Active, semi-supervised learning to utilize human oracles

Robert Fisher
Machine Learning Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3815
Email: rwfisher@cs.cmu.edu

Reid Simmons
Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213-3815
Email: reids@cs.cmu.edu

***Abstract*—We present an approach to interactive machine learning, in which unlabeled data is employed in conjunction with active learning to better utilize the valuable resources that the human oracles provide. We empirically evaluate the approach in two very different applications, smartphone interruptibility prediction and semantic parsing. In both applications, we show that the use of active, semi-supervised training results in an improvement compared to a traditionally trained classifier relying only on full-supervision with random sampling.**

I. Introduction

Historically, computer software has provided lay-users with only minimal capability to affect the functionality of their software. Without knowledge of computer programming, a user can only modify the software by manipulating pre-set preferences defined by the author of the software. However, with machine learning, software can be fundamentally modified after deployment, allowing for personalization and environmental adaptation. Yet, many systems that utilize machine learning are nevertheless delivered to users with static functionality. In order to allow our software to continue to improve post-deployment, as well as to expedite system training during development, we must refine the way our software interacts with human instructors.

Furthermore, a user's time is a precious resource, and learning algorithms should not give prompts for unnecessary feedback. Previous research has shown that users are willing to give instruction to a learning algorithm if they perceive the performance of the system as improving [24], yet questions to the user should be issued wisely and with discretion. One study showed that user willingness to answer questions diminished when they perceived that too many questions were being posed, even though the accuracy of the learning algorithm was continuing to improve [32]. This suggests that question answering alone may be insufficient to produce a good learning algorithm. Instead, we propose a mixed-initiative learning strategy that allows a model to improve while only consulting a user for the most pressing questions. Some of the key ideas for better utilizing human instructors include the following:

- Information from sources other than the user should be given to the learning algorithm whenever possible. This may entail pooling feedback from many users or providing the learning algorithm with a pre-existing training set.
- While asking a user for explicit feedback is an expensive operation, passively observing the user is not. In this way, we can collect a large corpus of unlabeled data. When combined with labels given by the user, the learning problem can now be viewed through the lens of semi-supervised learning.
- If the user is to be consulted, an active learning framework should be utilized to select the most informative question to pose. This process can also include a decision-theoretic component, wherein we only ask the user for feedback if the potential information gain surpasses the cost of user annoyance [24].

In this paper we present two applications that entail the use of human oracles.

In the first application, we seek to learn the interruptibility of users with mobile devices. In this setting the target function is unique to the user and the current social context, so most of the dataset must be collected post-deployment after the user has begun using the device.

In the second application, we turn our attention to learning a semantic model for natural language. Using a framework that incorporates human oracles has two primary advantages: i) we can hasten the (sometimes arduous) process of training the model by requiring less supervision, and ii) we gain the ability to continue improving the model post-deployment through user interaction.

II. Application Overview

A. Smartphone Interruptibility

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 [20]. 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. We introduce the *In-Context* smartphone application, which uses a combination of signal processing, active learning, and supervised machine learning to create a personalized policy for changing

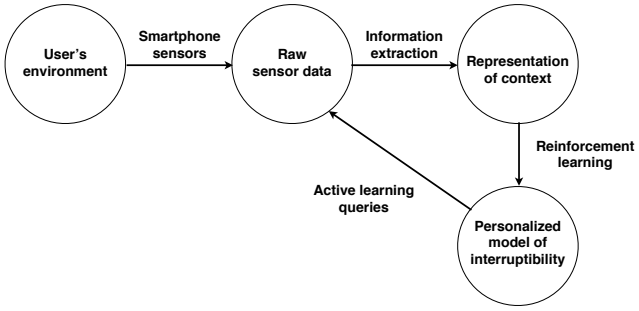

Fig. 1. The In-Context framework

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—for instance on a sensor-equipped power-wheelchair, we may wish to generate customized reminders for the user to conduct pressure relief exercises.

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 touch-screen 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. 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 In-Context system is shown in Figure 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. 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. We treat this behavior as a noisy reinforcement signal, because users may have forgotten to change their ringer setting, or they may be basing their decision on latent variables. For users who consistently change their ringer setting according to their preferences, the model trained on the passively collected 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.

Most previous attempts to determine user interruptibility, in mobile as well as desktop applications, have relied on active user input to determine their preferences [1], [13], [35]. Some of this work has explicitly considered the user's current interruptibility when deciding whether to issue prompts for input [24], but all of these systems perform poorly with users who are often unwilling or unable to respond to queries.

Our system expands on previous work in two central ways. First, we allow 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* [19], we propose the use of *density-weighted uncertainty sampling* [27], 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. 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 such as uncertainty sampling.

B. Semantic Parsing

Semantic parsing is the processing of natural language phrases, with the goal of producing a semantically equivalent representation in a machine-readable Meaning Representation Language (MRL). Interest in semantic parsing has grown significantly in the past few years, as state-of-the-art systems have begun to exceed human performance in certain non-trivial domains, such as Jeopardy! competition [7].

Although the accuracy of semantic parsers has risen dramatically, many existing parsers suffer from some key drawbacks. To begin with, the process of providing the algorithm with labeled training data is often costly and time-consuming. Many MRLs have a functional language structure, similar to Prolog. Annotating a sentence requires familiarity with the semantic ontology, and even for annotators familiar with the language, having a human parse a sentence to produce an MRL is slow and imprecise. Much work has been done recently to minimize the need for supervision in the training of semantic parsers [4], [16], [21].

