is a central hindrance to building an accurate model. Taking tasks ranging from healthcare and education to linguistic analysis we have greatly improved predictive performance using weakly labeled data. For tasks in which no pre-existing labeled dataset exists, we have seen that broad usage of weakly labeled data provides an avenue to make learning tractable. For applications in which labeled data is available, weakly labeled data can also significantly improve our ability to conduct inference and make predictions.

We have presented tools for combating common types of weakly labeled data. We have seen that weakly labeled data should also be weighted properly when training, and we have seen that while we do not need to know the noise and bias rates of a datasource, the weights chosen using a parameter validation set will be heavily influenced by the position of the dataset in the noise-bias spectrum. Furthermore, we have seen the usefulness of Internal Error Correction when dealing with noisy datasets. Unlike bootstrapping, which asks a model to generalize using existing information, IEC allows the model to remove inconsistencies and contradictions in a training set that could otherwise impact our ability to learn. For datasets with significant bias, density-weighted certainty sampling provides us with means to elevate the usefulness of a training set, given a sufficient volume of weakly labeled data. Taken together, these tools allow us to learn effective models, even without access to hand labeled training data.

With the four applications discussed in this thesis, we have examined a broad swath of the noise-bias spectrum. The simplest application, InContext, featured relatively high quality weakly labeled data, requiring only minor internal error correction. This application serves as an initial demonstration that weakly labeled data acquired using heuristics can supplement or replace labeled data when the cost of annotation is extremely high. Compared to InContext, the virtual coach includes a much larger, weaker training set. This allows us to bring our more sophisticated methods to bear, in particular a spectral method that makes working with large datasets tractable and improves performance by modeling the sequential structure that is inherent in the application. The computational efficiency of the spectral method becomes particularly significant when we exponentially increase the size of the training set, as is the case with the discourse parsing task. In this application, using density sampling to filter an extremely biased dataset allows us to acquire a nearly endless, high-quality training set for a subset of the discourse relations, and the spectral optimization gives us the speed we need to contend with such a dataset. When we take this framework into the real-world, with the MOOC dataset, we see a significant increase in label noise as well, which we are able to combat using internal error correction. Although these applications are collectively very diverse, we see that our empirical observations of the effects of weakly labeled data in the real-world very closely reflect the initial observations that we made using the basic simulation model. This suggests that the specific impacts of noise and bias we have seen are much more general than any one of these applications would suggest on its own.

One promising direction for future work is to develop generalized strategies for acquiring weakly labeled data. Many of the sources of weakly labeled data in this thesis are application specific. One of the most effective sources of weakly labeled data we have seen—explicit relations used in discourse parsing—may suggest a possible path to a general method of acquiring weakly labeled data. In this task, a simpler sub-task is identified, and unlabeled data is used to generate weakly labeled data. As we have seen, we are able to tolerate significant bias in this weakly labeled set if a large enough volume of unlabeled data is available. This could suggest a framework, similar to cross-training, in which a simpler classifier is used to generate weakly labeled data, which is then filtered using density-weighted certainty sampling. Further analysis is required to determine the general efficacy of such an approach, but if an application-agnostic method of generating weakly labeled data could be created, it would have the potential to make a tremendous impact on how we train data-driven models.

The metric for measuring the weakness of data we have presented in this work, the noise-bias spectrum, generalizes well and captures a wide variety of problems encountered when training with heuristically labeled data. However there is nuance not captured with this two-dimensional representation that could be improved upon in future work. One could imagine many additional dimensions that we could use to expand this spectrum into three or more dimensions, but one issue in particular is correlated noise. As we saw in section 3.3 noise that obeys a sequential structure does not significantly impact training compared to independent noise. However, if the noise of the labels in the training data correlates with the existing beliefs of the model, as in the case of bootstrapped labels, the weakly labeled data is significantly less useful. A bootstrapped dataset may appear very appealing when positioned in the noise-bias spectrum, but would be much less useful for training compared to a dataset in which the noise of the labeled does not correlate with the model's existing mistakes. Although this type of weakly labeled data appears to be an outlier, it shows one instance in which the noise-bias spectrum does not capture the entire picture. Future work could modify the definition of the noise axis of the noise-bias spectrum to account for specific types of problematic correlated noise such as this.

Data is becoming more abundant in every facet or our lives, but the time we have to annotate that data is not. It is imperative that we continue to build new methods to automate the learning process. Fully unsupervised learning is one exciting avenue of research, but it will never be appropriate for all tasks. The framework presented in this thesis allows us to incorporate large, unlabeled datasets, while continuing to benefit from the decades of research that have been poured into fully supervised classification. In the decades to come, as the availability of data continues to grow exponentially, machine learning with weakly labeled data may give us the tools to bridge the divide between the capabilities of the models we've built with the needs of the tasks on-hand. All together, the contributions presented in this thesis help to reduce the need for human annotations, in a time when the increasing availability of data is forcing us to rethink decades old methods. Only by embracing the changing landscape and leveraging the structure in unlabeled data can automated machine learning realize its full potential.

