Reducing Labeling Effort for Structured Prediction Tasks

Aron Culotta and **Andrew McCallum**
Department of Computer Science
140 Governor's Drive
University of Massachusetts
Amherst, MA 01003–4601
{culotta, mccallum}@cs.umass.edu

Abstract

A common obstacle preventing the rapid deployment of supervised machine learning algorithms is the lack of labeled training data. This is particularly expensive to obtain for structured prediction tasks, where each training instance may have multiple, interacting labels, all of which must be correctly annotated for the instance to be of use to the learner. Traditional active learning addresses this problem by optimizing the order in which the examples are labeled to increase learning efficiency. However, this approach does not consider the *difficulty* of labeling each example, which can vary widely in structured prediction tasks. For example, the labeling predicted by a partially trained system may be easier to correct for some instances than for others.

We propose a new active learning paradigm which reduces not only *how many* instances the annotator must label, but also *how difficult* each instance is to annotate. The system also leverages information from partially correct predictions to efficiently solicit annotations from the user. We validate this active learning framework in an interactive information extraction system, reducing the total number of annotation actions by 22%.

Introduction

Supervised machine learning algorithms require a set of fully labeled training examples for accurate and robust performance. Unfortunately, for many tasks, this labeled data is costly and time-consuming to obtain.

Active learning is a framework that aims to reduce this burden, typically by optimizing the order in which the examples are labeled (Cohn, Ghahramani, & Jordan 1995; Lewis & Catlett 1994). For instance, one might order the examples such that those with the least confident predictions are labeled first. By seeing the most *valuable* examples early in training, the algorithm can learn more efficiently.

Most active learners are evaluated by plotting a "learning curve" that displays the learner's performance on a held-out data set as the number of labeled examples increases. An active learner is considered successful if it obtains better performance than a traditional learner given the same number of labeled examples. Thus, active learning expedites annotation by reducing the number of labeled examples required to train an accurate model.

However, this paradigm assumes each example is equally difficult to annotate. While this assumption may hold in traditional classification tasks, in structured classification tasks it does not. For example, consider an information extraction system that labels segments of free text with tags corresponding to entities of interest. An annotated example might look like the following:

```
        <name> Jane Smith </name>
           <title> CEO </title>
    <company> Unicorp, LLC </company>
 Phone: <phone> (555)555-5555 </phone>
```

To label this example, the user must not only specify which type of entity each token is, but also must determine the start and end boundaries for each entity. Clearly, the amount of work required to label an example such as this will vary between examples, based on the number of entities. However, this effort is not reflected by the standard evaluation metrics from active learning. Since our goal is to reduce annotation effort, it is desirable to design a labeling framework that considers not only *how many* instances the annotator must label, but also *how difficult* each instance is to annotate.

Additionally, unlike in traditional classification tasks, a structured prediction system may be able to *partially* label an example, which can simplify annotation. In the above example, the partially-trained system might correctly segment the title field, but mislabel it as a company name. We would like to leverage these partial predictions to reduce labeling effort.

We propose a framework to address these shortcomings for machine learning applied to information extraction. We first provide a way to quantify the number of actions a user must perform to label each training example, distinguishing between boundary and classification annotations. We then demonstrate an interactive information extraction system that minimizes the amount of effort required to train an accurate extractor.

To expedite annotation for information extraction (IE), we first note that the main difference between labeling IE examples and labeling traditional classification examples is the problem of boundary annotation (or *segmentation*). Given a sequence of text that is correctly segmented, choosing the correct type for each entity is simply a classification task: the annotator must choose among a finite set of labels for each entity. However, determining the boundaries of each entity is an intrinsically distinct task, since the number of ways to segment a sequence is exponential in the sequence length. Additionally, from a human-computer interaction perspective, the "clicking and dragging" involved in boundary annotation generally requires more hand-eye coordination from the user than does classification annotation.

