Active Learning by Labeling Features

Gregory Druck
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
gdruck@cs.umass.edu

Burr Settles
Dept. of Biostatistics & Medical Informatics
Dept. of Computer Sciences
University of Wisconsin
Madison, WI 53706
bsettles@cs.wisc.edu

Andrew McCallum
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
mccallum@cs.umass.edu

Abstract

Methods that learn from prior information about input features such as *generalized expectation (GE)* have been used to train accurate models with very little effort. In this paper, we propose an *active learning* approach in which the machine solicits "labels" on *features* rather than instances. In both simulated and real user experiments on two sequence labeling tasks we show that our active learning method outperforms passive learning with features as well as traditional active learning with instances. Preliminary experiments suggest that novel interfaces which intelligently solicit labels on multiple features facilitate more efficient annotation.

1 Introduction

The application of machine learning to new problems is slowed by the need for labeled training data. When output variables are structured, annotation can be particularly difficult and time-consuming. For example, when training a conditional random field (Lafferty et al., 2001) to extract fields such as rent, contact, features, and utilities from apartment classifieds, labeling 22 instances (2,540 tokens) provides only 66.1% accuracy.[1]

Recent work has used unlabeled data and limited prior information about input features to bootstrap accurate structured output models. For example, both Haghighi and Klein (2006) and Mann and McCallum (2008) have demonstrated results better than 66.1% on the *apartments* task described above using only a list of 33 highly discriminative features and the labels they indicate. However, these methods have only been applied in scenarios in which the user supplies such prior knowledge before learning begins.

[1] Averaged over 10 randomly selected sets of 22 instances.

In traditional *active learning* (Settles, 2009), the machine queries the user for only the labels of instances that would be most helpful to the machine. This paper proposes an active learning approach in which the user provides "labels" for input features, rather than instances. A labeled input feature denotes that a particular input feature, for example the word *call*, is highly indicative of a particular label, such as contact. Table 1 provides an excerpt of a feature active learning session.

In this paper, we advocate using generalized expectation (GE) criteria (Mann and McCallum, 2008) for learning with labeled features. We provide an alternate treatment of the GE objective function used by Mann and McCallum (2008) and a novel speedup to the gradient computation. We then provide a pool-based feature active learning algorithm that includes an option to skip queries, for cases in which a feature has no clear label. We propose and evaluate feature query selection algorithms that aim to reduce model uncertainty, and compare to several baselines. We evaluate our method using both real and simulated user experiments on two sequence labeling tasks. Compared to previous approaches (Raghavan and Allan, 2007), our method can be used for both classification and structured tasks, and the feature query selection methods we propose perform better.

We use experiments with simulated labelers on real data to extensively compare feature query selection algorithms and evaluate on multiple random splits. To make these simulations more realistic, the effort required to perform different labeling actions is estimated from additional experiments with real users. The results show that active learning with features outperforms both passive learning with features and traditional active learning with instances.

In the user experiments, each annotator actively labels instances, actively labels features one at a time, and actively labels batches of features orga-

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 81–90,
Singapore, 6-7 August 2009.

accuracy 46.5 → 60.5	
feature	label
*PHONE**	contact
call	contact
deposit	rent
month	rent
pets	restrict.
lease	rent
appointment	contact
parking	features
*EMAIL**	contact
information	contact

accuracy 60.5 → 67.1	
feature	label
water	utilities
close	neighbor.
garbage	utilities
included	utilities
	features
shopping	neighbor.
bart	neighbor.
downtown	neighbor.
*TIME**	contact
bath	size

Table 1: Two iterations of feature active learning. Each table shows the features labeled, and the resulting change in accuracy. Note that the word *included* was labeled as both utilities and features, and that $*$ denotes a regular expression feature.

nized using a "grid" interface. The results support the findings of the simulated experiments and provide evidence that the "grid" interface can facilitate more efficient annotation.

2 Conditional Random Fields

In this section we describe the underlying probabilistic model for all methods in this paper. We focus on sequence labeling, though the described methods could be applied to other structured output or classification tasks. We model the probability of the label sequence $\mathbf{y} \in \mathcal{Y}^n$ conditioned on the input sequence $\mathbf{x} \in \mathcal{X}^n$, $p(\mathbf{y}|\mathbf{x};\theta)$ using first-order linear-chain conditional random fields (CRFs) (Lafferty et al., 2001). This probability is

$$p(\mathbf{y}|\mathbf{x};\theta) = \frac{1}{Z_\mathbf{x}} \exp\big(\sum_i \sum_j \theta_j f_j(y_i, y_{i+1}, \mathbf{x}, i)\big),$$

where $Z_\mathbf{x}$ is the partition function and feature functions f_j consider the entire input sequence and at most two consecutive output variables. The most probable output sequence and transition marginal distributions can be computed using variants of Viterbi and forward-backward.

