Active Learning for Natural Language Processing

Literature Review

Shilpa Arora and Sachin Agarwal

Masters Students
shilpaa@cs.cmu.edu , sachina@cs.cmu.edu

Language Technologies Institute

School of Computer Science
Carnegie Mellon University

Supervised by

Noah A. Smith

Assistant Professor

LTI, MLD

School of Computer Science

Carnegie Mellon University

Contents

List of Figures

1 Introduction to Active Learning

Machine Learning methods usually require supervised data to learn a concept. Labeling data is time consuming, tedious, error prone and expensive. The research community has looked at semi-supervised (Chapelle et al., 2006) and unsupervised learning leachniques in order to obviate the need of labeled data to a certain extent. In addition to the above mentioned problems with labeled data, all examples are not equally informative or equally easy to label. For instance, the examples similar to what the learner has already seen are not as useful as new examples. Moreover, different examples may require different amount of user's labeling effort, for instance, a longer sentence is likely to have more ambiguities and hence would be harder to parse manually. Active learning is the task of reducing the amount of labeled data required to learn the target concept by querying the user for labels for the most informative examples so that the concept is learnt with fewer examples.

An active learning problem setting typically consists of a small set of labeled examples and a large set of unlabeled examples. An initial classifier is trained on the labeled examples and/or the unlabeled examples. From the pool of unlabeled examples, *selective sampling* is used to create a small subset of examples for the user to label. This iterative process of training, selective sampling and annotation is repeated until convergence.

In this literature review, we present the research done in active learning applied to natural language processing (NLP). Active learning has been applied to two types of problems in NLP, classification tasks such as text classification (McCallum and Nigam, 1998) or structured prediction task such as named entity recogonition (Shen et al., 2004), semantic role labeling (Roth and Small, 2006), and parsing (Hwa, 2000). The main focus of this literature survey is on structured prediction. The orgranization of the literature review is as follows: we start by discussing the different evaluation meassures used for active learning in NLP (section 2). In section 3, we discuss the various selective sampling approaches for active learning. Broadly these can be classified as uncertainty-based sampling and query-by committee. We also discuss some other approaches used based on diversity and representativeness of an example. Section 6 concludes and discusses some open issues in active learning for NLP.

2 Evaluation Measures for Active Learning

An active learning experiment is usually described by five properties: number of bootstrap examples, batch size, supervised learner, data set and a stopping criterion. The supervised learner is trained on the bootstrap examples which are labeled by the user initially. Batch size is the number of examples that are selectively sampled from the unlabeled pool and added to the training pool in each iteration. The stopping criterion can be either a desired performance level or the number of iterations. Performance is evaluated on the test set in each iteration.

Active learners are usually evaluated by plotting a *learning curve* of performance vs. number of labeled examples as shown in figure 1. Success of an active learner is demonstrated by showing that it achieves better performance than a traditional learner given the same number of labeled examples; i.e., for achieving the desired performance, the active learner needs fewer examples than the traditional learner.

To measure the actual effort a user puts in labeling the data he is presented, it is very important to define an appropriate performance measure for the annotation effort. This is especially important for structured outputs, such as named entities, semantic roles, parsers etc., as each example is not equally difficult to annotate and examples might require different number of user actions (e.g. clicks). Kristjannson et al. (2004) introduce an efficiency measure called Expected Number of User Actions (ENUA) defined as the

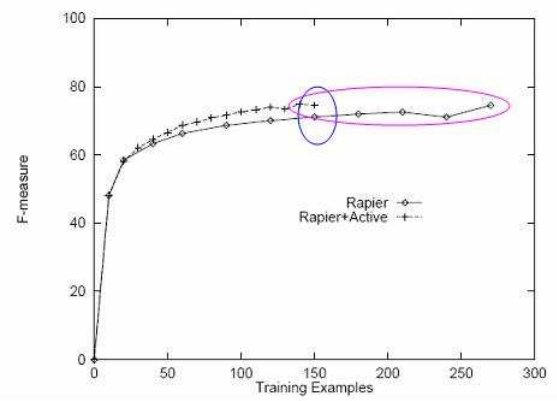


Figure 1: Active learning curve: f-measure with 150 examples and active learning is same as the f-measure with 270 random examples. This represents a saving of 120 examples, or 44% (Thompson et al., 1999).

number of user actions, such as clicks, required to correctly label all the fields. They use this measure for evaluting user's effort on a sequence prediction task.

There can be many ways to calculate ENUA. Kristjannson et al. (2004) express ENUA in terms of $P_i(j)$ - probability distribution over the number of errors j after i manual corrections. For the case when user is presented with the result of automatic field assignment and has to correct all the errors, ENUA would be defined as:

$$ENUA = \sum_{n=0}^{\infty} nP_0(n) \tag{1}$$

where $P_0(n)$ is the distribution over the number of incorrect fields. This can be estimated from the number of incorrect fields in a record as shown in figure 2

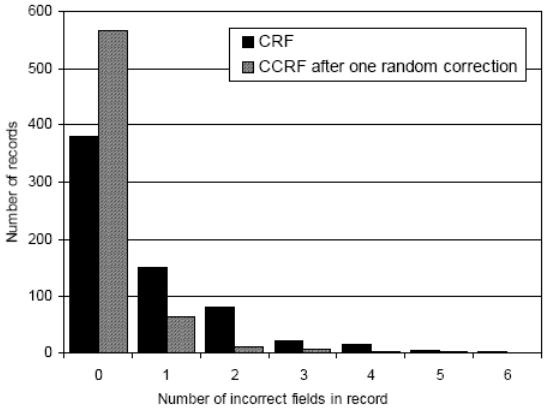


Figure 2: Histogram where records fall into bins depending on the number of incorrect fields in a record. Solid bar - CRF with no correction, shaded bar - CRF with one random correction. These can be used to estimate $P_0(n)$ and $P_1(n)$ (Kristjannson et al., 2004).

For another case when the model has an initial automatic field assignment, followed by a single manual correction by the user, the ENUA is defined as:

$$ENUA_1 = (1 - P_0(0)) + \sum_{n} nP_1(n) \tag{2}$$

where $P_0(0)$ is the probability that all fields are correctly assigned initially and $P_1(n)$

is the distribution over the number of incorrect fields in a record after one field has been corrected. The distribution P_1 will depend on which incorrect field is corrected, e.g. a random incorrect field or the least confident incorrect field. Thus, this measure can be used to compare selective sampling strategies for active learning.

The ENUA (Expected Number of User Action) measure does not distinguish between boundary detection and classification. Segmenting and labeling an entity is considered as one single task.

Culotta and McCallum (2005) define four types of user actions: START and END for identifying the span of the annotation, TYPE for labeling an annotation and CHOICE for choosing the correct prediction from the top k predictions recommended by the system. They assume that each atomic action has a unit cost; however START and END actions may require more hand-eye coordination.

The measures for user effort defined in literature do not take into account the effort involved in reading the text (including the surrounding context) for labeling. There is a need for some formal user studies to identify and accurately quantify the user effort involved in the annotation task.

ENUA is a more direct measure of reduction in human effort. However, not much work in literature has reported results in ENUA.

3 Active Learning by Selective Sampling

Active learning aims at reducing the number of examples required to achieve the desired accuracy by selectively sampling the examples for user to label and train the classifier with. Several different strategies for selective sampling have been explored in the literature. In this review, we present some of the selective sampling techniques used for active learning in NLP.

3.1 Uncertainty-based sampling

Uncentainty-based sampling selects examples that the model is least certain about and presents them to the user for correction/verification. A lot of work on active learning has used uncertainty-based sampling. In this section, we describe some of this work.

3.1.1 Parsing and rule-based Information Extraction

Thompson et al. (1999) applied active learning techniques to two natural language learning systems CHILL (Zelle and Mooney, 1996) and RAPIER (Califf, 1998). CHILL uses inductive logic programming (ILP)(Muggleton and Raedt, 1994) to learn a parser that maps sentences to their semantic representation. The labeled data in form of sentences paired with their meaning representation is used to learn rules for parsing. They used the CHILL system with a geographical database interface to learn a parser that maps natural language questions into Prolog queries which can be used to retrieve an answer from the database. RAPIER is a rule-based information extraction system that uses a sequence of patterns to identify relevant phrases in the document. The task is to extract information about computer-related job postings on the web to develop a jobs database.

In CHILL, the parser may get stuck and not complete the parse. If a sentence cannot be parsed, the model is definitely uncertain about it and it's a good candidate for showing to the user. From the list of sentences that cannot be parsed, most uncertain examples are selected for presenting to the user. Uncertainty is calculated as the maximum number of sequential operators/rules successfully applied while attempting to parse the sentence and dividing it by the number of words in the sentence to give an estimate of how close the parser came to completing a parse. Sentences with a lower value for this metric are

selected for annotation. For a sentence that can be parsed completely by the model, the certainty is calculated using the certainty of the rules applied to parse it. Certainty of the rule is estimated as the number of examples that induce the rule in the training data.