With this distinction in mind, our system reduces annotation effort in two ways. First, many segmentation decisions are converted into classification decisions by presenting the user with multiple predicted segmentations to choose from. Thus, instead of hand segmenting each field, the user may select the correct segmentation from the given choices.

Second, the system allows the user to correct partially labeled examples, and then constrains its predictions to respect these corrections. This interaction further reduces the number of segmentation decisions the user must make: Corrections to one part of the sequence often propagate to fix segmentation errors in other parts of the sequence.

The resulting system allows the user to constrain the predictions of the learner without manually annotating the boundaries of incorrect segments. Very often, these constraints will allow the user to simply select the correct annotation from among the provided choices. Thus, the annotator can frequently label a record without explicitly annotating the boundaries.

We demonstrate the performance of this framework in the domain of contact record extraction. The task of the annotator is to train a system that can accurately extract contact information (such as names and addresses) from unstructured text. In particular, the model we use is a linear-chain conditional random field (CRF) (Lafferty, McCallum, & Pereira 2001). The probabilistic foundations of CRFs make them well-suited to the confidence estimation and correction propagation methods required by our framework.

We present results demonstrating that our framework reduces the total number of annotation actions required to train an IE system by 22%, and furthermore that it reduces the number of boundary annotations by 46%, as compared with competing methods.

By reducing the effort required to train an extractor, this work can lead to more wide-spread acceptance of end-user information extraction systems that incorporate machine learning techniques.

Related Work

To the best of our knowledge, this is the first active learning framework that (1) is sensitive to the *difficulty* of labeling each training example and (2) uses partially labeled instances to reduce this labeling difficulty.

Part of our framework can be viewed as a type of *selective sampling* (Cohn, Atlas, & Ladner 1994), which proposes an order in which to label the training instances such that learning is most efficient. In particular, ours is a *certainty-based* method in that it prefers to label instances for which the system has low confidence in its predictions (Lewis & Catlett 1994). Our work, however, incorporates user feedback to more efficiently solicit annotated examples.

Methods for computing confidence estimates in natural language tasks have been studied in domains such as text classification (Gandrabur & Foster 2003), information extraction (Scheffer, Decomain, & Wrobel 2001; Culotta & McCallum 2004), and speech recognition (Gunawardana, Hon, & Jiang 1998), although none of these consider labeling difficulty in their confidence estimates.

Thompson, Califf, & Mooney (1999) present an active learning system for information extraction and parsing, which are instances of structured learning tasks. While they demonstrate the advantage of active learning for these tasks, they require the annotator to fully label each training instance, which is precisely what this paper aims to avoid.

Others have studied efficient ways to interactively train an extraction system (Cardie & Pierce 1998; Caruana, Hodor, & Rosenberg 2000); however, these methods do not use partially labeled instances to reduce labeling effort. Partially correct annotations are marked as incorrect.

This work can be viewed as an active learning extension to Kristjannson *et al.* (2004), which presents a framework for *interactive information extraction* and describes the details of *correction propagation* and *confidence estimation* for CRFs. A CRF for contact record extraction is fully trained and used to automatically populate a contact record database. The interactive framework provides a minimal-effort way to iteratively correct system errors until the predicted database is error-free. However, that work requires that all corrections be manually provided by the user, including segmentation decisions (with the exception of those corrections enabled by *correction propagation*). Therefore, it is not sensitive the amount of effort the user must invest to correct each example. This paper presents a way to leverage correction propagation in an active learning setting to directly reduce the number of segmentation labels the user must provide, as well as a way to exploit multiple system predictions to reduce overall labeling effort.

Additionally, Kristjannson *et al.* (2004) propose the *Expected Number of User Actions* (ENUA) measure to estimate the labeling effort to correctly enter all fields of a record. This measure, however, does not address the distinction between boundary and classification labels. In particular, ENUA assumes it takes one action to segment and label an entity. In this paper, we present measures that account for the effort required for each of these actions.