Another disadvantage of most semantic parsers is that they do not leverage user interaction post-deployment to continue training. With a fully supervised parser, it would be unrealistic to imagine that a user would be willing or able to provide a correct MRL annotation of a sentence. However, as new methods of supervision have arisen in the literature, it is becoming possible to learn from users without requiring them to know anything of the semantic representation language. In some cases, the user may not even know that they are helping to improve the system.

We begin with a framework proposed by Clarke, *et al* [2]. This approach has the distinct advantage of relying on binary feedback, rather than full supervision. When an annotator provides a sentence to the parser, the sentence is parsed, and

the resulting MRL is fed into a response generator. The user then simply tells the system if this response was appropriate for the input or not. In this way, the MRL representation of the sentence is completely obfuscated from the annotator. This eliminates the need for the user to be familiar with the semantic language, and allows many sentence to be annotated much more quickly.

We seek to extend this approach by including bootstrapping and active learning in the framework. In this way, a corpus of unlabeled data can be used to improve the accuracy of the classifier without requiring annotation, and annotations will only be requested for the most informative sentences. Furthermore, the binary feedback mechanism could allow the system to continue to learn post deployment by evaluating user responses. For example, when a user says "thank you" after issuing a query to the semantic parser, we can infer that the query was parsed correctly and add it to the training corpus.

III. Related Work

Although human annotators have been used in machine learning since its inception, the problem of true interactive machine learning is less well studied [32]. Many systems that continually learn from users rely only on labeled data, without utilizing unlabeled data [24], or they do not leverage any sort of active learning framework to seek information from users [6], [39].

A. 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 [25], but more recent work has instead been using sensors to perceive the user's environment in order to achieve context-aware interruptibility [8], [11].

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 [37], 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 [10], [29]. 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 [24].

Nearly all previous work in the field of interruptibility has relied on actively asking the user questions about their preferences in different contexts [12], [14]. Unfortunately, this approach leads to scenarios where the system interrupts the user at inopportune moments in order to issue a query about their preferences in the current context. Some systems have alleviated this drawback somewhat by introducing a decision-theoretic component that allows the system to ask questions only when the cost of interruption is low [24]. However, when there is a high cost of asking questions, or when the user ignores the queries, many of these system fail to perform better than random guessing.

B. Semantic Parsing

A great deal of work has gone into creating accurate, fully supervised semantic parsers [41], [42]. These systems tend not to scale well to large amounts of data due to the costliness of annotating data with a full equivalent MRL. New approaches to semantic parsing have tried to lessen the burden of labeling data by using Markov Logic Networks [21], support vector machines [16], or binary feedback mechanisms [2]. In general, the semi-supervised parsers have lower classification accuracy than their fully supervised cousins when given a fixed amount of data. What has not been explored in detail is how well the semi-supervised approaches will scale, but it is believed that they will perform better than the full supervised models when only a fixed amount of time is devoted to annotation [2].

While active learning is a very mature and well studied topic [26], it has only been applied to semantic parsing in limited ways [34], [36]. Because the cost of annotation is high when building semantic parsers, it is thought that active learning can help to make the most of a limited amount of labeled data, but further study in this field is required [28].

IV. Datasets

A. Smartphone user study

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, as well as user activity and responses to incoming 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 2.

After the raw data was collected, it was passed through information extraction algorithms on board the smartphone, and the output of these algorithms was stored. The details of the information extraction are provided in section V. In particular, we represented a user's context using seven core pieces of information, described in Table I. We have done our

Fig. 2. User interface for data collection

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 I
THE REPRESENTATION OF A USER'S CONTEXT

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 [35].

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

B. GeoQuery

GeoQuery is a database containing facts about American geography, and it as been used to evaluate many semantic parsers [42]. The data contains natural language sentences (*e.g.* "What is the population of California?) with equivalent MRL representations (*e.g.* query(population(state('California')))). The data contains 1,130 such pairs, broken into one set of size 880 and one set of size 250. In our work, we set aside 150 samples from the larger set for the purpose of parameter validation, and the smaller dataset was used as a testing set.

The ontology present in the GeoQuery dataset includes many functions, such as $population(city)$, $elevation(state)$, and $length(river)$, as well as concepts for all the major US cities, rivers, mountains, and states. When the items of this ontology are combined, it allows users to ask questions such as "what is the largest state that the Mississippi river runs through?".

V. In-Context System Overview

This section describes the primary components of the In-Context 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 passively collected reinforcement signal (changes to ringer settings and responses to phone calls). 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.

A. Phone Posture Recognition

Previous work has shown that having knowledge of the user's physical activities can be used to help determine interruptibility [10]. 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.

B. 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 II. 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 [17], [23], there is one complication that arises in a mobile devices application: the

#	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 II
VOICE ACTIVITY DETECTION FEATURES

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

TABLE III
VOICE ACTIVITY DETECTION ACCURACY

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 III. 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 In-Context application.

C. Smooth Constraint Learning

Now we consider the problem of trying to smooth the sensor data predictions, to better fit the structure of the ground truth labels. This task can be though of as a constraint learning problem, wherein we constrain data over an interval to share a label, and we must discover the endpoints of these constraint intervals. There are two ways that these learned smoothing constraints can improve performance: applying constraints during training can improve parameter estimates, and applying constraints to the predicted labels during classification can improve accuracy. This approach can be generalized to other constraint learning tasks, for instance if we are trying to discover predicate arguments in an information extraction task, we may believe that some predicates obey a mutual exclusion constraint that could be discovered autonomously.