The certainty of a slot in RAPIER's IE task is calculated from the certainty of the rules that fill it. In the case where a single rule fills a slot, the certainty for the slot is simply the certainty of the rule that filled it; otherwise, the certainty of the slot is defined as the minimum of the certainties of the rules that produced these fillers. The certainty of an individual extraction rule is based on its coverage of the training data: $pos - 5 * neg$, where pos is the number of correct fillers generated by the rule and neg is the number of incorrect ones. Rules that account for fewer examples are viewed as less certain.

Experiments and Results: To evaluate parser acquisition in CHILL, they used the corpus containing 250 questions about U.S. geography and their corresponding prolog queries. To calculate performance, the answers retrieved from queries generated by the model and answers from gold labeled queries were compared and percentage of correct answers was used as the performance measure. They used 25 bootstrap examples (initial training data) and in each iteration, 25 examples were selectively sampled and added to the training pool. The results are shown in figure 3. With active learning, the CHILL system requires 29% fewer training examples to reach within 5% of the final accuracy levels.

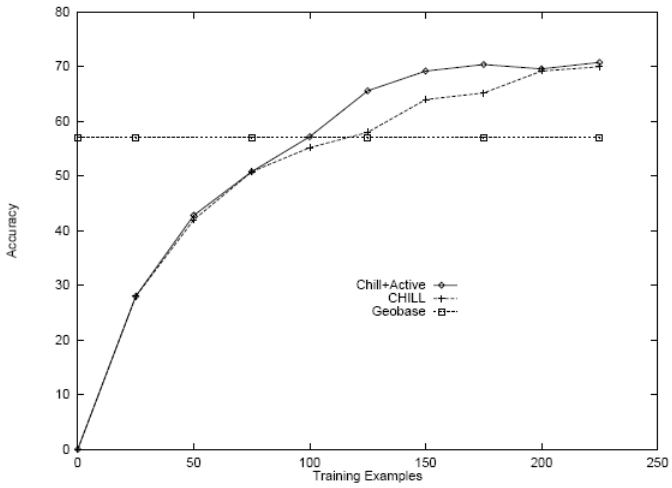


Figure 3: Parser acquisition results for Geography Corpus (CHILL system) (Thompson et al., 1999)

To evaluate active learning in RAPIER, they used the computer-related job-posting corpus consisting of 300 postings to the local newsgroup *austin.jobs*. With 10 bootstrap examples and selectively sampling one example at a time from 260 examples, F-measure performance at 150 examples (with active learning) was same as that with 270 examples (without active learning). This represents a savings of 44%.

3.1.2 Application to Interactive Information Extraction

Kristjannson et al. (2004) apply active learning to an interactive system with an interface for user to make corrections and view results of active learning applied to Information Extraction. They present their work on the task of extracting contact addresses from on-line sources such as email messages or web pages. A contact address consists of more than 20 fields such as last name, first name, address, city, state, phone etc. State of the art

automatic systems for this task have low error rate (about 10%). The goal of an interactive information extraction system is to provide an interface for the user to make corrections to the errors made by the trained model, with minimal effort. The fields extracted in this task are interdependent. Given a name "Charles Stanley", if the system makes a mistake and classifies 'Stanley' as the first name and 'Charles' as the last name, a naïve correction system would require two corrections from the user, whereas an interactive interface would require only one correction and would automatically correct the other one. The authors call this capability *correction propagation*.

They model the task as a sequence labeling task with begin, inside and outside (BIO) labels for each field such as $B - FirstName$, $I - LastName$ etc. Several statistical approaches have been used for information extraction. Any such technique can be used in this paradigm as long as it gives confidence for its predictions. Maximum entropy is one of the popular techniques used information extraction, since it allows use of arbitrary overlapping features. However, maximum entropy estimates each field independently and hence potential for correction propagation is minimal. Conditional Random Fields (CRFs) are well suited for the interactive information task since the confidence of fields can be estimated and it provides a natural framework for optimally propagating user corrections.

Conditional Random fields (CRFs) (Lafferty et al., 2001) are discriminative probabilistic models used for labeling sequence data. Like Markov Random Fields, CRFs are undirected graphical models that define a log-linear distribution over label sequences given a particular observation sequence. Let $o =< o_1, o_2, ..., o_T >$ be the observed sequence of words in a document, and $s =< s_1, s_2, ..., s_T >$ be the hidden label sequence (such as the label lastname). CRFs define the probability of a state sequence given the observation as:

$$p(s \mid o) = \frac{1}{Z_o}(\sum_{t=1}^{T} \sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t)) \tag{3}$$

where Z_o is the normalization factor over all state sequences for a given observation sequence o. $f_k(s_{t-1}, s_t, o, t)$ is an arbitrary feature function and λ_k is the learned weight vector for k^{th} feature function. Belief propagation using dynamic programming can be used to calculate Z_o efficiently. Maximum aposteriori training is performed efficiently using hill-climbing methods such as conjugate gradient or limited memory BFGS (Sha and Pereira, 2003). Most likely sequence is calculated using dynamic programming method similar to Viterbi's algorithm. For Hidden Markov Models (HMMs), Viterbi (Rabiner, 1989) algorithm is used to find the most likely state sequence to have generated the observation sequence (sequence that maximizes the joint probability). Since CRFs are conditional model, Viterbi algorithm for CRFs instead gives us the most likely state sequence given the observation sequence:

$$s^* = \arg\max_{s} p(s \mid o) \tag{4}$$

Viterbi algorithm stores the most likely path (through the state sequence) at time t, which accounts for the first t observations, and ending in state s_i. The probability of this path is given as:

$$\delta_{t+1}(s_i) = \max_{s'}[\delta_t(s')exp(\sum_{k} \lambda_k f_k(s_{t-1}, s_t, o, t))] \tag{5}$$

where $\delta_0(s_i)$ is the probability of starting in state s_i. The probability of the most likely state sequence is given as:

$$p^* = \arg\max_{i}[\delta_T(s_i)] \tag{6}$$

The most likely state sequence can be recovered by back tracking through the dynamic programming table.

To support the user interaction model, some of the hidden variables are clamped to the corrections provided by the user. These values are called constraints and the inference algorithm is called constrained Viterbi algorithm for CRFs. In constrained Viterbi algorithm, the path is constrained to pass through some sub-path $C =< s_t, s_{t+1}... >$. Based on these constraints, the probability of the path at time $(t+1)$ ending in state i is given as:

$$\delta_{t+1}(s_i) = \begin{cases} \max_{s'}[\delta_t(s')exp(\sum_k \lambda_k f_k(s_{t-1}, s_t, o, t)] & \text{if } s_i = s_{t+1} \\ 0 & otherwise \end{cases} \quad (7)$$

The user may correct the whole instance (i.e. all the fields - first name, last name, address etc.) or only one field. The user's corrections are immediately propagated using the constrained Viterbi algorithm. In addition to correcting the field, the constrained Viterbi algorithm may also change the predicted states for other variables because with the recursive formulation in Viterbi, the constraints can affect the optimal paths before and after the time steps specified in C. This is known as *correction propagation*. An interesting question here is whether to propagate corrections after single manual annotation or after a batch of annotations.

To reduce the human effort, examples that would benefit the learner the most should be presented to the user. Since, the fields the learner is most uncertain about, would provide it most information, examples with least confidence from the model are presented to the user. For confidence estimation of the extracted field, constrained Forward-Backward algorithm (Culotta and McCallum, 2004) is used. Constrained Forward-Backward algorithm calculates the probability of any sequence passing through a set of constraints where constraints are the assignment of labels to the extracted fields identified by the model.

The positive constraints correspond to labels inside the identified fields and the negative constraints correspond to the boundary of the field. For example, if labels B-JOBTITLE and I-JOBTITLE are used to label tokens that begin and continue a JOBTITLE field, and a given sequence $< o_2, ..., o_5 >$ is labeled as JOBTITLE, then the constraint corresponds to $s_2 = B - JOBTITLE, s_3 = ... = s_5 = I - JOBTITLE, s_6 \neq I - JOBTITLE$. The assignment with the least confidence score is recommended to the user for verification/ correction.

Experiments and Results: Kristjannson et al. (2004) used a collection of 2187 records from web pages and emails, hand-labeled with 25 classes of data fields. The baseline uses the learned model to populate the fields and the user corrects all the errors (Model 1). Model 2 asks the user to correct one random error and correction propagation is used to correct the other errors. This reduces the ENUA by 13.9% i.e. the user has to correct 13.9% lesser examples compared to the baseline with no correction propagation. Model 3 asks the user to correct the least confident field followed by correction propagation. This model reduces the ENUA by 11.3%. Interestingly, correcting a random field is slightly more informative than correcting the least confident field which could be surprising at first. However, at a closer look, a field will have low confidence if the posterior probability of the competing classes is close to the score for the chosen class. Thus, it requires a small amount of extra information to boost the posterior of one of the other classes and change the classification. On the other hand, by getting the correction from the user for a randomly selected field with higher confidence which is incorrect, the classifier would gain more knowledge from the user and hence would be more suitable for correction propagation.