The main contributions of this paper are (1) a new active learning framework that incorporates the difficulty of labeling each example, (2) a method to convert segmentation labeling into classification labeling using partially correct annotations, (3) a more detailed estimate of the number of annotation actions required to label each example, and (4) a mechanism for performing correction propagation when corrections are given across multiple system predictions.

Annotation framework

We first provide a brief overview of the annotation framework applied to IE. Given an IE learning algorithm L and a set of unlabeled data U, the task is to iteratively solicit annotations from the user and retrain the extractor.

At iteration t, the system operates as follows:

1. Rank each unlabeled instance by its confidence value given by the current extractor L^t.
2. Select the least confident example $u \in U$ to be labeled.
3. Present the user the top k labelings of u predicted by L^t.
4. If the correct labeling exists in the top k choices, allow the user to select that labeling, and add u to the labeled data set.
5. Otherwise, for *any* entity in these k predictions that is *segmented* correctly but *classified* incorrectly, allow the user to provide the correct type for this entity.
6. Based on these corrections, generate a new set of k predictions, propagating these corrections to possibly fix other errors.
7. If the correct labeling exists in the top k choices, allow the user to select that labeling and add u to the labeled dataset.
8. Otherwise, if the correct labeling still does not exist in these k predictions, allow the user to manually correct one of these incorrect k predictions with the true labeling.

Notice that the only step in which the user must manually segment entities is step 8. Steps 4 and 7 allow the user to label the sequence by making a choice among k predictions. Step 5 allows the user to provide correct entity types to the learner, without manually segmenting fields. In step 6, the system performs constrained inference to generate a new set of predictions that conform to the user's corrections. It is in this step that the system often automatically corrects segmentation errors present in the first k choices.

This framework allows the user to rapidly and easily annotate examples and correct the system's predictions, while reducing the amount of effort spent labeling boundaries.

In the remaining sections, we describe in more detail the components of this system. As some of these details are dependent on the learner being used, we first briefly describe CRFs, which we will use in our experiments.

Conditional Random Fields

The machine learning method we apply is a conditional random field (CRF) (Lafferty, McCallum, & Pereira 2001), a model successfully used in information extraction for tasks such as named entity recognition. CRFs are undirected graphical models that encode the conditional probability of a set of output variables $\mathbf{Y}$ given a set of evidence variables $\mathbf{X}$. The set of distributions expressible by a CRF is specified by an undirected graph $\mathcal{G}$, where each vertex corresponds to a random variable. If $C = \{\{\mathbf{y}_c, \mathbf{x}_c\}\}$ is the set of cliques determined by the edges in $\mathcal{G}$, then the conditional probability of $\mathbf{y}$ given $\mathbf{x}$ is

$$p_\Lambda(\mathbf{y}|\mathbf{x}) = \frac{1}{Z_\mathbf{x}} \prod_{c \in C} \phi_c(\mathbf{y}_c, \mathbf{x}_c; \Lambda)$$

where ϕ is a potential function parameterized by Λ and $Z_x = \sum_\mathbf{y} \prod_{c \in C} \phi(\mathbf{y}_c, \mathbf{x}_c)$ is a normalization factor. We assume ϕ_c factorizes as a log-linear combination of arbitrary features computed over clique c, therefore

$$\phi_c(\mathbf{y}_c, \mathbf{x}_c; \Lambda) = \exp\left(\sum_k \lambda_k f_k(\mathbf{y}_c, \mathbf{x}_c)\right)$$

The model parameters $\Lambda = \{\lambda_k\}$ are a set of real-valued weights typically learned from labeled training data by maximum likelihood estimation.

In the special case in which the designated output nodes of the graphical model are linked by edges in a *linear chain*, CRFs make a first-order Markov independence assumption among output nodes, and thus correspond to finite state machines (FSMs). In this case CRFs can be roughly understood as conditionally-trained hidden Markov models, with additional flexibility to effectively take advantage of complex overlapping features.