Bibliography

[1] David Adamson, Akash Bharadwaj, Ashudeep Singh, Colin Ashe, David Yaron, and Carolyn P Rosé. Predicting student learning from conversational cues. In *Intelligent Tutoring Systems*, pages 220–229. Springer, 2014. 2.2, 7.1

[2] Dyke G. Jang H. J. Rosé C. P. Adamson, D. Towards an agile approach to adapting dynamic collaboration support to student needs. *International Journal of AI in Education*, 24(1):91–121, 2014. 7.1

[3] Jaime Arguello, Brian S Butler, Elisabeth Joyce, Robert Kraut, Kimberly S Ling, Carolyn Rosé, and Xiaoqing Wang. Talk to me: Foundations for successful individual-group interactions in online communities. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 959–968. ACM, 2006. 7.1

[4] Pranjal Awasthi, Delip Rao, and Balaraman Ravindran. Part of speech tagging and chunking with hmm and crf. *Proceedings of NLP Association of India (NLPAI) Machine Learning Contest 2006*, 2006. 6.5

[5] Scott R Beach, Richard Schulz, Judith T Matthews, Karen Courtney, and Annette DeVito Dabbs. Preferences for technology versus human assistance and control over technology in the performance of kitchen and personal care tasks in baby boomers and older adults. *Disability and Rehabilitation: Assistive Technology*, (0):1–13, 2013. 5.5

[6] Alessandro Bergamo and Lorenzo Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *Advances in Neural Infor-*

mation Processing Systems, pages 181–189, 2010. 2.2

[7] Maarten AS Boksem and Ale Smidts. Brain responses to movie trailers predict individual preferences for movies and their population-wide commercial success. *Journal of Marketing Research*, 52(4):482–492, 2015. 3.1.2

[8] Byron Boots and Geoffrey J Gordon. Predictive state temporal difference learning. *arXiv preprint arXiv:1011.0041*, 2010. 2.2

[9] Byron Boots and Geoffrey J Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *AAAI*, 2011. 2.2

[10] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, Citeseer, 2000. 4.1

[11] M Keith Chen, Jonathan E Ingersoll Jr, and Edward H Kaplan. Modeling a presidential prediction market. *Management Science*, 54(8):1381–1394, 2008. 3.1.2

[12] Hannah Rohde Anna Dickinson Chris Clark and Annie Louis Bonnie Webber. Recovering discourse relations: Varying influence of discourse adverbials. In *Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics (LSDSem)*, page 22, 2015. 6.4

[13] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. Spectral learning of latent-variable pcfgs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 223–231. Association for Computational Linguistics, 2012. 2.2

[14] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. Experiments with spectral learning of latent-variable pcfgs. In *Proceedings of NAACL-HLT*, pages 148–157, 2013. (document), 2.2, 2.1.2, 2.1.2

[15] Shay B Cohen, Karl Stratos, Michael Collins, Dean P Foster, and Lyle Ungar. Spectral learning of latent-variable pcfgs: Algorithms and sample complexity. *The Journal of Machine Learning Research*, 15(1):2399–2449, 2014. 2.2

[16] Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, 2:59, 2006. 2.2

[17] M. Csikszentmihalyi and R. Larson. Validity and reliability of the experience sampling method. *The experience of psychopathology: Investigating mental disorders in their natural settings*, pages 43–57, 1992. 4.4.5

[18] L. Del Pero, S. Ricco, R. Sukthankar, and V. Ferrari. Articulated motion discovery using pairs of trajectories. 2015. 1

[19] Paramveer S Dhillon, Jordan Rodu, Michael Collins, Dean P Foster, and Lyle H Ungar. Spectral dependency parsing with latent variables. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 205–213. Association for Computational Linguistics, 2012. 2.2, 6.3

[20] Anil K Dubey. Using rough sets, neural networks, and logistic regression to predict compliance with cholesterol guidelines goals in patients with coronary artery disease. In *AMIA Annual Symposium Proceedings*, volume 2003, page 834. American Medical Informatics Association, 2003. 2.2

[21] Mohammad H Falakmasir, Zachary A Pardos, Geoffrey J Gordon, and Peter Brusilovsky. A spectral learning approach to knowledge tracing. 2010. 2.2

[22] Vanessa Wei Feng and Graeme Hirst. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 60–68. Association for Computational Linguistics, 2012. 2.2, 7.3