So, given that the field selected for showing to the user is incorrect, randomly selected

examples are more useful than least confident examples. However, it is also important to consider how many randomly selected or least confident fields were actually incorrect. Kristjannson et al. (2004) found that a least confident field was truly incorrect 81.9% of the times whereas randomly chosen fields are incorrect only 29.0% of the time. Thus, confidence estimates are more effective in directing the user to incorrect fields.

3.1.3 Local margin vs. Global margin, Complete labels vs. Partial labels

Roth and Small (2006) present a margin-based method for active learning in structured outputs such as semantic roles, where the interdependencies between output variables are described as a general set of constraints that can be used to represent any structural form. Automatic semantic role labeling (Carreras and Màrques, 2004), also called shallow semantic parsing, is the task of identifying and labeling the semantic arguments of a predicate. This helps us in extracting information such as *who* did *what* to *whom*, *when* and *how*, from the sentence. The semantic arguments for a predicate have two categories, core arguments and adjunctive arguments. Core arguments are labeled ARG0 to ARG5, where ARG0 is generally the prototypical agent and ARG1 is the prototypical patient. No generalization is made for higher numbered arguments. Examples of Adjunctive arguments (ARG-Ms) are ARG-LOC for location, ARG-TMP for time etc. In semantic role labeling, the structural constraints could be that no two semantic roles for a single verb can overlap or other linguistic constraints that yield a restricted output space. The paper presents results for active learning techniques applied to semantic role labeling.

Roth and Small (2006) model the semantic role labeling problem with input X as a set of natural language features and output Y as the position and type of a semantic-role in the sentence. To learn semantic roles, one can either learn a set of local functions such as " phrase is an argument of *run*", or a global classifier to predict all semantic-roles at once.

Given an assignment $\mathrm{x} \in \mathcal{X}^{n_x}$ to a collection of input variables, $X = (X_1, ..., X_{n_x})$ $X_t \in \mathbb{R}^{d_t}$, the structured classification problem involves identifying the *best* assignment $\mathrm{y} \in \mathcal{C}(\mathcal{Y}^{n_y})$ to a collection of output variables $Y = (Y_1, ..., Y_{n_y}); Y_t \in \{w_1, ..., w_{k_t}\}$ that are consistent with a defined structure on Y. The structure can be thought of as constraining the output space to a smaller space $\mathcal{C}(\mathcal{Y}^{n_y}) \subseteq \mathcal{Y}^{n_y}$, where $\mathcal{C} : 2^{\mathcal{Y}^{n_y}} \rightarrow 2^{\mathcal{Y}^{n_y}}$ constraints the output space to be structurally consistent.

The learning algorithm takes as input m training instances, $S = \{(\mathrm{x}_1, \mathrm{y}_1), ..., (\mathrm{x}_m, \mathrm{y}_m)\}$ drawn *i.i.d.* over $\mathcal{X}$ x $\mathcal{C}(\mathcal{Y}^{n_y})$ and returns a classifier $h : \mathcal{X}^{n_x} \rightarrow \mathcal{C}(\mathcal{Y}^{n_y})$. Let $f : \mathcal{X}^{n_x}$ x $\mathcal{Y}^{n_y} \rightarrow \mathbb{R}$ be the global scoring function such that, given an instance x, the resulting classification is given by

$$\hat{\mathrm{y}}_{\mathcal{C}} = h(\mathrm{x}) = \underset{\mathrm{y}' \in \mathcal{C}(\mathcal{Y}^{n_y})}{argmax} f(\mathrm{x}, \mathrm{y}') \tag{8}$$

Global scoring function might be decomposable into local scoring functions $f_{y_t}(\mathrm{x}, \mathrm{y})$ such that $f(\mathrm{x}, \mathrm{y}) = f(\mathrm{x}, (y_1, ...y_{n_y})) = \sum_{t=1}^{n_y} f_{y_t}(\mathrm{x}, t)$. If it is, then with a set of local classifiers, each of which outputs a local distribution $\hat{y}_t$, and an optimization procedure (combined with constraints) we can generate a global prediction $\hat{\mathrm{y}}_c$.

Given a classifier $f \in H$, an instance can be locally learnable/separable, globally learnable/separable or *exclusively* globally learnable/separable. In the locally learnable case, the identity function is a sufficient 'optimization procedure' to learn the global predictions i.e. structural constraints are not necessary to learn the desired function. A classifier $f \in H$, locally separates the data D, if for all examples $(\mathrm{x}, \mathrm{y}) \in D$, $f_{y_t}(\mathrm{x}, t) > f_{y'}(\mathrm{x}, t)$ for all $y' \in \mathcal{Y} \setminus y_t$, and all $\mathrm{y}'' \in \mathcal{Y}^{n_y} \setminus \mathrm{y}$. Classifier f globally separates the data D, if for all examples $(\mathrm{x}, \mathrm{y}) \in D$, $f(\mathrm{x}, \mathrm{y}) > f(\mathrm{x}, \mathrm{y}')$ for all $\mathrm{y}' \in \mathcal{Y}^{n_y} \setminus \mathrm{y}$.

Instances that are globally learnable but not locally learnable are called *exclusively* globally learnable. In the *exclusively* globally learnable case, the local predictions will

be incorrect, but the global prediction following optimization will be correct (once learning is complete). Local learnability implies global learnability, but not *exclusively* global learnability since we require constraints to learn the required function (the local classifiers alone are not sufficient). For a locally learnable instance, $\hat{y}_{\mathcal{C}} = \hat{y} = y$ and for an *exclusively* globally learnable instance, $\hat{y}_{\mathcal{C}} \neq \hat{y} = y$.

For margin-based classifiers, uncertainty translates to distance from the hyper-plane. Examples with the smallest margin and thereby least certainty are presented for user's input. For the multi-class classification, the examples with minimum difference of margin for the predicted label and other labels are selected. The multi-class margin can be defined as:

$$\rho_{multiclass}(x, y, f) = f(x, y) - f(x, \dot{y}) \tag{9}$$

where y represents the true label and $\dot{y}$ represents the highest activation value such that $\dot{y} \neq y$.

Roth and Small (2006) explore the tradeoff between selecting instances based on a global margin or a combination of the margin of local classifiers. Let $\mathcal{Q}$ be a querying function which given unlabeled data $\mathcal{S}_u$ and the learner, it returns a set of unlabeled examples $\mathcal{S}_{select} \subseteq \mathcal{S}_u$. Using the global margin as defined in the above equation, the querying function $\mathcal{Q}$ can be written as:

$$\mathcal{Q}_{global} : x^* = \underset{x \in S_u}{argmin}[f(x, \hat{y}_{\mathcal{C}}) - f(x, \tilde{y}_{\mathcal{C}})] \tag{10}$$

where $\hat{y_{\mathcal{C}}}$ represents the predicted label(under constraints) and $\tilde{y}_{\mathcal{C}}$ represents the second highest activation value(under constraints). Here, the scoring function $f(x, y)$ doesn't need to be decomposable into local scoring functions. However, for many structured learning problems, it's possible to decompose the global learning function into local classification problems. Also, local classification problems have lower sample complexity. A querying function for optimizing local predictions would be to select examples with a small average local multi-class margin defined as,

$$Q_{local(\mathcal{C})} : x^* = \underset{x \in S_u}{argmin} \frac{\sum_{t=1}^{n_y} [f_{\hat{y_{\mathcal{C},t}}}(x, t) - f_{\tilde{y}_{\mathcal{C},t}}(x, t)]}{n_y} \tag{11}$$

where $\hat{y_{\mathcal{C},t}}$ represents the local predicted label consistent with the global constraints and $\tilde{y}_{\mathcal{C},t}$ represents the second highest local prediction consistent with the global constraints.

So far, the user was given the complete structure to label. However, in querying for complete labels, we are hindered by cases where for an instance, one output variable is very informative but other output variables associated with the same instance are minimally useful. Thus, it could be useful to query only very informative partial labels. The querying function for partial labels is defined as:

$$Q_{local(\mathcal{C})} : (x, t)^* = \underset{(x, y_t) \in S_u}{argmin}[f_{\hat{y_{\mathcal{C},t}}}(x, t) - f_{\tilde{y}_{\mathcal{C},t}}(x, t)] \tag{12}$$

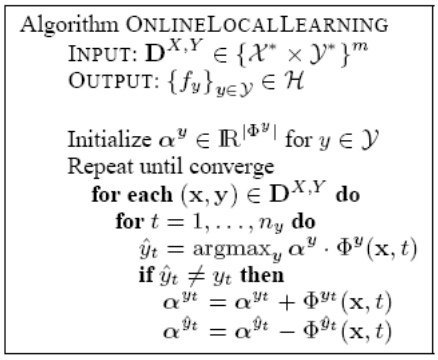
In using the partial labels, a notion similar to correction propagation (Culotta et al., 2006), explained in section 3.1.2 can be applied here. In addition to the global constraints, the partial labels would reduce the output space size for remaining local variables.