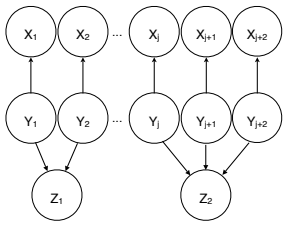

Fig. 3. An example element of $\mathcal{G}$ with interval agreement

We propose a restricted structure learning algorithm that learns constraints by searching over a class of possible constraints supplied by the user. The input to this algorithm is a class $\mathcal{G}$ of graphical model structures which contains possible expressions of constraints; the algorithm only searches over graphical model structures within this class. Exactly which structures are members of $\mathcal{G}$ depends on which constraints the user wishes to encode.

As an example of a class $\mathcal{G}$, we may believe that our data is arranged in a sequence, with labels adopting the same value over intervals (such as [11100001111]). This situation arises naturally in problems with temporal structure, such as speaker identification. In this case, we would like to enforce an agreement constraint over intervals of the sequence. For this problem, we define the class $\mathcal{G}$ such that (1) every label Y^t in the sequence is associated with exactly one true interval, represented by a constraint variable Z^i, (2) all labels within a given interval have the same value, and (3) each interval must span a minimum number of label variables. In this class, every label Y^t is the parent of exactly one constraint node, and each constraint node Z^i has a continuous set of parent variables $Y^t, ..., Y^{t+k}$. An example element of $\mathcal{G}$ is shown in Figure 3. As shown in Section VII-A this class $\mathcal{G}$ increases prediction accuracy and quality of parameter estimation significantly when applied to the smartphone sensor data. Note that $\mathcal{G}$ forces every label variable to participate in a constraint; without this requirement, we would trivially select a model with no constraint variables.

The object of structure learning is to select the element of $\mathcal{G}$ which maximizes the data likelihood. Say the structure $g \in \mathcal{G}$ contains constraint variables $\mathbf{Z}_g$. We can reparameterize the discriminative objective to include the graph structure:

$$\ell_D(\theta, g) = \sum_{i=1}^{n} \log P(\mathbf{Z}_g = \mathbf{1}, \mathbf{Y} = \mathbf{y}^i | \mathbf{X} = \mathbf{x}^i; \theta) + \sum_{i=n+1}^{n+m} \log P(\mathbf{Z}_g = \mathbf{1} | \mathbf{X} = \mathbf{x}^i; \theta)$$

Note that the objective is now a function of both the graph structure g and the model parameters θ. We propose to optimize this objective by alternately estimating each set of parameters. On the tth iteration, we compute:

$$\theta^{(t+1)} \leftarrow \arg\max_{\theta \in \Theta} \ell_D(\theta, g^{(t)})$$

$$g^{(t+1)} \leftarrow \arg\max_{g \in \mathcal{G}} \ell_D(\theta^{(t+1)}, g)$$

The above optimization over g can be performed efficiently for many classes $\mathcal{G}$, including the class of interval agreement

constraints.

D. 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. 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 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 I as features, and we used the hardware ringer switch as the classification label. However, it has been noted in previous work that many users forget to set their ringer switch to their preferred setting [20], so the setting of the hardware ringer switch will not always reflect the user's true preference. However, 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, rather than treat the ringer switch as a strict binary label, we think of it more as a reinforcement signal, which degrades over time. If it has been several hours since the user set the ringer preference, we are less confident in this signal, whereas if the switch has just been set, we are much more confident. To capture this behavior, equation 1 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 was 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.

$$W_i = \begin{cases} Y_i e^{-h/\lambda} & : h \leq 12 \\ 0 & : h > 12 \end{cases} \quad (1)$$

The distance function for the nearest neighbor classifier is given in equation 2. 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 $max(|h_i - h_j|, 4)$. 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.

$$D(C^i, C^j) = \sum_{k=1}^{6} d_k |f_d^i - f_d^j| \quad (2)$$

Using the distance function given by equation 2, we have the decision policy given in equation 3. 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 VII.

$$Pred(C) = I\left(\left[\sum_{i=1}^{k} \frac{W_i}{D(C, C^i)^2}\right] \geq 0\right) \quad (3)$$

E. Active Learning

Active learning is a framework in which a learning algorithm is able to query an oracle for the label of specific data-points. 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 [3]. In scenarios in which a labeling oracle is available, active learning has been shown to greatly increase classification accuracy [26].

Uncertainty sampling is a popular metric for selecting which points to query the oracle about. With uncertainty sampling, the data-points that the classification algorithm is most uncertain about are selected for labeling by the active learning oracle [19]. Entropy is a common measure of algorithm uncertainty, as given in equation 4. A high entropy value for a data-point, X, indicates high uncertainty.

$$H(X) = -\sum_{l=0}^{1} P(Y(X) = l|X) \log[P(Y(X) = l|X)] \quad (4)$$

While uncertainty sampling often works well, 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). Although the algorithm may be highly uncertain about an unusual context, such as the restaurant, this context is not representative of much of the user's activity, so labeling it will provide limited benefit.