Provided a training data distribution $\tilde{p}$, we estimate CRF parameters by maximizing the conditional log likelihood of the training data.

$$\mathcal{L}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{y}|\mathbf{x};\theta)]$$

We use numerical optimization to maximize $\mathcal{L}(\theta)$, which requires the gradient of $\mathcal{L}(\theta)$ with respect to the parameters. It can be shown that the partial derivative with respect to parameter j is equal to the difference between the empirical expectation of F_j and the model expectation of F_j, where $F_j(\mathbf{y},\mathbf{x}) = \sum_i f_j(y_i, y_{i+1}, \mathbf{x}, i)$.

$$\frac{\partial}{\partial\theta_j}\mathcal{L}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[F_j(\mathbf{y},\mathbf{x})] - E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y}|\mathbf{x};\theta)}[F_j(\mathbf{y},\mathbf{x})]].$$

We also include a zero-mean variance $\sigma^2 = 10$ Gaussian prior on parameters in all experiments.[2]

2.1 Learning with missing labels

The training set may contain partially labeled sequences. Let $\mathbf{z}$ denote missing labels. We estimate parameters with this data by maximizing the marginal log-likelihood of the observed labels.

$$\mathcal{L}_{MML}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log \sum_\mathbf{z} p(\mathbf{y},\mathbf{z}|\mathbf{x};\theta)]$$

We refer to this training method as *maximum marginal likelihood* (MML); it has also been explored by Quattoni et al. (2007).

The gradient of $\mathcal{L}_{MML}(\theta)$ can also be written as the difference of two expectations. The first is an expectation over the empirical distribution of $\mathbf{x}$ and $\mathbf{y}$, and the model distribution of $\mathbf{z}$. The second is a double expectation over the empirical distribution of $\mathbf{x}$ and the model distribution of $\mathbf{y}$ and $\mathbf{z}$.

$$\frac{\partial}{\partial\theta_j}\mathcal{L}_{MML}(\theta) = E_{\tilde{p}(\mathbf{x},\mathbf{y})}[E_{p(\mathbf{z}|\mathbf{y},\mathbf{x};\theta)}[F_j(\mathbf{y},\mathbf{z},\mathbf{x})]] - E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y},\mathbf{z}|\mathbf{x};\theta)}[F_j(\mathbf{y},\mathbf{z},\mathbf{x})]].$$

We train models using $\mathcal{L}_{MML}(\theta)$ with expected gradient (Salakhutdinov et al., 2003).

To additionally leverage unlabeled data, we compare with *entropy regularization* (ER). ER adds a term to the objective function that encourages confident predictions on unlabeled data. Training of linear-chain CRFs with ER is described by Jiao et al. (2006).

3 Generalized Expectation Criteria

In this section, we give a brief overview of *generalized expectation criteria* (GE) (Mann and McCallum, 2008; Druck et al., 2008) and explain how we can use GE to learn CRF parameters with estimates of feature expectations and unlabeled data.

GE criteria are terms in a parameter estimation objective function that express preferences on the

[2] 10 is a default value that works well in many settings.

value of a model expectation of some function. Given a score function S, an empirical distribution $\tilde{p}(\mathbf{x})$, a model distribution $p(\mathbf{y}|\mathbf{x};\theta)$, and a constraint function $G_k(\mathbf{x},\mathbf{y})$, the value of a GE criterion is $\mathcal{G}(\theta) = S(E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y}|\mathbf{x};\theta)}[G_k(\mathbf{x},\mathbf{y})]])$.

GE provides a flexible framework for parameter estimation because each of these elements can take an arbitrary form. The most important difference between GE and other parameter estimation methods is that it does not require a one-to-one correspondence between constraint functions G_k and model feature functions. We leverage this flexibility to estimate parameters of feature-rich CRFs with a very small set of expectation constraints.

Constraint functions G_k can be normalized so that the sum of the expectations of a set of functions is 1. In this case, S may measure the divergence between the expectation of the constraint function and a target expectation $\hat{G}_k$.

$$\mathcal{G}(\theta) = \hat{G}_k \log(E[G_k(\mathbf{x},\mathbf{y})]), \tag{1}$$

where $E[G_k(\mathbf{x},\mathbf{y})] = E_{\tilde{p}(\mathbf{x})}[E_{p(\mathbf{y}|\mathbf{x};\theta)}[G_k(\mathbf{x},\mathbf{y})]]$.

It can be shown that the partial derivative of $\mathcal{G}(\theta)$ with respect to parameter j is proportional to the predicted covariance between the model feature function F_j and the constraint function G_k.[3]

$$\frac{\partial}{\partial\theta_j}\mathcal{G}(\theta) = \frac{\hat{G}_k}{E[G_k(\mathbf{x},\mathbf{y})]}\times \tag{2}$$
$$\Big(E_{\tilde{p}(\mathbf{x})}\big[E_{p(\mathbf{y}|\mathbf{x};\theta)}[F_j(\mathbf{x},\mathbf{y})G_k(\mathbf{x},\mathbf{y})]$$
$$- E_{p(\mathbf{y}|\mathbf{x};\theta)}[F_j(\mathbf{x},\mathbf{y})]E_{p(\mathbf{y}|\mathbf{x};\theta)}[G_k(\mathbf{x},\mathbf{y})]\big]\Big)$$

The partial derivative shows that GE learns parameter values for model feature functions based on their predicted covariance with the constraint functions. GE can thus be interpreted as a bootstrapping method that uses the limited training signal to learn about parameters for related model feature functions.