This work uses classifiers with linear representation where parameters are learned using the perceptron algorithm. The linear local classifier is associated with linear functions, $f_y(x, t) = \alpha^y \cdot \phi^y(x, t)$, where $\alpha^y \in \mathbb{R}^{d_y}$ is the learned weight vector and $\phi^y(x, t) \in \mathbb{R}^{d_y}$ is the feature vector for local classification. The global scoring function, $f(x, y) = \alpha \cdot \phi(x, y)$ where $\alpha = (\alpha^1, ..., \alpha^{|\mathcal{Y}|})$ is a concatenation of the local α^y vectors and $\phi(x, y) = (\phi^1(x, y), ..., \phi^{|Y|}(x, y))$ is a concatenation of the local feature vectors, $\phi^y(x, y)$. Where,

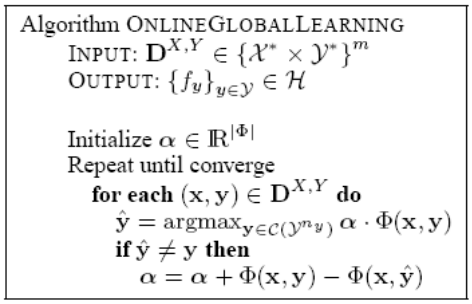
$\phi^y(\mathrm{x},\mathrm{y}) = \sum_{t=1}^{n_y} \phi^{y_t}(\mathrm{x},t) I_{\{y_t=y\}}$ is an accumulation over all output variables of features occurring for class y.

They used inference based training (IBT) proposed by (Punyakanok et al., 2005) for learning. The IBT algorithms as described by (Punyakanok et al., 2005) for online local and global learning are shown in Figure 4. While learning the local classifier, for each example $(\mathrm{x},\mathrm{y}) \in D$, the learning algorithm must ensure that $f_{y_t}(\mathrm{x},t) > f_{y'}(\mathrm{x},t)$, for all $t = 1,...,n_y$ and all $y' = y_t$. The online perceptron style algorithm for local learning (Har-Peled et al., 2002) is presented in figure 4(a). For learning a global classifier that produces the correct global classification, feedback from the inference process determines which classifiers to modify so that together, the classifier and the inference procedure yield the desired result. Training is done for a global criterion as in (Collins, 2002) (Carreras and Màrquez, 2003). Figure 4(b) presents the perceptron like algorithm for learning with inference feedback.

Roth and Small (2006) modified the inference based training for partial labels by updating only local components with visible labels. To ensure large margin, they require the separation between class activations to be above certain threshold γ. The modified algorithm by (Roth and Small, 2006) is shown in Figure 5.



(a) Online Local Learning with Perceptrons



(b) Online Global Learning with Perceptrons

Figure 4: Online Global and Local Learning with Perceptrons (Punyakanok et al., 2005)

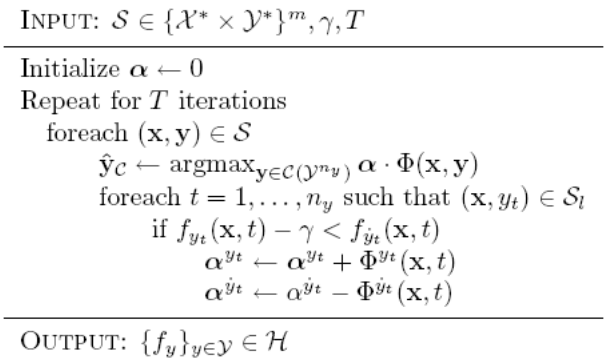


Figure 5: Online Local Learning with thick separation (Roth and Small, 2006)

Experiments and Results: The experiments for active learning were run on synthetic data and data for semantic role labeling from the CoNLL-2004 (Carreras and Màrques, 2004) shared task. Here, we discuss the results on semantic role labeling task only. The

testing was restricted to sentences with greater than five arguments to increase the number of instances with interdependent variables. The learner is bootstrapped with 100 examples and batch size of 100 examples is used for first 500 training examples and a larger batch size of 1000 examples is used after 6000 examples, since initially adding small amounts of examples to training set makes a more visible difference than adding small amounts at a later stage when the learner's performance is reasonably good.

Roth and Small (2006) show that when the data is completely locally separable, combination of local margins performs better than the global margin. For the semantic role labeling task, both functions Q_{global} and $Q_{local(\mathcal{C})}$ perform better than random selection, reducing labeling effort by 35% i.e. to achieve the maximum performance achieved by random, 35% lesser data is required for active learning strategies.

Roth and Small (2006) demonstrate that in cases where local labels can be queried independently, labeling effort can be drastically reduced using partial label queries. For semantic role labeling task, querying partial labels reduced the amount of labeled data required by 50%. But querying local variables may not be feasible for all structured output problems. For example, in parsing, where we try to learn a parse tree for a given sentence, a sequence of productions is applied to the sentence. To select the next production, one must have the entire sequence of productions preceding the prediction.

In querying for partial labels, the user is shown the entire sentence for context. The user spends sometime reading and analyzing the sentence. So, should we ask the user to label all the words in the sentence he is shown (since he is reading the whole sentence anyways) or for just the most informative one. How do we compare the effort in reading a word vs. classifying it. These questions call for some user studies to understand the real user effort involved.

3.1.4 Reducing annotation effort for Grammar Induction

Grammar induction (Hwa, 1999) is the task of inferring grammatical structure of a language by learning from example sentences in the language. (Charniak, 1996) show that grammar can be easily constructed if we have sentences labeled with their parse trees. Grammar induction using raw sentences in the language is difficult as shown in (Gold, 1967). Greedy approach using EM (Dempster et al., 1977), as in inside-outside algorithm (Baker, 1979), induces locally optimal grammar with aim to minimize entropy of the training data. Hwa (2000) use a variant of inside-outside algorithm to induce grammar expressed in Probabilistic Lexicalized Tree Insertion Grammar representation (Schabes and Waters, 1993),(Hwa, 1998)

Given enough labeled data with good quality annotations, grammar induction can be done with reasonable accuracy. However, like grammar induction is a difficult learning task, same is true for creating labeled data for learning grammars i.e. creating hand-parsed sentences. Hwa (2000) use selective sampling to minimize the amount of annotation needed for corpus based grammar induction. There are two possible ways to minimizing the annotation effort for the user: 1) reducing the amount of annotations in each sentence, 2) reducing the number of training sentences.

Hwa (2000) focus on the later and select examples from the unlabeled pool using Training Utility Value (TUV). They used uncertainty based evaluation function for estimating TUV by quantifying grammar's uncertainty about assigning a parse tree to this sentence. They consider two functions as a measure of grammar's uncertainty: 1) sentence length 2) tree entropy of the sentence. The intuition behind using sentence length as a utility measure is that longer sentences tend to have complex structures and ambiguities. This measure has a major advantage that it is easy and fast to compute.

Selective sampling based on sentence length does not take into account the hypothesis grammar. The tree-entropy selection criterion considers the classifier's distribution over

all possible parse trees which tells us about the uncertainty of the grammar. A uniform distribution implies highest uncertainty. Entropy is used as a measure to quantitatively characterize a distribution. Entropy $H(V)$ is the expected negative log likelihood of a random variable V:

$$H(V) = -\sum_{v \in V} p(v) log_2(p(v)) \tag{13}$$

Entropy definition can be applied to the probability distribution of parses for a sentence s in grammar G to calculate the tree entropy TE(s,G) as the expected number of bits needed to encode the distribution of possible parses. To compare the entropy of sentences of different lengths, we need to normalize TE. TE can be normalized using the uniform distribution over all parses for a sentence of that length. For a sentence s of length l, the number of all possible parses is $O(2^l)$ and with uniform distribution, its entropy is $O(l)$ bits. Hence, the normalized tree entropy can be defined as:

$$f_{te}(s, G) = \frac{TE(s, G)}{l} \tag{14}$$

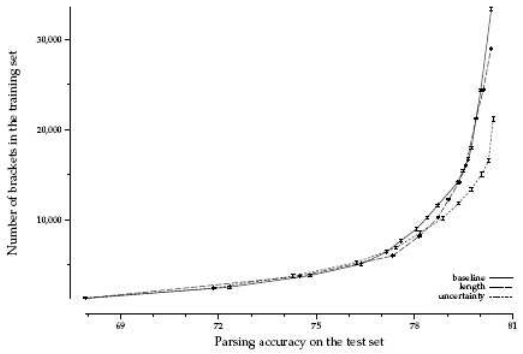
where, $TE(s, G)$ is derived as:

$$TE(s, G) = -\frac{\sum_{v \in V} Pr(v|G) log_2 Pr(v|G)}{Pr(s|G)} + log_2 Pr(s|G) \tag{15}$$

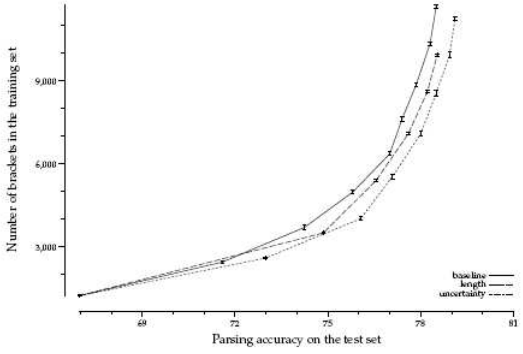
where V is the set of all possible parses G generates for s. Inside probabilities are used to compute the other probabilities $Pr(s|G)$ and $Pr(v|G)$.