Confidence estimation

A common form of active learning is certainty-based selective sampling (Lewis & Catlett 1994), which gives higher priority to unlabeled examples for which the learner has a low confidence in its predictions. Culotta & McCallum (2004) describe the *constrained forward-backward* algorithm to estimate the confidence of CRF predictions. This algorithm calculates the probability that an entity (or an entire sequence) has a particular labeling, which follows directly from the semantics of undirected models: The probability of the hidden states corresponding to an entity's labeling is the marginal probability of those hidden states given the observed input. We refer the reader to Culotta & McCallum (2004) for more details on this algorithm.

Using this method, we can assign a confidence estimate to each unlabeled training example. By labeling the least confident examples first, we can increase the CRF learning rate.

Selecting top predictions

To present the user with the top k predictions, we must extend the CRF inference algorithm to return k predictions, instead of simply the top prediction. For linear-chain CRFs, inference is performed using an analog of the Viterbi algorithm, a dynamic program well-known for its use in inference in hidden Markov models (Rabiner 1989). There are also well-established, efficient modifications to the Viterbi algorithm that can calculate the top k optimal predictions, often called *n-best Viterbi* (Schwartz & Chow 1990). This algorithm can be viewed as a beam search through the space of possible predictions. We apply this algorithm to inference in CRFs to generate the k most probable predictions.

Correction propagation

In step 5, the annotator provides the true type for entities that have been correctly segmented but incorrectly classified. The system must then produce the top k predictions that conform to these new annotations.

Kristjannson *et al.* (2004) present the *constrained Viterbi* algorithm, which modifies the traditional Viterbi algorithm to prune from the search space those labelings that do not agree with the given annotations.

The interesting capability of this algorithm is that by constraining the predicted label for one entity, the prediction for another entity may change as well. As a simple example, consider labeling the name “Charles Stanley” with the fields FIRSTNAME and LASTNAME. If the system confuses the first and last names, a naïve correction system will require two corrective actions. Using *constrained Viterbi*, when the user corrects the FIRSTNAME field to be “Stanley,” the system automatically changes the LASTNAME field to “Charles.” Kristjannson *et al.* (2004) call this capability *correction propagation*.

We extend this to our current task using an algorithm we call *n-best constrained Viterbi*, which, as its name suggests, combines *n-best Viterbi* with *constrained Viterbi*. This extension can be straight-forwardly implemented by constraining the *n-best Viterbi* algorithm to prune predictions that do not agree with the annotations.

Using this algorithm, we enable the system to solicit corrections for the *classification* of entities, which are then propagated to correct both the *classification* and *segmentation* of other entities. In this way, we can reduce the amount of effort expended on segmentation labeling.

Measuring annotation effort

To calculate the amount of effort required to label a training example, we wish to abstract from the details of a particular user interface, and instead quantify atomic user actions. In the case of IE annotation, we define three atomic labeling actions: START, END, and TYPE, corresponding to labeling the start boundary, end boundary, and type of an entity.

Thus, labeling the input

```
<name> Jane Smith </name>
  <title> CEO </title>
```

requires 2 START, 2 END, and 2 TYPE actions. The goal of our annotation framework is to reduce the total number of annotation actions.

We can see that a partially labeled example can require fewer annotation actions. For example, consider the following partially labeled record:

```
<name> Jane </name> Smith
 <company> CEO </company>
```

This requires one END action to fix the ending boundary of “Jane,” and one TYPE action to change “CEO” from a company to a title. Thus, using the partial labeling has reduced the total number of required actions from 6 to 2.

By presenting the user with k predictions, we introduce another action: If one of the k predictions is correct, the user must choose this prediction. We call this action CHOICE.