3.1 Learning with feature-label distributions

Mann and McCallum (2008) apply GE to a linear-chain, first-order CRF. In this section we provide an alternate treatment that arrives at the same objective function from the general form described in the previous section.

Often, feature functions in a first-order linear-chain CRF f are binary, and are the conjunction of an observational test $q(\mathbf{x},i)$ and a label pair test $\mathbf{1}_{\{y_i=y',y_{i+1}=y''\}}$.[4]

$$f(y_i,y_{i+1},\mathbf{x},i) = \mathbf{1}_{\{y_i=y',y_{i+1}=y''\}}q(\mathbf{x},i)$$

The constraint functions G_k we use here decompose and operate similarly, except that they only include a test for a single label. Single label constraints are easier for users to estimate and make GE training more efficient. Label transition structure can be learned automatically from single label constraints through the covariance-based parameter update of Equation 2. For convenience, we can write G_{yk} to denote the constraint function that combines observation test k with a test for label y. We also add a normalization constant $C_k = E_{\tilde{p}(\mathbf{x})}[\sum_i q_k(\mathbf{x},i)]$,

$$G_{yk}(\mathbf{x},\mathbf{y}) = \sum_i \frac{1}{C_k}\mathbf{1}_{\{y_i=y\}}q_k(\mathbf{x},i)$$

Under this construction the expectation of G_{yk} is the predicted conditional probability that the label at some arbitrary position i is y when the observational test at i succeeds, $\tilde{p}(y_i{=}y|q_k(\mathbf{x},i){=}1;\theta)$.

If we have a set of constraint functions $\{G_{yk} : y \in \mathcal{Y}\}$, and we use the score function in Equation 1, then the GE objective function specifies the minimization of the KL divergence between the model and target distributions over labels conditioned on the success of the observational test. In general the objective function will consist of many such KL divergence penalties.

Computing the first term of the covariance in Equation 2 requires a marginal distribution over three labels, two of which will be consecutive, but the other of which could appear anywhere in the sequence. We can compute this marginal using the algorithm of Mann and McCallum (2008). As previously described, this algorithm is $O(n|\mathcal{Y}|^3)$ for a sequence of length n. However, we make the following novel observation: we do not need to compute the extra lattices for feature label pairs with $\hat{G}_{yk} = 0$, since this makes Equation 2 equal to zero. In Mann and McCallum (2008), probabilities were smoothed so that $\forall_y\ \hat{G}_{yk} > 0$. If we assume that only a small number of labels m have non-zero probability, then the time complexity of the gradient computation is $O(nm|\mathcal{Y}|^2)$. In this paper typically $1 \leq m \leq 4$, while $|\mathcal{Y}|$ is 11 or 13.

[3] If we use squared error for S, the partial derivative is the covariance multiplied by $2(\hat{G}_k - E[G_k(\mathbf{x},\mathbf{y})])$.

[4] We this notation for an indicator function that returns 1 if the condition in braces is satisfied, and 0 otherwise.

In experiments in this paper, using this optimization does not significantly affect final accuracy.

We use numerical optimization to estimate model parameters. In general GE objective functions are not convex. Consequently, we initialize 0th-order CRF parameters using a sliding window logistic regression model trained with GE. We also include a Gaussian prior on parameters with $\sigma^2 = 10$ in the objective function.

3.2 Learning with labeled features

The training procedure described above requires a set of observational tests or input features with target distributions over labels. Estimating a distribution could be a difficult task for an annotator. Consequently, we abstract away from specifying a distribution by allowing the user to assign labels to features (c.f. Haghighi and Klein (2006) , Druck et al. (2008)). For example, we say that the word feature *call* has label contact. A label for a feature simply indicates that the feature is a good indicator of the label. Note that features can have multiple labels, as does *included* in the active learning session shown in Table 1. We convert an input feature with a set of labels L into a distribution by assigning probability $1/|L|$ for each $l \in L$ and probability 0 for each $l \notin L$. By assigning 0 probability to labels $l \notin L$, we can use the speed-up described in the previous section.

3.3 Related Work

Other proposed learning methods use labeled features to label unlabeled data. The resulting partially-labeled corpus can be used to train a CRF by maximizing MML. Similarly, *prototype-driven learning* (PDL) (Haghighi and Klein, 2006) optimizes the joint marginal likelihood of data labeled with *prototype* input features for each label. Additional features that indicate similarity to the prototypes help the model to generalize. In a previous comparison between GE and PDL (Mann and McCallum, 2008), GE outperformed PDL without the extra similarity features, whose construction may be problem-specific. GE also performed better when supplied accurate label distributions.

Additionally, both MML and PDL do not naturally generalize to learning with features that have multiple labels or distributions over labels, as in these scenarios labeling the unlabeled data is not straightforward. In this paper, we attempt to address this problem using a simple heuristic: when there are multiple choices for a token's label, sample a label. In Section 5 we use this heuristic with MML, but in general obtain poor results.