Experiments and Results: Hwa (2000) used selected training sentences from the Wall Street Journal (WSJ) corpus (Marcus et al., 1993) for inducing grammars and results were reported on unseen sentences. To reduce the vocabulary size, they replaced words with their part-of-speech tags. Two candidate pool sizes of training data were used, 3500 sentences and 900 sentences. The second experiment was used to study how scarcity of training data affects the evaluation function. Annotation effort is measured in terms of number of brackets user adds, which is a more appropriate measure than number of sentences because it more accurately quantifies the effort by the human annotator. The performance of the parser is measured in terms of the consistent bracketing metric commonly used for evaluating parsers. The results for two selective sampling methods were compared with random sampling. The learner was bootstrapped with 100 examples and in every iteration 100 examples were selected from the pool and added to the training set.

For experiment 1, with larger candidate-pool, to achieve same performance as random selective sampling, tree entropy based method requires 36% lesser annotations and sentence length based method requires 9% lesser annotations than random (baseline). For experiment 2, with a smaller candidate-pool, tree entropy based method requires 27% lesser annotations and length based method requires 15% lesser annotations. It was observed that the three methods give different results when all the sentences in the pool have been added to the pool. Thus, presenting the training examples in different order affects the search path of the induction process. With the smaller pool size, gain with TE based selective sampling is lesser but the order in which training examples are presented helped in inducing slightly better grammars. The results are shown in figure 6 (a) and 6(b).



(a) Parsing accuracy on test set when candidate-pool is large



(b) Parsing accuracy on test set when candidate-pool is small

Figure 6: The learning rates for three selective sampling techniques: baseline(random), uncertainty-based (tree-entropy) and length-based (Hwa, 2000)

3.1.5 Uncertainty-based Online Active Learning

The work presented so far discusses pool-based active learning i.e. we have a pool of unlabeled examples from which examples are selected for user's annotation. However, in certain application such as spam-filtering, task of classifying email messages as spam(unwanted or harmful electronic messages) and ham(legitimate electronic messages), pool-based learning is not feasible. The messages come in a stream and the decision to recommend the example for user's analysis has to be made in real time. Moreover, pool-based active learning methods are computationally expensive requiring many passes over the entire unlabeled data. Sculley (2007) present a confidence based selective sampling method in an online learning scenario for spam filtering task.

In their approach they use a linear classifier L with weight vector w. A value p_i is defined for each x_i as $p_i = \overrightarrow{w}.\overrightarrow{x}$, and the prediction by L for example x_i is given by $sign(pi)$. $\mid p_i \mid$ signifies the distance of x_i from the classification hyper-plane. In this paper, they used three linear classifiers: classical perceptron, perceptron with margins and linear online SVMs.

The classical perceptron algorithm, an online linear classifier, was introduced in 1958 by Rosenblatt (Rosenblatt, 1958). The online training method for classical perceptron starts by initializing weight vectors to 0 ($w \leftarrow 0$). For each example x_i in training data, compute $p_i = \overrightarrow{w}.\overrightarrow{x}$, predict y_i' as the $sign(p_i)$. If $y_i' \neq y_i$, update the weights as $w \leftarrow w + y_i \eta x_i$, where η is the learning rate. Perceptron with margin is a variant of classical perceptron that tries to maintain an approximate margin between the data classes (Gentile, 2002), has good tolerance to noise (Khardon and Wachman, 2007) and performs well on spam filtering task (Schulley et al., 2006). The update algorithm changes such that, when $y_i pi \leq m$, weights are updated, $w \leftarrow w + y_i \eta x_i$ (m is the required margin).

Support Vector Machines(SVMs) (Scholkopf and Smola, 2001) is a maximum margin classifier that finds a separating hyper-plane that maximizes the distance between two data classes. In the soft margin case (i.e. allowing noise), given m training examples, in $n - dimensional$ feature space, and the slack variable $C > 0$, the hypothesis vector w and slack vector ξ is calculated to minimize:

$$\tau(\mathrm{w}, \xi) = \frac{1}{2}\|\mathrm{w}\|^2 + C\sum_{i=1}^{m}\xi_i \tag{16}$$

subject to the constraints (Scholkopf and Smola, 2001) that $\xi \geq 0$ and $y_i p_i \geq 1 - \xi_i$ for i from 1 to m. An expensive but straightforward way of converting batch-mode SVMs

to online SVM is to re-train each time an example is classified poorly i.e. $y_i p_i < 1$. By using old hypothesis as the starting point, this process can be made less expensive using iterative solvers such as sequential minimal optimization(SMO) (Platt, 1998). The online SVM performs well on spam filtering task as shown in (Sculley and Wachman, 2007) with appropriate setting for C parameter.

Online active learning is sometimes also referred to as *label efficient* learning (Helmbold and Panizza, 1997), at the end of each online classification, the learner chooses whether or not to ask for the correct classification. Sculley (2007) present three uncertainty based sampling algorithms for online active learning and compare them with a uniform sub-sampling algorithm where all samples are equally likely to be selected. First method is a randomized *label efficient* method for linear classifiers such as classical perceptron and Winnow. Given a sampling parameter 'b' (b-sampling (Cesa-Bianchi et al., 2006)), label for an example x_i is requested with probability:

$$P_i = \frac{b}{b + \mid p_i \mid} \tag{17}$$

As $\mid p_i \mid$ approaches zero, the probability of a label request for x_i approaches 1 i.e. closer an example is to the hyper-plane, less confidence the classifier has in its classification and hence it should be presented to the user. The parameter b defines a function relating the sampling probability P_i to the classification confidence $\mid p_i \mid$. Figure 7 shows how P_i varies with $\mid p_i \mid$ for difference values of b. (Cesa-Bianchi et al., 2006) give theoretical mistake bounds and expected sampling rates for b-sampling.

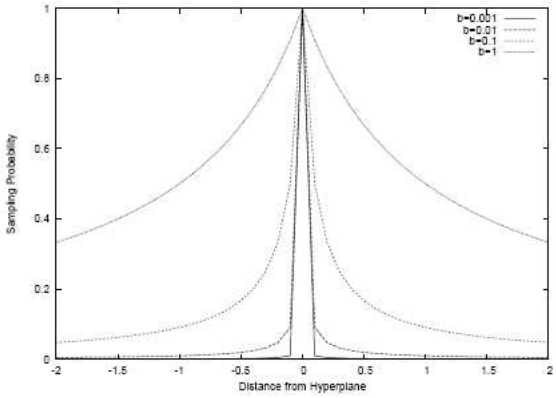


Figure 7: b-sampling probabilities, sampling probability (P_i) vs. distance from hyper-plane (p_i) (Sculley, 2007)

In the second method called Logistic Margin Sampling, another mapping from $\mid p \mid$ to P based on logistic model of confidence probabilities is used. For a given example, the confidence of a learner in the classification can be modeled as a logistic function of p_i, the signed distance of x_i from the classification hyper-plane. This approximation is reasonable given the work of (Platt, 2000). Thus, the confidence of the learner in classification of an example x_i is given as:

$$q_i = \frac{1}{1 + \exp^{-\gamma |p_i|}} \tag{18}$$

The label for x_i is requested with probability $P_i = 1 - q_i$. On simplification, we get the logistic margin sampling probability as

$$P_i = \frac{\exp^{-\gamma|p_i|}}{1 + \exp^{-\gamma|p_i|}} \tag{19}$$

Figure 8 shows how P_i varies with $\mid p_i \mid$ for difference values of γ.

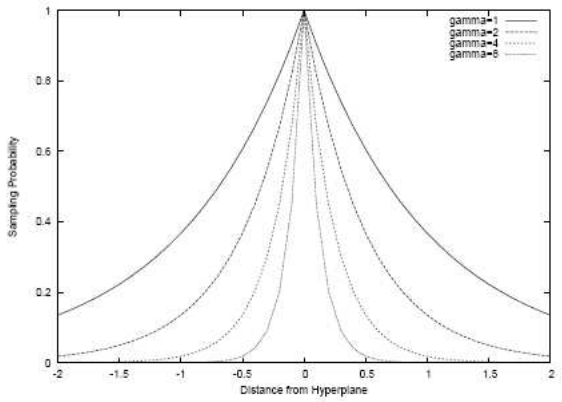


Figure 8: Logistic margin sampling probabilities, sampling probability (P_i) vs. distance from hyper-plane (p_i) (Sculley, 2007)

Third method uses fixed margin sampling, where an example x_i is selected if $\mid p_i \mid < c$, where c is a fixed threshold. Unlike the previous two methods, fixed margin sampling does not assign a non-zero sampling probability to all examples. Thus, theoretical guarantees of b-Sampling do not apply here.