When simulating corrections from the annotator, we accumulate the number of times each action is performed. In the first round, when the user corrects the types of correctly segmented fields, the only action incurred is the TYPE action. If none of the k constrained predictions are correct, then (and only then) the user must perform the segmentation actions START and END.

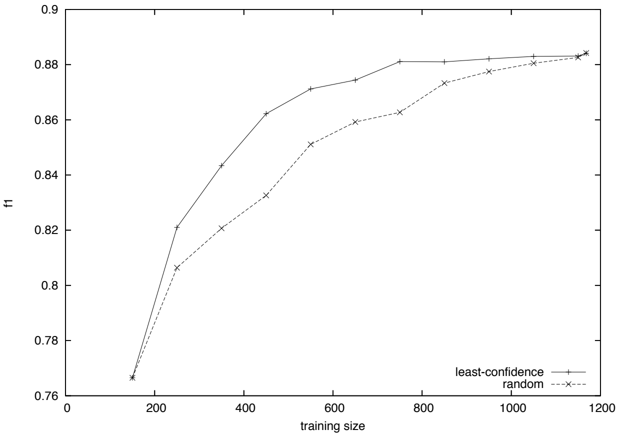

Figure 1: Testing label F1 as a function of training set size. LEASTCONFIDENCE labels the least confident instances first, while RANDOM labels the instances in a random order.

It will generally be the case that some actions are more expensive than others. For example, as mentioned earlier, START and END actions may require more hand-eye coordination than TYPE actions. A cost-sensitive approach could take this into account; however, in these experiments, we assume each atomic action has unit cost.

Experiments

Using the fully annotated collection of extracted contact records from Kristjannson *et al.* (2004), we simulate our annotation framework and measure the performance of the CRF with respect to the number of actions required to train it.

For training and testing 2187 contact records (27,560 words) were collected from web pages and e-mails and 25 classes of entities were hand-labeled.[1] Some data came from pages containing lists of addresses, and about half came from disparate web pages found by searching for valid pairs of city name and zip code.

The features used in the CRF consisted of capitalization features, 24 regular expressions over the token text (e.g. CONSTAINSHYPHEN), and offsets of these features within a window of size 5. We also used 19 lexicons, including “US Last Names,” “US First Names,” “State names,” “Titles/Suffixes,” “Job titles,” and “Road endings.” Feature induction was not used in these experiments.

We use 150 examples to train an initial CRF, 1018 to simulate user annotation, and 1019 to evaluate performance.

We first show that traditional active learning is beneficial in this domain. Figure 1 plots the average label F1 versus training size where the order in which instances are

[1] The 25 fields are: FIRSTNAME, MIDDLENAME, LASTNAME, NICKNAME, SUFFIX, TITLE, JOBTITLE, COMPANYNAME, DEPARTMENT, ADDRESSLINE, CITY1, CITY2, STATE, COUNTRY, POSTALCODE, HOMEPHONE, FAX, COMPANYPHONE, DIRECTCOMPANYPHONE, MOBILE, PAGER, VOICEMAIL, URL, EMAIL, INSTANTMESSAGE

	START + END	TYPE	CHOICE	START + END + TYPE	TOTAL
BASELINE	1345	999	**519**	2444	2963
$k = 1$	1342	1078	639	2420	3059
$k = 2$	977	916	719	1893	2612
$k = 3$	827	870	748	1697	2445
$k = 4$	**722**	**837**	766	**1559**	**2325**

Table 1: Number of actions to label 1018 examples. By converting segmentation actions into classification actions, we can reduce the total number of annotation actions by 22%.