[23] Vanessa Wei Feng and Graeme Hirst. A linear-time bottom-up discourse parser with constraints and post-editing. In *Proceedings of The 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), Baltimore, USA, June*, 2014. 2.2, 6.3.1, 6.4, 6.5

[24] Robert Fisher and Reid Simmons. Smartphone interruptibility using density-weighted uncertainty sampling with reinforcement learning. In *Proceedings International Conference on Machine Learning and Applications (ICMLA)*, Hawaii, December 2011. 3.6

[25] Robert Fisher, Reid Simmons, Cheng-Shiu Chung, Rory Cooper, Garrett Grindle, Annmarie Kelleher, Hsinyi Liu, and Yu Kuang Wu. Spectral machine learning for predicting power wheelchair exercise compliance. In *Foundations of Intelligent Systems*, pages 174–183. Springer, 2014. 2.2, 3.6

[26] J. Fogarty, S.E. Hudson, C.G. Atkeson, D. Avrahami, J. Forlizzi, S. Kiesler, J.C. Lee, and J. Yang. Predicting human interruptibility with sensors. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 12(1):119–146, 2005. 2.2

[27] Jim Giles. Internet encyclopaedias go head to head. *Nature*, 438(7070):900–901, 2005. 3.1.1

[28] Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. Semi-supervised discourse relation classification with structural learning. In *Computational Linguistics and Intelligent Text Processing*, pages 340–352. Springer, 2011. 2.2, 6.5

[29] J. Ho and S.S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 909–918. ACM, 2005. 2.2, 4.4.1, 4.4.5

[30] E. Horvitz and J. Apacible. Learning and reasoning about interruption. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 20–27. ACM, 2003. 2.2

[31] E. Horvitz, P. Koch, R. Sarin, J. Apacible, and M. Subramani. Bayesphone: Precomputation of context-sensitive policies for inquiry and action in mobile devices. *User Modeling 2005*, pages 251–260, 2005. 4.1

[32] Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012. 2.1.2,

2.2, 6.3

[33] Michael I Jordany, Zoubin Ghahramaniz, and Lawrence K Sauly. Hidden markov decision trees. *Advances in neural information processing systems*, pages 501–507, 1997. 5.7

[34] Shafiq R Joty, Giuseppe Carenini, Raymond T Ng, and Yashar Mehdad. Combining intra- and multi-sentential rhetorical parsing for document-level discourse analysis. In *ACL (1)*, pages 486–496, 2013. 2.2, 6.5

[35] A. Kapoor and E. Horvitz. Experience sampling for building predictive user models: a comparative study. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 657–666. ACM, 2008. 4.4.5

[36] Firas Khatib, Frank DiMaio, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, et al. Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature structural & molecular biology*, 18(10):1175–1177, 2011. 3.1.2

[37] T. Kinnunen, E. Chernenko, M. Tuononen, P. Fr
"anti, and H. Li. Voice activity detection using mfcc features and support vector machine. In *Int. Conf. on Speech and Computer (SPECOM07), Moscow, Russia*, volume 2, pages 556–561. Citeseer, 2007. 4.4.2

[38] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 453–456. ACM, 2008. 2.2

[39] Ranjay Krishna, Kenji Hata, Stephanie Chen, Joshua Kravitz, David A Shamma, Li Fei-Fei, and Michael S Bernstein. Embracing error to enable rapid crowdsourcing. *arXiv preprint arXiv:1602.04506*, 2016. 2.2

[40] Michèle Lacoste, Rhoda Weiss-Lambrou, Magali Allard, and Jean Dansereau. Powered tilt/recline systems: why and how are they used? *Assistive Technology*, 15(1):58–68, 2003.

5.1

[41] D.D. Lewis and W.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. 4.1

[42] Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 343–351. Association for Computational Linguistics, 2009. 6.2

[43] Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, pages 1–34, 2012. 2.1.3

[44] Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184, 2014. 2.2, 6.2, 6.4, 6.5, 6.5

[45] Hsin-Yi Liu, Rosemary Cooper, Rory Cooper, Asim Smailagic, Dan Siewiorek, Dan Ding, and Fu-Chieh Chuang. Seating virtual coach: A smart reminder for power seat function usage. *Technology and Disability*, 22(1, 2):53–60, 2010. 5.1, 5.2

[46] Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *The Journal of Machine Learning Research*, 11:955–984, 2010. 2.2

[47] Daniel Marcu. *The theory and practice of discourse parsing and summarization*. MIT press, 2000. 6.2

[48] Elijah Mayfield, David Adamson, and Carolyn Penstein Rosé. Hierarchical conversation structure prediction in multi-party chat. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 60–69. Association for Computational Linguistics, 2012. 2.2