Experiments and Results: Three techniques for selective sampling described above were evaluated against uniform sampling as the baseline. In uniform sampling, each example is selected with a fixed probability q. For b-sampling, b was varied from 0.001 to 1, for logistic margin sampling γ was varied from 1 to 16, for fixed margin sampling, the confidence threshold c was varied from 0.001 to 2.4 and lastly for uniform sampling, q was varied from 0.001 to 0.512. These values were selected to ensure total label request is always between 0 and $30,000$. Results reported are based on average over 10 probabilistic tests.

The data used for evaluation was TREC spam filtering data from 2005-06 (trec05p-1: 92,189 messages, trec06p-37,822 messages). All three active learning strategies achieved equivalent performance level using only 10% of the labels needed by uniform sampling to achieve same performance level. As the quality of the learner improves, allowing the learner to make more predictions with high confidence, the number of label requests reduces overtime. With the logistic margin and fixed margin methods, the number of labels requested goes down to 1% of the examples by the end of the trial. The sampling rate of b-Sampling decreases steadily, but less slowly over time. The sampling rate of uniform sub-sampling remains constant. Although fixed margin sampling performs better than b-sampling, it offers no theoretical guarantees and experiences some volatility between 1000 and 2000 label requests for the trec05p-1 tests with perceptron with margins. Figure 9 shows the results for online-SVM on trec06p. The performance is reported using the standard (1-ROCA)% measure (Cormack and Lynam, 2005). The graph shows (1-ROCHA)% score achieved over the entire online test for the given number of labels requested.

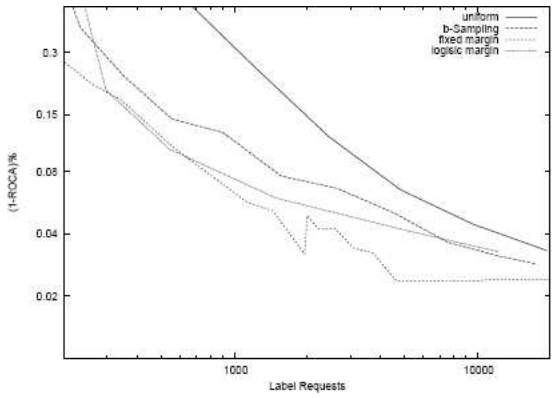


Figure 9: Online SVM on tec06p data. Scores using all data is 0.025 (Sculley, 2007)

3.2 Query-by-committee

The query-by-committee(QBC) method of active learning selects/generates a committee of hypotheses and selects the unlabeled examples on the basis of disagreement among different hypotheses. It selects the examples on which the disagreement within the committee is highest. A committee is usually generated by randomly sampling the hypothesis from the version space. This method of active learning has been applied to many machine learning algorithms such as perceptrons (Freund et al., 1997), Naïve Bayes (McCallum and Nigam, 1998) and Winnow (Liere and Tadepalli, 1997). Many semi-supervised learning methods have been used with Query-by-committee method of active learning to select the examples to be shown to the user with the aim of improving performance of classifiers using minimum amount of user labeling. Some of these techniques have been discussed below.

3.2.1 QBC Semi-supervised learning using EM

McCallum and Nigam (1998) present a technique for combining QBC based active learning with Expectation-Maximization (Dempster et al., 1977) for text classification. They use naïve bayes classifier with the assumption that the words in a document are generated independently of the context and the probability of a word is independent of its position in the document. The word probabilities conditioned on class are sampled from the posterior Dirichlet distribution based on training data word counts. The parameters are sampled k times to create a committee of k classifiers. Vote Entropy (Dagan and Engelson, 1995) (entropy of the class label distribution obtained when each committee member *vote* with probability $1/k$ for its winning class) and KL divergence to the mean (Pereira et al., 1993) are the two metrics chosen to measure disagreement among the committee of classifiers. The documents are selected on the basis of disagreement among committee members, where disagreement is calculated using either of the two metrics.

McCallum and Nigam (1998) also present *density-weighted pool-based sampling* method that prefers documents with high classification variance that are also similar to many other documents in the pool. The density in a region around a particular document is approximated by measuring the average distance from that document to all other documents. The distance $Y(d_i, d_h)$ between documents d_i and d_h is calculated by using:

$$Y(d_i, d_h) = \exp\{-\beta D(P(W|d_h)||(\lambda P(W|d_i) + (1-\lambda)P(W)))\} \tag{20}$$

where W is a random variable over the words in the vocabulary; $P(W|d_i)$ is MLE of words sampled from document d_i; $P(W)$ is the marginal distribution over words; λ determines smoothing and β determines sharpness of distance metric. The density Z of a document

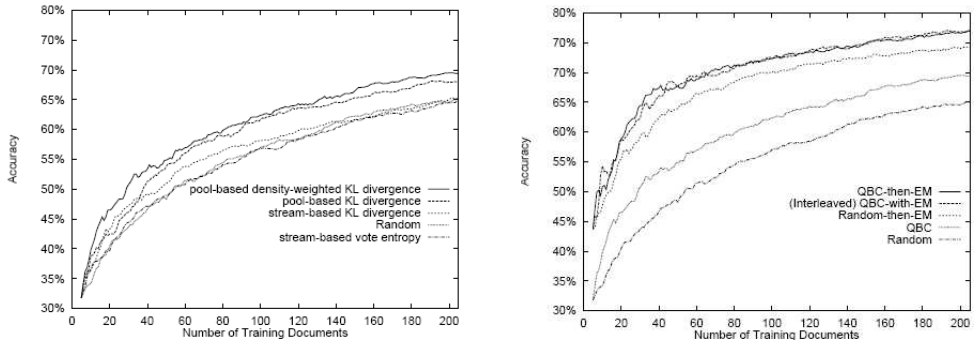
d_i is defined as

$$Z(d_i) = \exp\left\{\frac{1}{|D|}\sum_{d_h \in D} \ln(Y(d_i, d_h))\right\} \tag{21}$$

This density metric is combined with disagreement by selecting the document with highest product of density and disagreement.

One of the ways of combining active learning with EM is to run EM to convergence after actively selecting examples to be labeled. In this manner, active learning helps in selecting a better starting point for EM. McCallum and Nigam (1998) propose *pool-leveraged sampling* which interleaves EM with Active Learning. In this approach, EM is run to convergence on each committee member prior to disagreement calculations. This results in avoiding the request of labels for examples that can be already reliably filled by EM, and encouraging selection of those examples for users input that would help EM find a local maximum with higher accuracy.

Experiments and results: McCallum and Nigam (1998) use UseNet and Reuters datasets in their experiments. An initial classifier was trained with one randomly-selected labeled document per class. The QBC method uses a committee of 3 classifiers. Figure 10 shows the results for different disagreement metrics and selection strategies for QBC with and without EM. The results for QBC without EM show that density-weighted pool-based sampling with KL Divergence to the mean shows the best performance among the disagreement metrics with an accuracy of 51% requiring only 30 labeled documents compared to 40 labeled documents for unweighted measure and 59 labeled documents for random sampling. The QBC-then-EM and interleaved QBC-with-EM show comparable performance by requiring 30 and 32 labeled documents respectively for achieving an accuracy of 64%. Both the approaches do better than QBC-without-EM and random sampling strategies which require 118 and 179 labeled documents respectively to achieve the same performance. Thus, EM helps in boosting the performance of active learning a great deal.



(a) (b)

Figure 10: (a) Comparison of disagreement metrics and selection strategies for QBC shows that density weighted pool-based KL sampling does better than other metrics, (b) combinations of QBC and EM outperform stand-alone QBC or EM (McCallum and Nigam, 1998)

3.2.2 Multi-view active learning

Muslea et al. (2006) discuss the active learning techniques for *multi-view* learning tasks. *Multi-view* tasks are defined as ones which have disjoint subsets of features and each of

these subsets is sufficient to learn the target concept. The subsets of features are known as different *views* of the learning task. (Blum and Mitchell, 1998) describe the *multiple views* with reference to web-page classification where the web-pages can be classified on the basis of words in the document or the hyperlinks pointing to the documents.

Muslea et al. (2006) introduce Co-Testing, which is a two-stage iterative algorithm. It takes a few labeled examples as input along with a large number of unlabeled examples. In the first stage, Co-Testing uses the labeled examples to learn a hypothesis in each view and in the second stage, the learned hypotheses are applied to unlabeled examples to find a set of data points on which hypotheses predict different labels. Such data points are known as *contention points*. In the final stage, it selects one of the contention points to be labeled by the user and adds it to the training set. The above process is repeated to select more examples for user labeling. After the *allowed* number of queries are made, this algorithm creates a final *output hypothesis* to make actual predictions. The above process is common to a whole family of Co-Testing algorithms which differ from each other in the strategies used to select the next unlabeled training examples from the contention points and the process by which the output hypothesis is constructed.

Muslea et al. (2006) discuss three types of strategies for query selection.

1. *naïve*: This strategy chooses a query point randomly from the set of contention points. This is useful for the learners which cannot estimate confidence for their predictions reliably. This strategy is independent of domain and learner, so it can be used with any multi-view learning task.