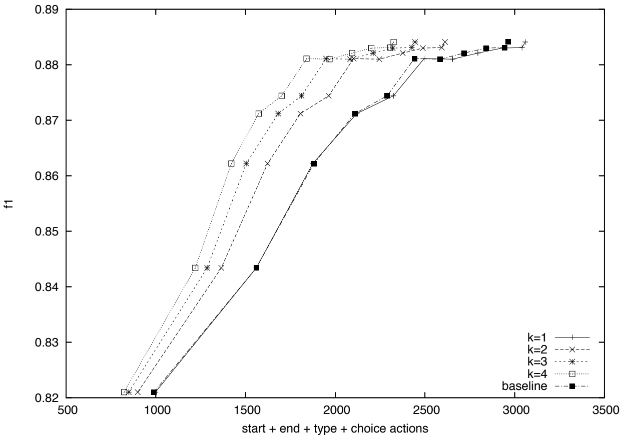

Figure 2: Testing label F1 as a function of the total number of annotation actions. At $k = 4$, performance plateaus with roughly 800 fewer actions than the baseline.

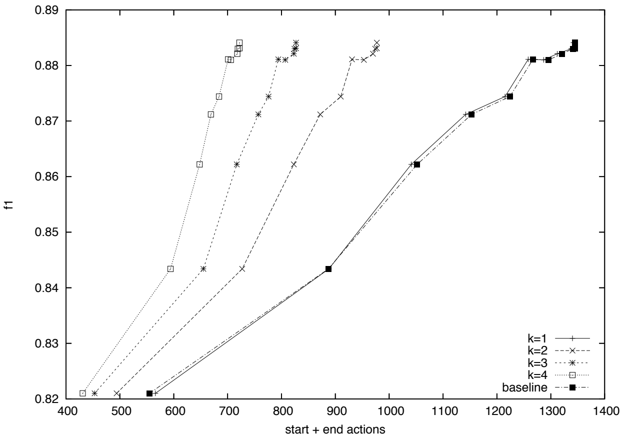

Figure 3: Testing label F1 as a function of the total number of segmentation actions. The interactive system with $k = 4$ requires just over half the number of segmentation actions of the baseline.

labeled is either random (RANDOM) or by order of least-confidence (LEASTCONFIDENCE). Note that here each labeled instance must be manually labeled by the annotator. This figure demonstrates that the order in which examples are labeled can affect learning efficiency.

However, we desire a more explicit measure of labeling effort, so we examine how F1 varies with the number of annotation actions. The next set of experiments all label the training examples in order of least-confidence. We compare two competing methods. BASELINE presents the user with the top prediction, and the user must hand annotate all corrections. The other method is the learning framework advocated in this paper, which presents the user with k possible segmentation, and interactively solicits label corrections. We vary k from 1 to 4. Constrained inference and correction propagation are used in one round of interactive labeling.

Figure 2 compares these methods, measuring the total number of annotation actions required for each F1 value. The interactive framework outperforms the baseline consistently. On average, interactive labeling with $k = 4$ requires 22% fewer actions than BASELINE.

Note that $k = 1$ closely tracks the BASELINE performance. This suggests that when we restrict the user corrections to TYPE errors only, there are not enough errors corrected by correction propagation to overcome the additional cost of a round of user interaction. This is further confirmed by the fact that performance increases with k.

To demonstrate the reduction in segmentation labeling, Figure 3 displays the number of segmentation actions (START or END) needed to achieve a particular F1 value. On average across the sampled F1 values, interactive labeling with $k = 4$ requires 42% fewer segmentation actions.

Note the steep learning curve of the interactive method. This suggests that the CRF's poor segmentation performance early in training is quickly overcome. The result is that after a small number of actions, annotator can reduce the number of boundary labels needed to train the CRF, and instead mostly provide TYPE annotation.

Table 1 displays the total number of actions required to label all the unlabeled data. Note that BASELINE incurs a CHOICE action if the correct labeling is the top choice.

The results in Table 1 agree with the trends in Figures 2 and 3. Note that the increase in CHOICE actions is expected, since there are many instances where the correct labeling is in the top k choices. The advantage of this framework is that the cost incurred by these CHOICE actions are outweighed by the reduction in other actions that they enable. Note also that this reduction in effort is manifest even assuming all actions incur the same cost. If we assume that boundary annotation is more costly than TYPE annotation, these difference will be even more pronounced.