Therefore, we propose the use of *density-weighted uncertainty sampling* [27]. In this framework, the algorithm favors asking the user to label data-points that the system is uncertain about, but which are also representative of a large number of

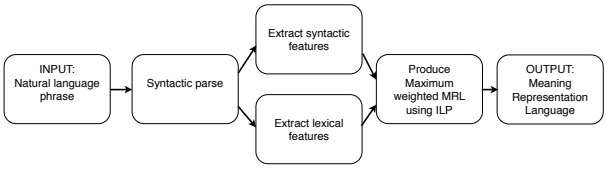

Fig. 4. Parsing Pipeline

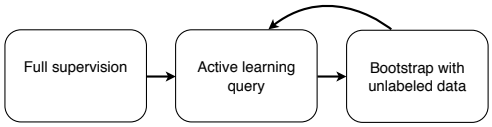

Fig. 5. Learning Regime

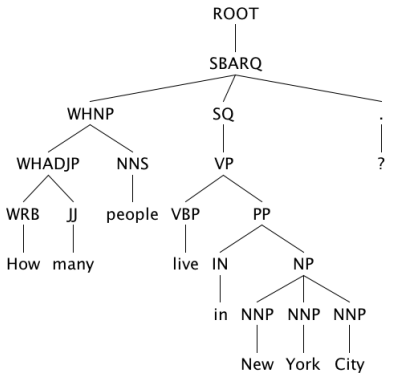

Fig. 6. Context-Tree Phrase Structure Tree

other samples in the data. The metric function for density-weighted uncertainty (DW) sampling is given in equation 5. In this equation, $sim(X, Y)$, is a function representing how similar two points X and Y are. For our similarity function, we used the squared reciprocal of the distance function from equation 2.

$$DW(X) = H(X) \sum_{i=1}^{n} sim(X, X^i) \tag{5}$$

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 In-Context 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 [10], [15].

VI. Semantic Parsing System Overview

This section describes the framework used for the semantic parsing system. We combine lexical features [5] with syntactic features given by a dependency tree [18]. Weights over these features provide a likelihood that a given MRL is correct. Previous work has shown that Integer Linear Programming (ILP), while NP-Hard in general, is very effective when used for the sparse problem of producing a maximum-weighted MRL given a set of features and weights [22]. This framework for parsing a given sentence was proposed in [2], and a graphical representation of the parsing pipeline is shown in figure 4.

The MRL that is output by the parser after the pipeline is executed depends on the vector of feature weights, W. Previous work has shown that these weights can be learned using only binary user feedback (i.e. the MRL produces a correct response, or an incorrect response). In our work, we extend this to include several new modalities of training. The training steps are shown in figure 5. The process begins by providing the parser with a small set of {Sentence, MRL} pairs with which a working baseline parser is constructed. At this point, queries are made to the human annotator according to the active learning framework described in section VI-D. After a label is received, and the weight vector is updated, the sentences in the unlabeled dataset are parsed, and the algorithm's confidence in these parsings is recorded. Any unlabeled sentences with sufficient confidence are given back to the parser with the predicted MRL used as a ground truth label.

A. Syntactic Features

The Stanford Parser Probabilistic Context-Free Grammar (PCFG) suite was used to extract syntactic information from the input sentence [18]. Some of this syntactic information is computing the lexical features of the sentence, described in section VI-B, while some information from the syntactic information is used for separate syntactic features. Figure 6 shows an phrase structure tree generated by the syntactic parser when applied to the sentence "How many people live in New York City?" This tree contains part of speech tags on each leaf of the tree, and also whole noun phrases, such as "New York City".

In addition to the phrase tree, the typed dependency tree of the sentence is also computed. This tree is used to predict the semantic relatedness of words in the sentence. For example, in the sentence "the man with the dog likes to run", we would like to know if the predicate *likes_to_run()* applies to the man or the dog. To help understand the relatedness of two phrases in a sentence, we compute the normalized distance in the dependency tree between the head words of these two phrases. In this way, we could infer that *likes_to_run(man)* is a more

probable MRL than *likes_to_run(dog)*, even though they are both valid semantic representations.

B. Lexical Features

Our system uses WordNet [31] to compute the lexical similarity of two natural language phrases. In WordNet, words are organized into synonym sets (*synsets*), for example the words "humans", "mankind", and "humanity" would share a common synset. The sysnsets are then organized into tree, with parenthood representing hypernym-hyponym relationships. As such, the "humans" synset will be a descendent of the "mammals" synset, but not the "invertebrate" synset. Nouns, verbs, and adjectives all lie in separate hierarchies.

To compute the lexical similarity of two words using the WordNet, we utilize the distance of these two words to their least common subsumer (lcs) in the WordNet tree [18]. All words in a given hierarchy descend from some common general concept. We denote l_1 the distance to the lcs from word 1, similarly for l_2. We further denote the depth of the least common subsumer as $d(lcd)$. The WordNet similarity is described by the following function.

$$WN(w_1, w_2) = \begin{cases} 0.3^{l_1+l_2} & \text{if } \; l_1 + l_2 \leq 3 \\ 0.3^3 & \text{if } \; l_1 + l_2 \leq \frac{2}{3} \cdot d(lcs) \\ 0 & Otherwise \end{cases}$$

We also use some syntactic information to inform our computation of lexical similarity. We first use the phrase structure tree to create a set of *constituents* from the input sentence, which includes both the literal words in the sentence (*e.g* people) as well as chunked noun phrases (*e.g.* new_york_city). At this stage we also remove determiners, conjunctions, and the like are removed from the set of constituents. Furthermore, we use part of speech tags to map the word to the correct synset tree. For instance, only when used as a noun will the constituent "in" have strong lexical similarity with the concept "indiana".

C. Integer Linear Programming

Given a sentence, a set of weights, $\mathbf{w}$, and feature functions $\mathbf{\Phi}$, we wish to find the MRL representation that best represents the semantics on the input. To accomplish this, we represent the mapping of the sentence into an MRL with binary variables and use an Integer Linear Program to assign values to these variables.

We denote our ontology as D and our input as X. X contains the constituents of the sentence. *First-order* variables indicate if text phrase c is mapped to ontological item s—we denote this variable $\alpha_{c,s}$. Items in the ontology may represent concepts (such as New York City) or functions (such as $population[x]$). Every concept has a type (city, river, etc), and every function has a list of valid arguments ($population[x]$ requires that x is of type city or state). To represent a valid MRL, we must consider how these ontological items are composed functionally. To represent this, we use *second-order* variables. For two items from the input, $c, d \in X$, and two items from the ontology, $s, t \in D$ the variable $\beta_{cs,dt}$ represents that these items are composed in the form $d(t)$.