2. *aggressive*: This strategy selects the contention point on which least confident of the hypotheses makes the most confident prediction.

$$Q = \underset{x \in ContentionPoints}{\arg\max} \left\{ \min_{i \in \{1,2,\ldots,k\}} Confidence(h_i(x)) \right\} \tag{22}$$

   where $h_i, 1 \leq i \leq k$ are k hypotheses. This is useful for domains which are of high accuracy with little noise.

3. *conservative*: This strategy chooses the contention point on which the confidence of prediction by the different hypotheses is as close as possible.

$$Q = \underset{x \in ContentionPoints}{\arg\min} \left(\max_{f \in \{h_1,h_2,\ldots,h_k\}} Confidence(f(x)) - \min_{g \in \{h_1,h_2,\ldots,h_k\}} Confidence(g(x)) \right) \tag{23}$$

   where $h_i, 1 \leq i \leq k$ are k hypotheses. This strategy is useful for high noise domains where aggressive strategy would select mostly *noisy* examples.

Three strategies have been discussed for creating the output hypothesis.

1. *weighted vote*: The hypotheses' votes are combined using the confidence weights of their predictions

$$h_{OUT}(x) = \underset{l \in Labels}{\arg\max} \left(\sum_{\substack{g \in \{h_1,h_2,\ldots,h_k\} \\ g(x)=l}} Confidence(g(x)) \right) \tag{24}$$

   where $h_i, 1 \leq i \leq k$ are k hypotheses.

2. *majority vote*: The predicted label is the one that was predicted by most of the hypotheses learned in k views.

$$h_{OUT}(x) = \arg\max_{l \in Labels} \left(\sum_{\substack{g \in \{h_1, h_2, \ldots, h_k\} \\ g(x)=l}} 1 \right) \tag{25}$$

where $h_i, 1 \leq i \leq k$ are k hypotheses. This strategy is applicable only in the case of three or more views and its useful when the learners cannot estimate the confidence reliably.

3. *winner-takes-all*: The output hypothesis chosen is the one that makes smallest number of mistakes over N queries, learned in a view. If $Mistakes(h_i)$ is the number of mistakes made by hypothesis h_i learned in k views on N queries,

$$h_{OUT}(x) = \arg\max_{g \in \{h_1, \ldots, h_k\}} Mistakes(g) \tag{26}$$

Experiments and Results: The paper presents results over three domains: web-page classification, discourse tree parsing and advertisement removal. The data-sets used are as follows:

1. AD (Kushmerick, 1999): This is a classification problem where the web-pages are to be classified as **ads** and **non-ads**. The data-set has 2 classes, 1500 attributes and 3279 examples. The views used are V_1: textual features that describe the image e.g., 1-gram and 2-gram from caption, from URL of the page etc., V_2: properties of the image itself: length, width, aspect ratio, etc.

2. COURSES (Blum and Mitchell, 1998): This problem classifies web-pages as **course home-pages** and **other pages**. The data-set has 2 classes, 2206 features and 1042 examples. The views used are V_1:words that appear in the page and V_2 words that appear in hyperlinks pointing to them.

3. TF (Marcu et al., 2000): This classification problem, in the context of machine translation system, uses shift-reduce paradigm to learn how to re-write Japanese discourse trees as English-like discourse trees. The views used are V_1: features specific to shift-reduce parser(elements in the input list and partial trees in stack) and V_2: features specific to Japanese tree given as input.

Domain	$\mathcal{L}$	Co-Testing		Single-view Algorithms				
		Query Selection	Output Hypothesis	QBC	qBag	qBst	US	Rnd
AD	IB	*naive*	*winner*	–	√	√	√	√
TF	MC4	*naive*	*winner*	–	√	√	–	√
COURSES	Naive Bayes	*naive*	*weighted vote*	√	√	√	√	√
		conservative						

Figure 11: The algorithms used for classification. The last five columns denote Query-by-Committee/-Bagging/-Boosting, Uncertainty Sampling and Random Sampling. (Muslea et al., 2006)

The base learners used for AD is IB (instance-based learning) (Aha, 1992), for COURSES is naïve bayes (Blum and Mitchell, 1998) and for TF is MC4(which is an implementation of C4.5 (Quinlan, 1993)). Five single-view algorithms are used for comparison with co-testing, viz. random sampling, query-by-bagging, query-by-boosting, uncertainty

sampling and query-by-committee. Here, we discuss the comparison of co-testing with random-sampling, uncertainty-based sampling and query-by-committee. For comparison with query-by-bagging and query-by-boosting, the reader is encouraged to refer to the paper for details. Figure 11 presents an overview of the experiments done by (Muslea et al., 2006). The single-view algorithms were trained using all available features (i.e. $V_1 \cup V_2$). The uncertainty-based sampling is applied only on AD and COURSES because MC4, the base learner for TF, does not provide confidence scores. Query-by-committee(QBC) selects a committee of classifiers by randomly sampling hypotheses from the version space. QBC cannot be applied to AD and TF because randomly sampling from IB or MC4 version spaces is not possible. They applied QBC to COURSES by sampling the hypotheses from the *gamma* distribution of parameters in naïve bayes (McCallum and Nigam, 1998).

Given the limitations of base learners (not being able to estimate confidence) AD and TF, naïve co-testing with *winner-takes-all* output hypothesis is applied. For courses, weighted vote of the classifiers learned in each view is used for output hypotheses. For COURSES data, the two query selection strategies: naïve and conservative are compared. Aggressive query selection is not applicable here as one of views is significantly less accurate than the other.

Performance is evaluated on 10-fold cross validation. For each of the algorithm tested here, for AD, 150 randomly chosen examples are used initially followed by 10 queries in each iteration. For COURSES, 6 randomly chosen examples are used at the start and one query per iteration is made. Finally, for TF, initially 110 examples are chosen randomly and 20 queries are made in each iteration.

Figure 12 shows the results for statistical significance tests(t-test confidence of atleast 95%) from the pair-wise comparison of algorithms. The experiment results show that none of the tested *single-view algorithm* significantly outperforms co-testing on any of the comparison points. The results could be understood in the following manner. The numbers (0,0,19) in the first row show results of comparing naïve co-testing and random sampling on AD. This means that on all 19 comparison points, naïve co-testing outperforms random sampling significantly. In a similar manner, looking at the last row, conservative co-testing significantly outperforms naïve co-testing on 28 comparison points and the differences were statistically insignificant on the 21 points other points where there is a tie between the two. Figure 13 shows the learning curves of different algorithms used in experiments on AD, TF and COURSES.

Algorithm	Naive Co-Testing						Conservative Co-Testing		
	AD			TF			COURSES		
	Loss	Tie	Win	Loss	Tie	Win	Loss	Tie	Win
Random Sampling	0	0	19	0	21	70	0	0	49
Uncertainty Sampling	0	2	17	0	2	89	-	-	-
Query-by-Committee	-	-	-	0	60	31	-	-	-
Query-by-Bagging	0	18	1	0	6	85	0	28	21
Query-by-Boosting	0	15	4	0	0	91	0	0	49
Naive Co-Testing	-	-	-	-	-	-	0	21	28

Figure 12: Statistical significance results in the empirical (pair-wise) comparison of the various algorithms on the three domains. (Muslea et al., 2006)

3.2.3 Bootstrapping Statistical Parsers

Steedman et al. (2003) present selection methods for co-training (Blum and Mitchell, 1998) and corrected co-training (Pierce and Cardie, 2001) where selected parse trees from the bootstrapped examples are presented to the user for correction before adding them to the training pool. This approach is motivated by the observation that the quality

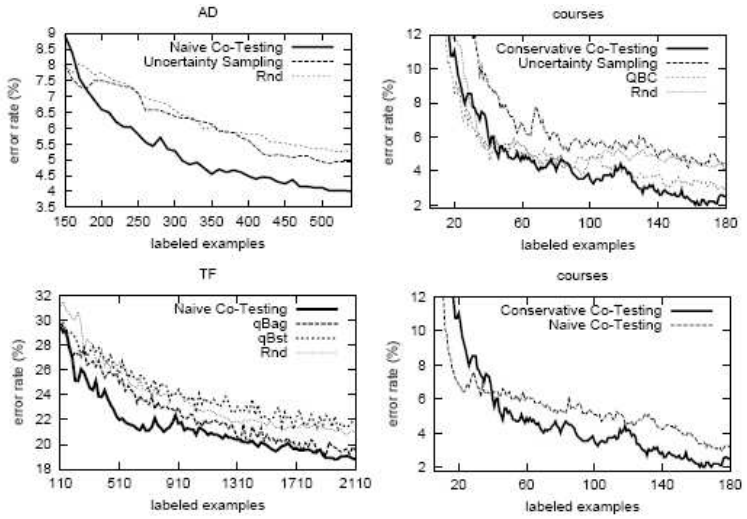


Figure 13: Empirical results on the AD and TF and COURSES problems (Muslea et al., 2006)

of the bootstrapped data is the key factor in the convergence of co-training. Corrected co-training is an extension to co-training where user corrects the training data selectively.