Raghavan and Allan (2007) also propose several methods for learning with labeled features, but in a previous comparison GE gave better results (Druck et al., 2008). Additionally, the generalization of these methods to structured output spaces is not straightforward. Chang et al. (2007) present an algorithm for learning with constraints, but this method requires users to set weights by hand. We plan to explore the use of the recently developed related methods of Bellare et al. (2009), Graça et al. (2008), and Liang et al. (2009) in future work. Druck et al. (2008) provide a survey of other related methods for learning with labeled input features.

4 Active Learning by Labeling Features

Feature active learning, presented in Algorithm 1, is a *pool-based* active learning algorithm (Lewis and Gale, 1994) (with a pool of features rather than instances). The novel components of the algorithm are an option to skip a query and the notion that skipping and labeling have different costs. The option to skip is important when using feature queries because a user may not know how to label some features. In each iteration the model is retrained using the train procedure, which takes as input a set of labeled features $\mathcal{C}$ and unlabeled data distribution $\tilde{p}$. For the reasons described in Section 3.3, we advocate using GE for the train procedure. Then, while the iteration cost c is less than the maximum cost c_{max}, the feature query q that maximizes the query selection metric ϕ is selected. The accept function determines whether the labeler will label q. If q is labeled, it is added to the set of labeled features $\mathcal{C}$, and the label cost c_{label} is added to c. Otherwise, the skip cost c_{skip} is added to c. This process continues for N iterations.

4.1 Feature query selection methods

In this section we propose feature query selection methods ϕ. Queries with a higher scores are considered better candidates. Note again that by features we mean observational tests $q_k(\mathbf{x}, i)$. It is also important to note these are not *feature selection* methods since we are determining the features for which supervisory feedback will be most helpful to the model, rather than determining which features will be part of the model.

Algorithm 1 Feature Active Learning

Input: empirical distribution $\tilde{p}$, initial feature constraints $\mathcal{C}$, label cost c_{label}, skip cost c_{skip}, max cost per iteration c_{max}, max iterations N
Output: model parameters θ
for $i = 1$ **to** N **do**
  $\theta = \text{train}(\tilde{p}, \mathcal{C})$
  $c = 0$
  while $c < c_{max}$ **do**
    $q = \text{argmax}_{q_k}\, \phi(q_k)$
    if $\text{accept}(q)$ **then**
      $\mathcal{C} = \mathcal{C} \cup \text{label}(q)$
      $c = c + c_{label}$
    else
      $c = c + c_{skip}$
    end if
  end while
end for
$\theta = \text{train}(\tilde{p}, \mathcal{C})$

We propose to select queries that provide the largest reduction in model uncertainty. We notate possible responses to a query q_k as $\hat{g}$. The **Expected Information Gain (EIG)** of a query is the expectation of the reduction in model uncertainty over all possible responses. Mathematically, IG is

$$\phi_{EIG}(q_k) = E_{p(\hat{g}|q_k;\theta)}[E_{\tilde{p}(\mathbf{x})}[H(p(\mathbf{y}|\mathbf{x};\theta) - H(p(\mathbf{y}|\mathbf{x};\theta_{\hat{g}})]],$$

where $\theta_{\hat{g}}$ are the new model parameters if the response to q_k is $\hat{g}$. Unfortunately, this method is computationally intractable. Re-estimating $\theta_{\hat{g}}$ will typically involve retraining the model, and doing this for each possible query-response pair is prohibitively expensive for structured output models. Computing the expectation over possible responses is also difficult, as in this paper users may provide a set of labels for a query, and more generally $\hat{g}$ could be a distribution over labels.

Instead, we propose a tractable strategy for reducing model uncertainty, motivated by traditional uncertainty sampling (Lewis and Gale, 1994). We assume that when a user responds to a query, the reduction in uncertainty will be equal to the **Total Uncertainty (TU)**, the sum of the marginal entropies at the positions where the feature occurs.

$$\phi_{TU}(q_k) = \sum_i \sum_j q_k(\mathbf{x}_i, j) H(p(y_j|\mathbf{x}_i;\theta))$$

Total uncertainty, however, is highly biased towards selecting frequent features. A mean uncertainty variant, normalized by the feature's count, would tend to choose very infrequent features. Consequently we propose a tradeoff between the two extremes, called **weighted uncertainty (WU)**, that scales the mean uncertainty by the log count of the feature in the corpus.

$$\phi_{WU}(q_k) = \log(C_k)\frac{\phi_{TU}(q_k)}{C_k}.$$

Finally, we also suggest an uncertainty-based metric called **diverse uncertainty (DU)** that encourages diversity among queries by multiplying TU by the mean dissimilarity between the feature and previously labeled features. For sequence labeling tasks, we can measure the relatedness of features using distributional similarity.[5]

$$\phi_{DU}(q_k) = \phi_{TU}(q_k)\frac{1}{|\mathcal{C}|}\sum_{j\in\mathcal{C}} 1 - \text{sim}(q_k, q_j)$$

We contrast the notion of uncertainty described above with another type of uncertainty: the entropy of the predicted label distribution for the feature, or **expectation uncertainty (EU)**. As above we also multiply by the log feature count.

$$\phi_{EU}(q_k) = \log(C_k) H(\tilde{p}(y_i = y | q_k(\mathbf{x}, i) = 1; \theta))$$

EU is flawed because it will have a large value for non-discriminative features.