The formulation of the integer linear program is given in Table IV. We wish to select the MRL that maximizes the linear weights multiplied over the lexical features Φ_1 and the syntactic features Φ_2. We use several ILP constraints to ensure that the values assigned to these variables after optimization corresponds to a valid MRL. The first constraint restrains all α and β variables to be binary, because those variables denote which ontological items are present and how they are composed. The second constraint requires that every constituent from the input text corresponds to exactly one item in the ontology (which may be the $NULL$ concept). The third constraint requires that second-order variable $\beta_{cs,dt}$ is active if and only if first-order variables α_{cs} and α_{dt} are active.

Integer Linear Programming is NP-Hard, and the problem as we have formulated it will produce an exact solution. If an input sentence has $|X| = n$ constituents, and the ontology is of size $|D| = m$, there are $O(mn)$ optimization variables for the ILP to consider. This fast becomes intractable, even with a modestly sized ontology. Fortunately, the number of variables that could possibly be active is much lower. We leverage two properties of the problem structure to regain tractability while maintaining the optimality of our final solution. These two properties are lexical feature sparseness and typed dependencies.

We first use lexical feature sparseness to reduce the number of first-order features, which in turn reduces the number of second-order features. The average synonym set in WordNet contains 7.82 entries [40]. Two words will usually only have non-zero similarity according to the WordNet metric if the larger distance to the least common subsumer is no more than three. This means that most words will have non-zero lexical similarity with only around 100 words, out of the 250,000 common words in the English dictionary. By automatically setting $\alpha_{cs} = 0$ when the lexical similarity between c and s is 0, we can reduce the number of first order variables considerably. We also only need to consider a second-order variable if both of its first-order variables could be active, thus the decrease in second order variables is even more dramatic.

Secondly, we can use the typing of items in the ontology to restrict the number of second-order variables. Second order variable $\beta_{cs,dt}$ can only be active if the composition $s(t)$ is valid, meaning that s is a function and t has a valid-type to be an argument to s. We explicitly set $\beta_{cs,dt} = 0$ whenever these requirements are not met.

Using these two methods for reducing the dimensionality of the optimization problem, we are typically left with an ILP over 200-400 variables, rather than the one million variables we would need contend with if no pre-processing was done. This makes the exact inference tractable to run in an interactive system, where promptness is essential.

D. Active Learning & Bootstrapping

In this section, we present three candidate metrics for selecting a sentence to be approved by a human annotator.

Maximize:

$$\sum_{c \in X} \sum_{s \in D} \alpha_{cs} \cdot \mathbf{w^T} \Phi_1(X, c, s) + \sum_{c,d \in X} \sum_{s,t \in D} \beta_{cs,dt} \cdot \mathbf{w^T} \Phi_2(X, c, d, s, t)$$

Objective Function

Subject to:

$\forall (c, s) \quad \alpha_{c,s} \in \{0, 1\}$

$\forall (c, d, s, t) \quad \beta cs, dt \in \{0, 1\}$

(All variables are binary)

Constraint I

$$\forall c \sum_{s \in D} \alpha_{c,s} = 1$$

(Every constituent mapped to exactly one item in Ontology)

Constraint II

$$\forall (c, d, s, t) \quad \frac{\alpha_{c,s}}{2} + \frac{\alpha_{d,t}}{2} \geq \beta_{cs,dt}$$

(An ontological combination is active if and only if both its constituents are active)

Constraint III

TABLE IV
INTEGER LINEAR PROGRAM FORMULATION

Given a corpus of unlabeled sentences, these metrics are computed for each sentence. An MRL is constructed for the sentence that is selected, and the GeoQuery database is queried with the MRL. At this point, the original sentence and the database response are given to the user, who indicates to the system that this pair is correct or incorrect. An example pair that could be given to the user is <"What is the capital of New York?", "Albany">.

1) Maximum weighted sampling:

$$w(X) = \frac{\sum_{\{\alpha_{c,s}=1\}} \mathbf{w^T} \Phi_1(X, c, s) + \sum_{\{\beta_{cs,dt}=1\}} \mathbf{w^T} \Phi_2(X, c, d, s, t)}{\sum w_i}$$

With this metric, we are leveraging the notion that samples which are labeled as correct by the user are much more informative than samples which are labeled as incorrect. This is due to the fact that a correct parsing implicitly gives the algorithm a good MRL for the sentence, whereas an incorrect parsing only states that the given MRL was incorrect. In the equation above $\{\alpha_{c,s} = 1\}$ denotes the set of first-order variables that are active after the ILP has been run. $\{\beta cs, dt = 1\}$ is defined similarly.

This approach to active learning will supply the user with sentences that we already believe are correctly parsed. While this will give the algorithm more "correct" labels, which are more informative than "incorrect" labels, we will not be improving the performance of the parser on labels which are likely incorrect to begin with.

2) Density sampling:

$$density(X) = \sum_{c \in X} \sum_{Y_i} \frac{1}{|Y_i|} \sum_{d \in Y_i} WN(c, d)$$

With density sampling, we wish to select sentences which are representative of many other sentences in the unlabeled dataset. For given sentence X, we compare the similarity of X to all other sentences in the corpus, Y. We use normalized lexical similarity between words to compute the similarity of these sentences. We assume that X and Y_i have already been syntactically parsed, the constituents have been chunked, and determiners and conjunctions have been removed.