[49] A.E. Milewski and T.M. Smith. Providing presence cues to telephone users. In *Proceedings*

of the 2000 ACM conference on Computer supported cooperative work, pages 89–96. ACM, 2000. 4.1

[50] Ha Quang Minh, Marco Cristani, Alessandro Perina, and Vittorio Murino. A regularized spectral algorithm for hidden markov models with applications in computer vision. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2384–2391. IEEE, 2012. 2.2

[51] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. 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 1003–1011. Association for Computational Linguistics, 2009. 2.2

[52] Ankur Parikh, Shay B Cohen, and Eric Xing. Spectral unsupervised parsing with additive tree metrics. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: Long Papers*. Association for Computational Linguistics, 2014. 2.2

[53] Natalia Ponomareva, Paolo Rosso, Ferrán Pla, and Antonio Molina. Conditional random fields vs. hidden markov models in a biomedical named entity recognition task. In *Proc. of Int. Conf. Recent Advances in Natural Language Processing, RANLP*, pages 479–483, 2007. 6.5

[54] Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K Joshi, and Bonnie L Webber. The penn discourse treebank 2.0. In *LREC*. Citeseer, 2008. 3.1, 3.6, 6.1, 6.5, 7.1

[55] J. Ramirez, P. Yélamos, JM Górriz, and JC Segura. Svm-based speech endpoint detection using contextual speech features. *Electronics letters*, 42(7):426–428, 2006. 4.4.2

[56] Madhuri Reddy, Sudeep S Gill, and Paula A Rochon. Preventing pressure ulcers: a systematic review. *Jama*, 296(8):974–984, 2006. 5.1

[57] Stephanie Rosenthal, Anind K Dey, and Manuela Veloso. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *Pervasive Computing*, pages 170–187. Springer, 2011. 2.2, 4.1, 4.5

[58] Borhan Samei, Andrew Olney, Sean Kelly, Martin Nystrand, Sidney D'Mello, Nathan Blanchard, Xiaoyi Sun, Marci Glaus, and Art Graesser. Domain independent assessment of dialogic properties of classroom discourse. In *Educational Data Mining 2014*, 2014. 2.2

[59] Evan Sandhaus. The new york times annotated corpus ldc2008t19. *Linguistic Data Consortium*, 2008. 6.2

[60] A. Sasse, C. Johnson, et al. Coordinating the interruption of people in human-computer interaction. In *Human-computer interaction, INTERACT'99: IFIP TC. 13 International Conference on Human-Computer Interaction, 30th August-3rd September 1999, Edinburgh, UK*, volume 1, page 295. IOS Press, 1999. 2.2

[61] B. Settles. Active learning literature survey. *Machine Learning*, 15(2):201–221, 1994. 4.4.5

[62] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079. Association for Computational Linguistics, 2008. 4.1

[63] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010. 3.5.2

[64] Amirreza Shaban, Mehrdad Farajtabar, Bo Xie, Le Song, and Byron Boots. Learning latent variable models by improving spectral solutions with exterior point methods. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI-2015)*, 2015. (document), 2.2, 2.1.2

[65] D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F.L. Wong. Sensay: A context-aware mobile phone. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*, volume 248. Citeseer, 2003. 2.2

[66] Xiaowei Song, Arnold Mitnitski, Jafna Cox, and Kenneth Rockwood. Comparison of machine learning techniques with classical statistical models in predicting health outcomes. *Medinfo*, 11(Pt 1):736–40, 2004. 2.2

[67] L. Terveen, J. McMackin, B. Amento, and W. Hill. Specifying preferences based on user history. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, pages 315–322. ACM, 2002. 4.1, 4.2

[68] A. Toninelli, D. Khushraj, O. Lassila, and R. Montanari. Towards socially aware mobile phones. In *First Workshop on Social Data on the Web (SDoW)*. Citeseer, 2008. 2.2

[69] Bonnie Webber, Markus Egg, and Valia Kordoni. Discourse structure and language technology. *Natural Language Engineering*, 18(4):437–490, 2012. 2.2, 6.2

[70] Allan P White and Wei Zhong Liu. Technical note: Bias in information-based measures in decision tree induction. *Machine Learning*, 15(3):321–329, 1994. 5.5

[71] Diyi Yang, Miaomiao Wen, and Carolyn Rose. Towards identifying the resolvability of threads in moocs. *EMNLP 2014*, page 21, 2014. 7.2

[72] John Zimmerman, Anthony Tomasic, Charles Garrod, Daisy Yoo, Chaya Hiruncharoenvate, Rafae Aziz, Nikhil Ravi Thiruvengadam, Yun Huang, and Aaron Steinfeld. Field trial of tiramisu: crowd-sourcing bus arrival times to spur co-design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1677–1686. ACM, 2011. 3.1.1