The methods described above require the model to be retrained between iterations. To verify that this is necessary, we compare against query selection methods that only consider the previously labeled features. First, we consider a feature query selection method called **coverage (cov)** that aims to select features that are dissimilar from existing labeled features, increasing the labeled features' "coverage" of the feature space. In order to compensate for choosing very infrequent features, we multiply by the log count of the feature.

$$\phi_{cov}(q_k) = \log(C_k)\frac{1}{|\mathcal{C}|}\sum_{j\in\mathcal{C}} 1 - \text{sim}(q_k, q_j)$$

Motivated by the feature query selection method of Tandem Learning (Raghavan and Allan, 2007) (see Section 4.2 for further discussion), we consider a feature selection metric **similarity (sim)** that is the maximum similarity to a labeled feature, weighted by the log count of the feature.

$$\phi_{sim}(q_k) = \log(C_k) \max_{j\in\mathcal{C}} \text{sim}(q_k, q_j)$$

[5] $\text{sim}(q_k, q_j)$ returns the cosine similarity between context vectors of words occurring in a window of ± 3.

Features similar to those already labeled are likely to be discriminative, and therefore likely to be labeled (rather than skipped). However, insufficient diversity may also result in an inaccurate model, suggesting that *coverage* should select more useful queries than *similarity*.

Finally, we compare with several passive baselines. **Random (rand)** assigns scores to features randomly. **Frequency (freq)** scores input features using their frequency in the training data.

$$\phi_{freq}(q_k) = \sum_i \sum_j q_k(\mathbf{x}_i, j)$$

Top LDA (LDA) selects the top words from 50 topics learned from the unlabeled data using latent Dirichlet allocation (LDA) (Blei et al., 2003). More specifically, the words w generated by each topic t are ranked using the conditional probability $p(w|t)$. The word feature is assigned its maximum rank across all topics.

$$\phi_{LDA}(q_k) = \max_t \ \mathrm{rank}_{\mathrm{LDA}}(q_k, t)$$

This method will select useful features if the topics discovered are relevant to the task. A similar heuristic was used by Druck et al. (2008).

4.2 Related Work

Tandem Learning (Raghavan and Allan, 2007) is an algorithm that combines feature and instance active learning for classification. The algorithm iteratively queries the user first for instance labels, then for feature labels. Feature queries are selected according to their co-occurrence with important model features and previously labeled features. As noted in Section 3.3, GE is preferable to the methods Tandem Learning uses to learn with labeled features. We address the mixing of feature and instance queries in Section 4.3.

In order to better understand differences in feature query selection methodology, we proposed a feature query selection method motivated[6] by the method used in Tandem Learning in Section 4.1. However, this method performs poorly in the experiments in Section 5.

Liang et al. (2009) simultaneously developed a method for learning with and actively selecting *measurements*, or target expectations with associated noise. The measurement selection method proposed by Liang et al. (2009) is based on Bayesian experimental design and is similar to the expected information gain method described above. Consequently this method is likely to be intractable for real applications. Note that Liang et al. (2009) only use this method in synthetic experiments, and instead use a method similar to total uncertainty for experiments in part-of-speech tagging. Unlike the experiments presented in this paper, Liang et al. (2009) conduct only simulated active learning experiments and do not consider skipping queries.

Sindhwani (Sindhwani et al., 2009) simultaneously developed an active learning method that queries for both instance and feature labels that are then used in a graph-based learning algorithm. They find that querying certain features outperforms querying uncertain features, but this is likely because their query selection method is similar to the expectation uncertainty method described above, and consequently non-discriminative features may be queried often (see also the discussion in Section 4.1). It is also not clear how this graph-based training method would generalize to structured output spaces.

4.3 Expectation Constraint Active Learning

Throughout this paper, we have focussed on labeling input features. However, the proposed methods generalize to queries for expectation estimates of arbitrary functions, for example queries for the label distributions for input features, labels for instances (using a function that is non-zero only for a particular instance), partial labels for instances, and class priors. The uncertainty-based query selection methods described in Section 4.1 apply naturally to these new query types. Importantly this framework would allow principled mixing of different query types, instead of alternating between them as in Tandem Learning (Raghavan and Allan, 2007). When mixing queries, it will be important to use different costs for different annotation types (Vijayanarasimhan and Grauman, 2008), and estimate the probability of obtaining a useful response to a query. We plan to pursue these directions in future work. This idea was also proposed by Liang et al. (2009), but no experiments with mixed active learning were presented.

[6]The query selection method of Raghavan and Allan (2007) requires a stack that is modified between queries within each iteration. Here query scores are only updated after each iteration of labeling.