E. Weighted density sampling

$$WD(X) = w(X) \cdot density(X)$$

For our final metric, we combine the two approaches described above. This will result in an active learning scheme that will sampling from the denser regions of the unlabeled dataset, while preferring parsings we believe are correct.

F. Bootstrapping

To conduct bootstrapping between active learning queries, we leverage the sentence weight described above. Using a validation set of unlabeled sentences, we compute $\hat{z}_{95}$, the 95^{th} percentile of weights for the current model. Any sentences that have weight larger than this threshold are labeled as positive and added to the training set.

$$\hat{z}_{95} \leq w(X)$$

VII. RESULTS

A. Smoothing Constraints

We now turn to the smoothing problem, as described in Section V-C. We will use a speaker identification task to evaluate the algorithm: given an audio signal, we wish to determine which of three known speakers is talking during each timestep[1]. This is a particularly difficult learning problem because of temporal correlation in the audio signal—if timestep t is assigned an incorrect label, then all other timesteps in the temporal neighborhood of t are also likely to be misclassified.

We will leverage the interval agreement constraint described in Section V-C, and make predictions using a Gaussian Mixture Model on Mel Frequency Cepstral Coefficients as in [33]. Previous work has shown that techniques like Hidden Markov Models do not work well in similar settings due to sudden drastic changes in the distribution, as occurs when there is a change of speakers [30].

We begin with 20 seconds of labeled data for each speaker (the set L), as well as another 225 seconds of unlabeled data for each speaker (the set U). To make predictions, the model clusters the data from each speaker using k-means ($k = 30$, selected using cross-validation), then fits one multivariate

[1]The audio data used for testing is used courtesy of the Sphinx Speech Consortium, and is available at http://www.speech.cs.cmu.edu/databases/pda/

TABLE V
PARAMETER ESTIMATION WITH STRUCTURE LEARNING. ENTRIES IN BOLD ARE STATISTICALLY SIGNIFICANT COMPARED TO THE ENTRY ABOVE THEM ($p < .05$ USING A TWO-SIDED t-TEST).

Method	Accuracy
Supervised, trained only on L	67.6
Semi-supervised, without constraints	**69.7**
Semi-supervised, fixed constraints	**72.6**
Semi-supervised, learned constraints classified with $P(Y\|X;\theta)$	**77.0**
Fully supervised on L and U classified with $P(Y\|X;\theta)$	**78.8**
Semi-supervised, learned constraints classified with $P(Y\|X,Z=1;\theta)$	**94.4**
Fully supervised on L and U classified with $P(Y\|X,Z=1;\theta)$	**95.3**

Gaussian to each cluster. The purpose of clustering is to produce a set of small peaked Gaussians, rather than one large highly variant Gaussian. We train the parameters of this mixture using the algorithm from Section V-C. As baselines, we trained the same model with (1) no constraints and (2) fixed constraints over intervals of 50 timesteps. Finally, we trained a classifier with the ground truth labels of the unlabeled data to estimate the maximum performance we can expect from a semi-supervised classifier. The results of these comparisons are shown in Table V.

After the model parameters have been learned using the semi-supervised approach, the final accuracy can be improved further by learning the set of constraints g for the testing set, and using $P(Y|X,Z=1;\theta)$ for classification rather than $P(Y|X;\theta)$—this has the effect of smoothing the final output. The final two entries in Table V use this method. Figure VII-A shows the impact of using the interval constraints to smooth the output by overlaying sequences of predicted labels with the ground truth. The dots indicate the predicted label at each timestep, and the dotted line indicates the true endpoints of each interval. The top two graphs in this figure do not use output smoothing, while the bottom two graphs do.

An important point of this evaluation is that there are two distinct manners in which constraint learning improves performance. Table V shows that training the semi-supervised model with learned constraints increases accuracy by 8%. Applying learned constraints to the output improves performance by an additional 16%.

This approach to smoothing was applied to the output of the voice activity detection and activity recognition algorithms used for the In-Context smartphone application.

B. Smartphone interruptibility

Figure 8 shows a comparison of four different classifiers for predicting ringer preferences. We evaluated the nearest-neighbors algorithm described in section V-D, 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

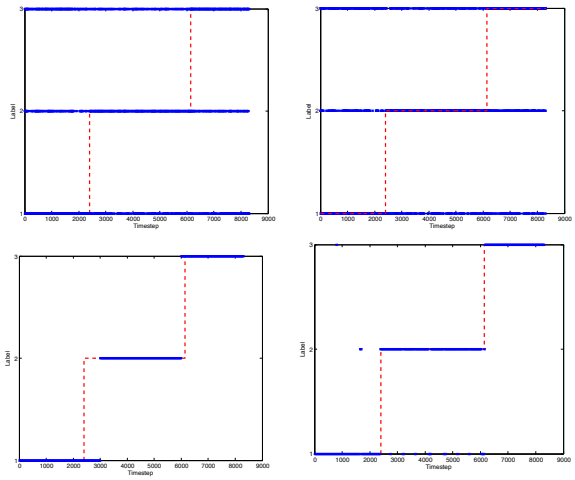

Fig. 7. Clockwise from top left: un-smoothed semi-supervised without constraints (69.7), un-smoothed constrained semi-supervised (77.0), fixed constraint smoothing (91.0), learned constraint smoothing (94.4)

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

TABLE VI
EFFECTS OF VOICE ACTIVITY DETECTION AND PHONE POSTURE RECOGNITION (INFORMATION EXTRACTION)

detection and phone posture recognition). From the summary of experimental results, shown in table VI, 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 9. 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 un-