Discussion

It is invariably difficult to simulate the effort of a user's interaction with a system; ultimately we would like to perform user studies to measure labeling time exactly. While the proposed metrics make few assumptions about the user interface, there are certainly some costs we have not considered. For example, the current metrics do not explicitly account for the time required to read a labeling. However, the action CHOICE, which is incremented whenever a user picks the correct labeling among the top k predictions, can be seen to encompass this action. Placing a higher cost on CHOICE can account for the reading effort, possibly altering the optimal value of k. Indeed, picking the best value of k can be achieved by first choosing a relative cost for reading, then performing simulations.

Also note that this work can be seen as a way to facilitate the wide-spread use of machine learning information extraction algorithms. End-user machine learning systems often require additional training examples to personalize the system to the user's data (for example, Apple Inc.'s trainable junk mail filter). The easier it is for an end-user to train a system, the more likely it is that the system will be well-received, frequently used, and be given enough training data to provide high accuracy performance. This "learning in the wild" capability can lead to end-users more rapidly adopting learning technologies to perform information extraction on their own data.

Conclusions

We have described an active learning framework that explicitly models the effort required to label each example, and have demonstrated that it can reduce the total number of annotation actions to train an information extraction system by 22%.

From these results, we can conclude that methods aiming to reduce labeling effort can benefit from considering not only how many examples an annotator must label, but also how much effort is required to label each example.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by U.S. Government contract #NBCH040171 through a subcontract with BBNT Solutions LLC, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under contract number NBCHD030010. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsor.

References

Cardie, C., and Pierce, D. 1998. Proposal for an interactive environment for information extraction. Technical Report TR98-1702, Cornell University.

Caruana, R.; Hodor, P.; and Rosenberg, J. 2000. High precision information extraction. In *KDD-2000 Workshop on Text Mining*.

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

Cohn, D. A.; Ghahramani, Z.; and Jordan, M. I. 1995. Active learning with statistical models. In Tesauro, G.; Touretzky, D.; and Leen, T., eds., *Advances in Neural Information Processing Systems*, volume 7, 705–712. The MIT Press.

Culotta, A., and McCallum, A. 2004. Confidence estimation for information extraction. In *Human Langauge Technology Conference (HLT 2004)*.

Gandrabur, S., and Foster, G. 2003. Confidence estimation for text prediction. In *Proceedings of the Conference on Natural Language Learning (CoNLL 2003)*.

Gunawardana, A.; Hon, H.; and Jiang, L. 1998. Word-based acoustic confidence measures for large-vocabulary speech recognition. In *Proc. ICSLP-98*, 791–794.

Kristjannson, T.; Culotta, A.; Viola, P.; and McCallum, A. 2004. Interactive information extraction with conditional random fields. *Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, 282–289. Morgan Kaufmann, San Francisco, CA.

Lewis, D. D., and Catlett, J. 1994. Heterogeneous uncertainty sampling for supervised learning. In Cohen, W. W., and Hirsh, H., eds., *Proceedings of ICML-94, 11th International Conference on Machine Learning*, 148–156. New Brunswick, US: Morgan Kaufmann Publishers, San Francisco, US.

Rabiner, L. 1989. A tutorial on hidden Markov models. In *IEEE*, volume 77, 257–286.

Scheffer, T.; Decomain, C.; and Wrobel, S. 2001. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001*.

Schwartz, R., and Chow, Y.-L. 1990. The n-best algorithms: an efficient and exact procedure for finding the n most likely sentence hypotheses. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP-90)*.

Thompson, C. A.; Califf, M. E.; and Mooney, R. J. 1999. Active learning for natural language parsing and information extraction. In *Proc. 16th International Conf. on Machine Learning*, 406–414. Morgan Kaufmann, San Francisco, CA.