5 Simulated User Experiments

In this section we experiment with an automated oracle labeler. When presented an instance query, the oracle simply provides the true labels. When presented a feature query, the oracle first decides whether to skip the query. We have found that users are more likely to label features that are relevant for only a few labels. Therefore, the oracle labels a feature if the entropy of its per occurrence label expectation, $\mathrm{H}(\tilde{p}(y_i = y|q_k(\mathbf{x}, i) = 1; \theta)) \leq 0.7$. The oracle then labels the feature using a heuristic: label the feature with the label whose expectation is highest, as well as any label whose expectation is at least half as large.

We estimate the effort of different labeling actions with preliminary experiments in which we observe users labeling data for ten minutes. Users took an average of 4 seconds to label a feature, 2 seconds to skip a feature, and 0.7 seconds to label a token. We setup experiments such that each iteration simulates one minute of labeling by setting $c_{max} = 60$, $c_{skip} = 2$ and $c_{label} = 4$. For instance active learning, we use Algorithm 1 but without the skip option, and set $c_{label} = 0.7$. We use $N = 10$ iterations, so the entire experiment simulates 10 minutes of annotation time. For efficiency, we consider the 500 most frequent unlabeled features in each iteration. To start, ten randomly selected seed labeled features are provided.

We use **random (rand)** selection, **uncertainty sampling (US)** (using sequence entropy, normalized by sequence length) and **information density (ID)** (Settles and Craven, 2008) to select instance queries. We use Entropy Regularization (ER) (Jiao et al., 2006) to leverage unlabeled instances.[7] We weight the ER term by choosing the best[8] weight in $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$ multiplied by $\frac{\#labeled}{\#unlabeled}$ for each data set and query selection method. Seed instances are provided such that the simulated labeling time is equivalent to labeling 10 features.

We evaluate on two sequence labeling tasks. The *apartments* task involves segmenting 300 apartment classified ads into 11 fields including features, rent, neighborhood, and contact. We use the same feature processing as Haghighi and Klein (2006), with the addition of context features in a window of ± 3. The *cora references* task is to extract 13 BibTeX fields such as author and booktitle from 500 research paper references. We use a standard set of word, regular expressions, and lexicon features, as well as context features in a window of ± 3. All results are averaged over ten random 80:20 splits of the data.

[7] Results using self-training instead of ER are similar.

[8] As measured by test accuracy, giving ER an advantage.

method	apartments		cora	
	mean	**final**	**mean**	**final**
ER rand	48.1	53.6	75.9	81.1
ER US	51.7	57.9	76.0	83.2
ER ID	51.4	56.9	75.9	83.1
MML rand	47.7	51.2	58.6	64.6
MML WU	57.6	60.8	61.0	66.2
GE rand	59.0	64.8*	77.6	83.7
GE freq	66.5*	71.6*	68.6	79.8
GE LDA	65.7*	71.4*	74.9	85.0
GE cov	68.2*†	72.6*	73.5	83.3
GE sim	57.8	65.9*	67.1	79.2
GE EU	66.5*	71.6*	68.6	79.8
GE TU	70.1*†	**73.6***†	76.9	**88.2***†
GE WU	**71.6***†	**74.6***†	**80.3***†	**88.1***†
GE DU	70.5*†	**74.4***†	78.4*	**87.5***†

Table 2: Mean and final token accuracy results. A * or † denotes that a GE method significantly outperforms all non-GE or passive GE methods, respectively. Bold entries significantly outperform all others. Methods in *italics* are passive.

5.1 Results

Table 2 presents mean (across all iterations) and final token accuracy results. On the *apartments* task, GE methods greatly outperform MML[9] and ER methods. Each uncertainty-based GE method also outperforms all passive GE methods. On the *cora* task, only GE with *weighted uncertainty* significantly outperforms ER and passive GE methods in terms of *mean* accuracy, but all uncertainty-based GE methods provide higher final accuracy. This suggests that on the *cora* task, active GE methods are performing better in later iterations. Figure 1, which compares the learning curves of the best performing methods of each type, shows this phenomenon. Further analysis reveals that the uncertainty-based methods are choosing frequent features that are more likely to be skipped than those selected randomly in early iterations.

We next compare with the results of related methods published elsewhere. We cannot make claims about statistical significance, but the results

[9] Only the best MML results are shown.

illustrate the competitiveness of our method. The 74.6% final accuracy on *apartments* is higher than any result obtained by Haghighi and Klein (2006) (the highest is 74.1%), higher than the *supervised* HMM results reported by Grenager et al. (2005) (74.4%), and matches the results of Mann and McCallum (2008) with GE with more accurate sampled label distributions and 10 labeled examples. Chang et al. (2007) only obtain better results than 88.2% on *cora* when using 300 labeled examples (two hours of estimated annotation time), 5000 additional unlabeled examples, and extra test time inference constraints. Note that obtaining these results required only 10 simulated minutes of annotation time, and that GE methods are provided no information about the label transition matrix.

6 User Experiments

Another advantage of feature queries is that feature names are concise enough to be browsed, rather than considered individually. This allows the design of improved interfaces that can further increase the speed of feature active learning. We built a prototype interface that allows the user to quickly browse many candidate features. The features are split into groups of five features each. Each group contains features that are related, as measured by distributional similarity. The features within each group are sorted according to the active learning metric. This interface, displayed in Figure 3, may be useful because features in the same group are likely to have the same label.