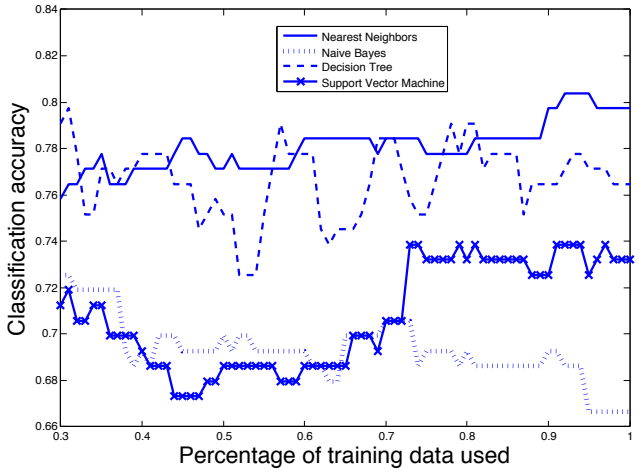

Fig. 8. Comparison of preference classifiers

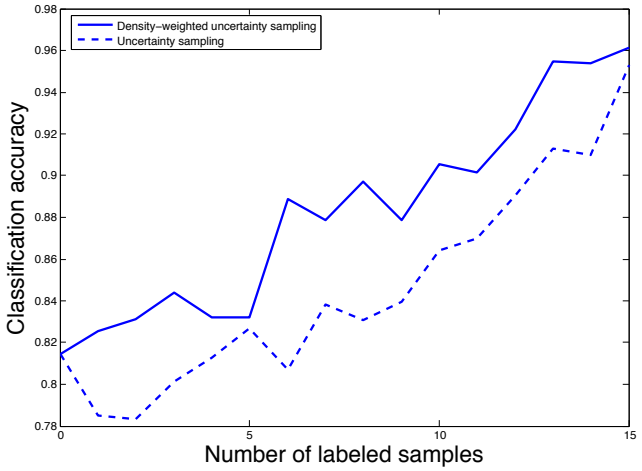

Fig. 9. Interruptibility learning curve

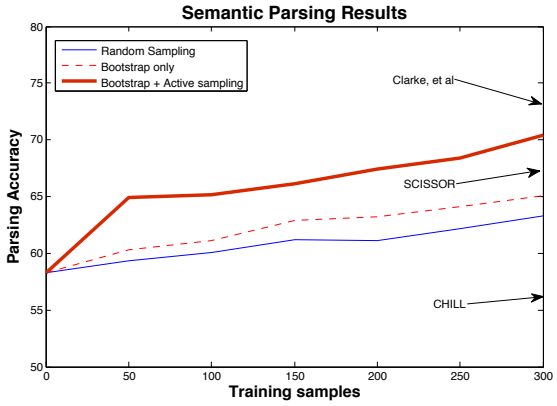

Fig. 10. Semantic parsing learning curve

C	Correct sentences	Accuracy percentage
0.3	158	63.2%
0.01	169	67.6%
0	186	74.4%

TABLE VII
THE EFFECT OF PRE-PROCESSING THRESHOLD C ON PARSING ACCURACY

certainty sampling. When given the maximum number of active labels (15), the classifier with density-weighted 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 $81.46\% \pm 6.20\%$. However, we note that there was one user who did not set his hardware ringer switch consistently, so the classifier with no active labels attains only 52.0% accuracy for this user. The other four users attain an average passive classifier accuracy of 87.82%.

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. 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 [24], we are able to produce comparable accuracy with much 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.

C. Semantic Parsing

The results of the GeoQuery evaluation of the semantic parser are shown in Figure 10. As expected, the inclusion of bootstrapping and active learning improve the performance of the parser compared with simply using random sampling to select data to be labeled. While using the weighted density active learning metric described in section VI-E, the accuracy of the parser improved by 7.1%.

Results from other semantic parsers on the GeoQuery dataset are shown on Figure 10, including SCISSOR [9] and CHILL [42]. SCISSOR and CHILL are capable of interactive speeds for parsing on consumer hardware, while Clarke *et al* is not. The accuracy of our parser exceeds that of the other real time parsers, while remaining competitive with the more computationally intensive unconstrained optimization used in the Clarke parser.

1) ILP Pre-Processing Experiments: For comparison, the semantic parser was executed with varying degrees of ILP pre-processing. As described in section VI-C, we explicitly set the value of α_{ij} to 0 if the feature value (lexical similarity) is below some threshold C. Because the lexical similarity metric is based on distance in the WordNet synonym trees, WNSim values only occur at specific discreet values. (0, 0.009, 0.27, etc). Setting $C = 0.3$ results in an average of around 200-400 optimization variables, and allows for parsing to be executed in under a second. Going down below of lexical similarity values ($C = 0.01$) results in approximately 5,000 ILP variables, and a parsing time on the order of 10 seconds. Setting $C = 0$ results in an optimal optimization, as was used in [2]. In this case, there is often hundreds of thousands of variables in the ILP optimization, and parsing takes 2 to 3 minutes on a modern, consumer-level, dual-core processor.

From Table VII, we can see the effects of the parameter C on classification accuracy. Active learning and bootstrapping were not used in the experiments described in this table. Training data was selected using random sampling without replacement. We see that when $C = 0$, the results we observe are quite similar to those described in [2], wherein the author's report an accuracy of 73.2%. These results imply

that the sacrifices made for parser speed are indeed having a detrimental effect on parsing accuracy.