We conduct three types of experiments. First, a user labels instances selected by *information density*, and models are trained using ER. The instance labeling interface allows the user to label tokens quickly by extending the current selection one token at a time and only requiring a single keystroke to label an entire segment. Second, the user labels features presented one-at-a-time by *weighted uncertainty*, and models are trained using GE. To aid the user in understanding the function of the feature quickly, we provide several examples of the feature occurring in context and the model's current predicted label distribution for the feature. Finally, the user labels features organized using the grid interface described in the previous paragraph. *Weighted uncertainty* is used to sort feature queries within each group, and GE is used to train models. Each iteration of labeling lasts two minutes, and there are five iterations. Retraining with ER between iterations takes an average of 5 minutes on *cora* and 3 minutes on *apartments*. With GE, the retraining times are on average 6 minutes on *cora* and 4 minutes on *apartments*. Consequently, even when viewed with *total time*, rather than *annotation time*, feature active learning is beneficial. While waiting for models to retrain, users can perform other tasks.

Figure 2 displays the results. User 1 labeled *apartments* data, while Users 2 and 3 labeled *cora* data. User 1 was able to obtain much better results with feature labeling than with instance labeling, but performed slightly worse with the grid interface than with the serial interface. User 1 commented that they found the label definitions for *apartments* to be imprecise, so the other experiments were conducted on the *cora* data. User 2 obtained better results with feature labeling than instance labeling, and obtained higher mean accuracy with the grid interface. User 3 was much better at labeling features than instances, and performed especially well using the grid interface.

7 Conclusion

We proposed an active learning approach in which features, rather than instances, are labeled. We presented an algorithm for active learning with features and several feature query selection methods that approximate the expected reduction in model uncertainty of a feature query. In simulated experiments, active learning with features outperformed passive learning with features, and uncertainty-based feature query selection outperformed other baseline methods. In both simulated and real user experiments, active learning with features outperformed passive and active learning with instances. Finally, we proposed a new labeling interface that leverages the conciseness of feature queries. User experiments suggested that this grid interface can improve labeling efficiency.

Acknowledgments

We thank Kedar Bellare for helpful discussions and Gaurav Chandalia for providing code. This work was supported in part by the Center for Intelligent Information Retrieval and the Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249. The second author was supported by a grant from National Human Genome Research Institute. Any opinions, findings and conclusions or recommendations are the authors' and do not necessarily reflect those of the sponsor.

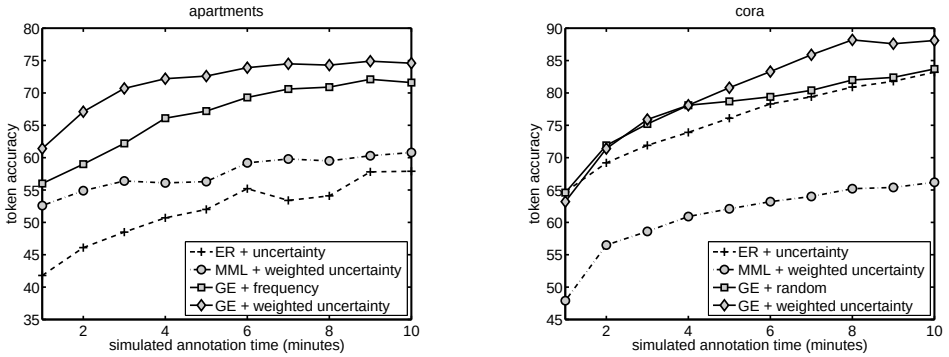

Figure 1: Token accuracy vs. time for best performing ER, MML, passive GE, and active GE methods.

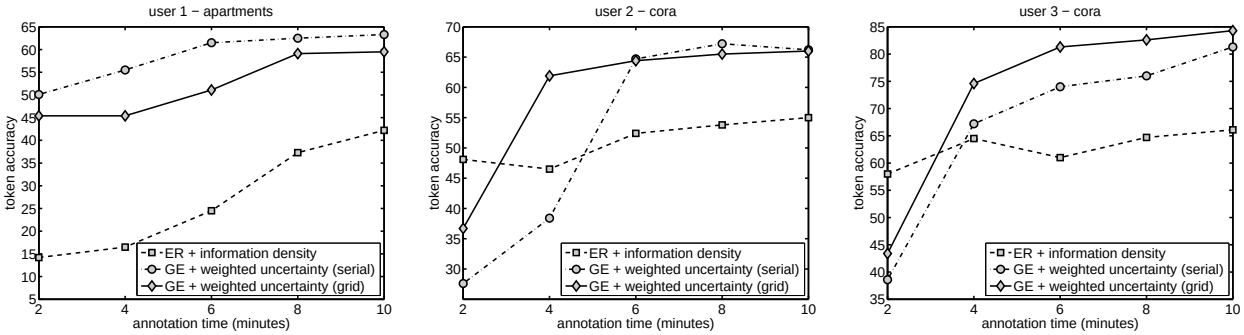

Figure 2: User experiments with instance labeling and feature labeling with the *serial* and *grid* interfaces.