VIII. Conclusions and Future Work

We see that both active sampling and semi-supervised learning on passively collected data are beneficial for classification accuracy in both of these applications. Future work should include further evaluation on the effects of using active sampling in an on-line setting. We conjecture that these sampling schemes could also be combined with a decision theoretic framework to improve user satisfaction as well as classification accuracy.

We are looking into new applications in both the context-awareness and semantic parsing domains.

The algorithms used for smartphone interruptibility are very general, and other applications should be explored in the future. The authors are currently working to bring this work into a smart-wheelchair application, in which the user of the chair is reminded to conduct pressure relief exercises according to a personalized model of interruptibility. Smart homes and automobiles would be other avenues for future research. The representation of context described in this paper could also be transmitted off the phone to allow for use in other devices, for instance a context-aware smartphone could notify the user's house that the user is on their way home, allowing the lights and heat to be activated in preparation for their arrival. As more of the devices in our environment become instrumented with sensors, we believe the importance of context-aware computing will continue to grow, as will the pressure to minimize the annoyance of being interrupted by those self-same devices.

Results from the smartphone application also lead us to believe that post-deployment training of the semantic parser would be possible as well. The parsing algorithm will be deployed on an interactive robot, called the GameBot, to be deployed in a public space. By collecting user responses to the robot's dialogue, we suspect that we may be able to infer in some instances when the robot has correctly or incorrectly parsed an input, which would allow us to continue adding to the training corpus in an online fashion. However, this approach comes with risk of adversarial users trying to manipulate the robot by corrupting the semantic model. We are investigating the possibility that we can use facial recognition software to build a persistent model of user trust. Questions with known answers could be mixed in with true questions to gauge a user's honesty when interacting with the system, similar to *ReCaptcha* [38].

Acknowledgments

This material is based upon work supported by the National Science Foundation under Cooperative Agreement EEC-0540865 as well as by a National Science Foundation Graduate Research Fellowship. We also acknowledge the Pittsburgh chapter of the American Rewards for College Scientists (ARCS) program for their generous support.

References

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

[2] J. Clarke, D. Goldwasser, M.W. Chang, and D. Roth. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27. Association for Computational Linguistics, 2010.

[3] 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] K. Deschacht and M.F. Moens. Semi-supervised semantic role labeling using the latent words language model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 21–29. Association for Computational Linguistics, 2009.

[5] Q. Do, D. Roth, M. Sammons, Y. Tu, and V.G.V. Vydiswaran. Robust, light-weight approaches to compute lexical similarity. Technical report, Technical report, Computer Science Department, University of Illinois, 2009. URL http://cogcomp. cs. illinois. edu/papers/DRSTV09. pdf, 2010.

[6] J.A. Fails and D.R. Olsen Jr. Interactive machine learning. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 39–45. ACM, 2003.

[7] D. Ferrucci. Build watson: an overview of deepqa for the jeopardy! challenge. In *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, pages 1–2. ACM, 2010.

[8] 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.

[9] R. Ge and R.J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 9–16. Association for Computational Linguistics, 2005.

[10] 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.

[11] 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.

[12] E. Horvitz, P. Koch, and J. Apacible. Busybody: creating and fielding personalized models of the cost of interruption. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 507–510. ACM, 2004.

[13] 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.

[14] S.S. Intille, J. Rondoni, C. Kukla, I. Ancona, and L. Bao. A context-aware experience sampling tool. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 972–973. ACM, 2003.

[15] 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.

[16] R.J. Kate and R.J. Mooney. Semi-supervised learning for semantic parsing using support vector machines. In *Human Language Technologies 2007: the Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 81–84. Association for Computational Linguistics, 2007.

[17] 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.

[18] D. Klein and C.D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

[19] 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.

[20] 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.

[21] H. Poon and P. Domingos. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 1–10. Association for Computational Linguistics, 2009.

[22] V. Punyakanok, D. Roth, W. Yih, and D. Zimak. Semantic role labeling via integer linear programming inference. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1346. Association for Computational Linguistics, 2004.

[23] 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.

[24] S. Rosenthal, A. Dey, and M. Veloso. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. *Pervasive Computing*, pages 170–187, 2011.

[25] 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.

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

[27] 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.

[28] B. Settles and X. Zhu. Behavioral factors in interactive training of text classifiers.

[29] 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.

[30] E. H. Spriggs, F. De La Torre, and M. Hebert. Temporal segmentation and activity classification from first-person sensing. *Computer Vision and Pattern Recognition Workshop*, 0:17–24, 2009.

[31] M.M. Stark and R.F. Riesenfeld. Wordnet: An electronic lexical database. In *Proceedings of 11th Eurographics Workshop on Rendering*. Citeseer, 1998.

[32] S. Stumpf, V. Rajaram, L. Li, M. Burnett, T. Dietterich, E. Sullivan, R. Drummond, and J. Herlocker. Toward harnessing user feedback for machine learning. Technical report, DTIC Document, 2006.

[33] DE Sturim, WM Campbell, ZN Karam, DA Reynolds, and FS Richardson. The MIT Lincoln Laboratory 2008 Speaker Recognition System. 2009.

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

[35] 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.

[36] C.A. Thompson, M.E. Califf, and R.J. Mooney. Active learning for natural language parsing and information extraction. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 406–414. Citeseer, 1999.

[37] 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.

[38] L. Von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

[39] M. Ware, E. Frank, G. Holmes, M. Hall, and I.H. Witten. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.

[40] E. Wolf and I. Gurevych. Aligning sense inventories in wikipedia and wordnet. In *Proceedings of the 1st Workshop on Automated Knowledge Base Construction*, pages 24–28, 2010.

[41] Y.W. Wong and R. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 960, 2007.

[42] J.M. Zelle and R.J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055, 1996.