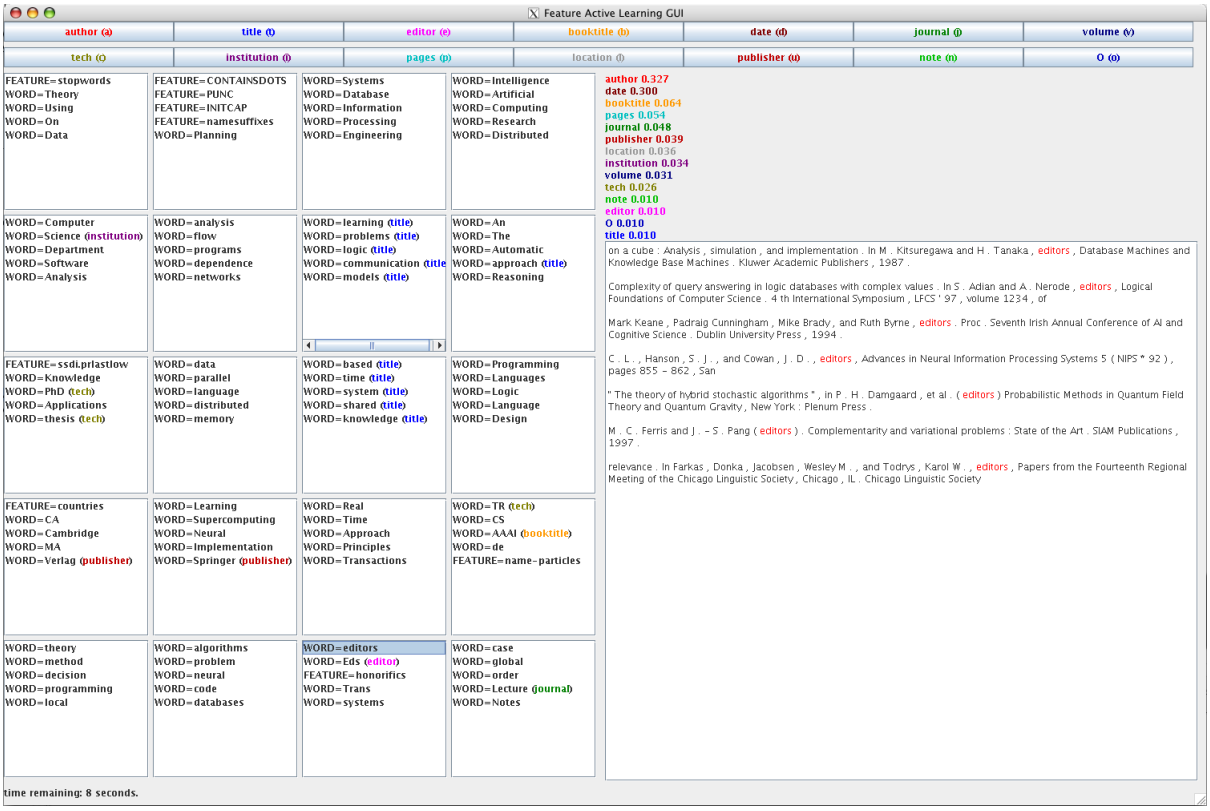

Figure 3: Grid feature labeling interface. Boxes on the left contain groups of features that appear in similar contexts. Features in the same group often receive the same label. On the right, the model's current expectation and occurrences of the selected feature in context are displayed.

References

Kedar Bellare, Gregory Druck, and Andrew McCallum. 2009. Alternating projections for learning with expectation constraints. In *UAI*.

David M. Blei, Andrew Y. Ng, Michael I. Jordan, and John Lafferty. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:2003.

Ming-Wei Chang, Lev Ratinov, and Dan Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *ACL*, pages 280–287.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *SIGIR*.

Joao Graça, Kuzman Ganchev, and Ben Taskar. 2008. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*. MIT Press.

Trond Grenager, Dan Klein, and Christopher D. Manning. 2005. Unsupervised learning of field segmentation models for information extraction. In *ACL*.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *HTL-NAACL*.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. 2006. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *ACL*, pages 209–216.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *SIGIR*, pages 3–12, New York, NY, USA. Springer-Verlag New York, Inc.

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *ICML*.

Gideon Mann and Andrew McCallum. 2008. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *ACL*.

A. Quattoni, S. Wang, L.-P Morency, M. Collins, and T. Darrell. 2007. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1848–1852, October.

Hema Raghavan and James Allan. 2007. An interactive algorithm for asking and incorporating feature feedback into support vector machines. In *SIGIR*, pages 79–86.

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. 2003. Optimization with em and expectation-conjugate-gradient. In *ICML*, pages 672–679.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *EMNLP*.

Burr Settles. 2009. Active learning literature survey. Technical Report 1648, University of Wisconsin - Madison.

Vikas Sindhwani, Prem Melville, and Richard D. Lawrence. 2009. Uncertainty sampling and transductive experimental design for active dual supervision. In *ICML*.

Sudheendra Vijayanarasimhan and Kristen Grauman. 2008. Multi-level active prediction of useful image annotations for recognition. In *NIPS*.