The goal for both parsers in co-training is to improve by learning from each others' strengths. The examples added to the training set of one parser (referred to as the student) are selected from those produced by the other parser (referred to as the teacher). The selection mechanism is important for both co-training and sample selection for user correction; however their selection methods focus on different criteria: co-training favors selecting accurately labeled examples, while sample selection favors selecting examples with high training utility, which are often not sentences that the parser already labeled accurately. Steedman et al. (2003) propose selection methods and their experiments explore the trade-off between maximizing training utility and minimizing errors. Accuracy and training utility are two selection criteria used for selecting documents to be added to the training set. Training utility of a sentence is estimated by comparing the score the student learner assigns to its parse against the score the teacher assigns to its own parse.

Sample selection is a two step process, where first each parser uses some scoring function to score the parses it generated on the unlabeled data and then the selection procedure chooses a subset of these labeled sentences to add to the training data. An ideal scoring function would be the true accuracy rate (e.g. F1-score). In practice, the scoring function can be approximated by a measure of confidence from the learner such as conditional probability of the most likely parse. Steedman et al. (2003) experiment with two scoring functions: 1) oracle scoring function $f_{F1-score}$ (F1-score of the parse as measured against a gold-standard) 2) conditional probability of the parse given the sentence f_{prob}.

Selection methods they used and compared are as follows:

1. Above-n : the score of the teacher's parse (using its scoring function) $\geq n$. It attempts to maximize the accuracy of the data.

2. Difference: the score of the teacher's parse is greater than the score of student's parse by some threshold n. It attempts to maximize the training utility.

3. Intersection: shortlist a set of sentences such that teacher's parse score on these sentences is in top n percent of scored sentences. Similarly shortlist a set of documents such that student's parse score on these sentences is in lowest n percent of scored

sentences. The intersection of these two sets is selected for addition to the training set. It attempts to maximize the accuracy as well as training utility.

This approach aims at reducing the number of corrections made by the human which differs from co-testing (Muslea, 2002), where the goal is to reduce the total number of labeled training examples. Therefore, the selection methods must take into account the quality of the parse produced by the teacher in addition to how different its parse is from the one produced by the student. The intersection method precisely aims at selecting sentences that satisfy both requirements.

Experiments and Results: The two parsers used in experiments are (1) Lexicalized CFG parser (Collins, 2003) and (2) Lexicalized Tree Adjoining Grammar parser (Sarkar, 2002). A seed set size of 1000 sentences was used, taken from section 2 of the Wall Street Journal (WSJ) Penn Treebank. The total pool of unlabeled sentences was the remainder of sections 2-21 (stripped of their annotations), consisting of about 38,000 sentences. The parsers were evaluated on unseen test sentences (section 23 of the WSJ corpus) and section 0 was used as a development set for determining parameters. The Parseval F1-score over labeled constituents is used as the evaluation metric.

$$F1 - Score = \frac{2 \times LR \times LP}{LR + LP} \tag{27}$$

where LP is labeled precision rate and LR is labeled recall rate.

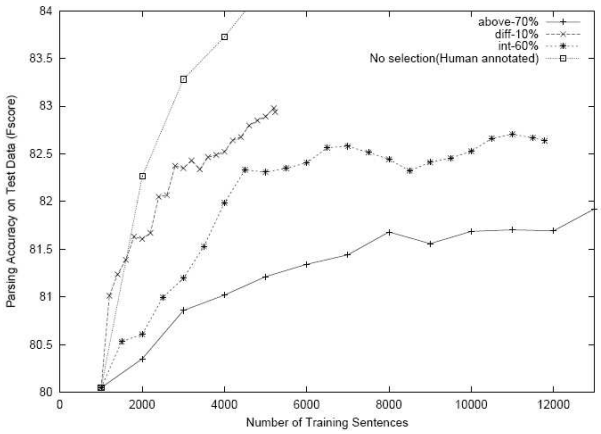


Figure 14: A comparison of selection methods for co-training with oracle scoring function (Steedman et al., 2003)

Selection methods and co-training: To evaluate the selection methods using a reliable scoring function, Steedman et al. (2003) use the oracle scoring function $f_{F1-score}$ as it guarantees a perfect assessment of the parser's output. However, in practice, a reliable scoring function can assign a high score to an incorrect parse tree. Thus, for quality control, the selection method's parameters are chosen such that average accuracy rate for newly labeled training data in each iteration is 85%. Three strategies that were evaluated are: $S_{above-70\%}$ strategy (select examples with $F1 - score > 70\%$) adds 330 labeled sentences (from the pool of 500 sentences) with an average accuracy rate of 85% per iteration on training data. $S_{diff-10\%}$ strategy (select examples with difference in $F1 - score$ of teacher and student learner $> 10\%$) adds 50 labeled sentences with an average accuracy rate of 80% on training data and $S_{int-60\%}$ (selects examples for which the teacher's parse is in its top 60% and student's parse is in its bottom 60%) adds about 150 labeled sentences with an average accuracy rate of 85% on training data. Figure 14 shows the results of the proposed selection methods in co-training using the oracle scoring function $f_{F1-score}$.

$S_{diff-10\%}$ strategy improves the parser the most. This suggest that training utility is an important criteria in selecting training examples for co-training.

The selection methods were also evaluated using the estimated scoring function, f_{porb}. The parameters for the selection methods were selected such that $30 - 50$ examples were added in each iteration. Three strategies that were evaluated are: $S_{above-70\%}$ (average accuracy rate of the training data - 85%),$S_{diff-30\%}$ (average accuracy rate of the training data - 75%) and $S_{int-30\%}$ (average accuracy rate of the training data - 75%). As expected, the parser performance with f_{prob} was lower than $f_{F1-score}$ as shown in 15. However, $S_{diff-30\%}$ and $S_{int-30\%}$ selection methods helped in improving the parser with 5% reduction in error.

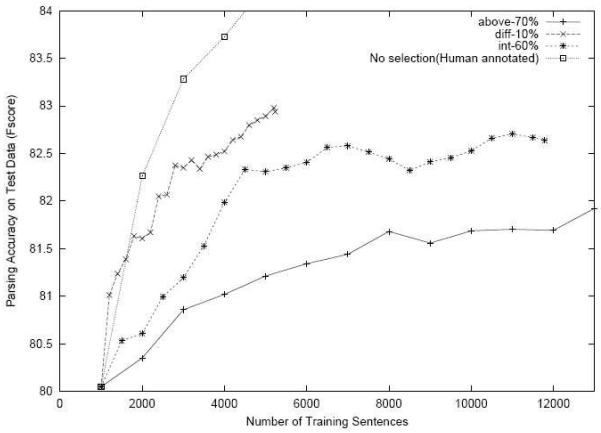
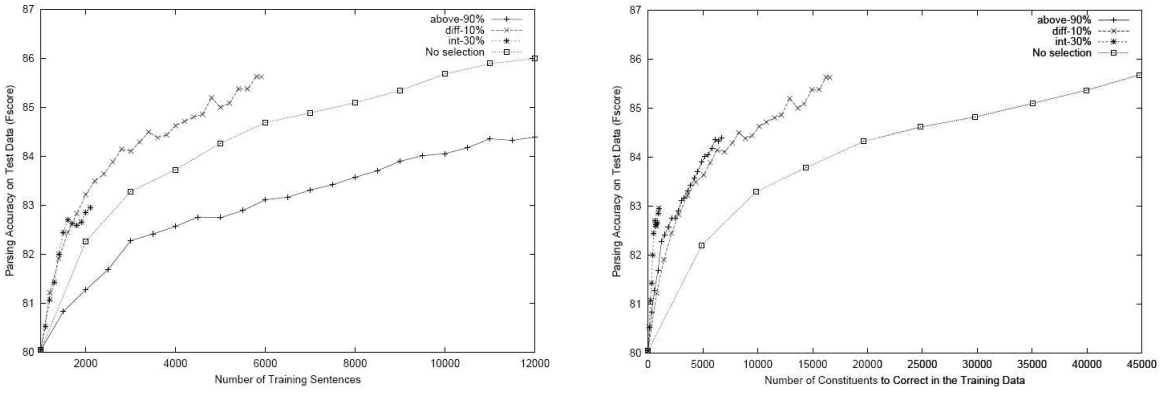


Figure 15: A comparison of selection methods for co-training with conditional probability scoring function(Steedman et al., 2003)

Selection methods and corrected co-training: Figure 16 shows the results of the proposed selection methods in corrected co-training using the oracle scoring function $f_{F-score}$. The graphs show the amount of human effort in terms of number of sentences a human has to check and number of constituents a human has to correct. The results show that $S_{int-30\%}$ improves the parser at the fastest rate and for the same performance level, it selects fewest number of sentences for a human to check in addition to ensuring that human makes least number of corrections.



(a) (b)

Figure 16: A comparison of selection methods for corrected co-training (a) in terms of the number of sentences added to the training data; (b) in terms of the number of manually corrected constituents (Steedman et al., 2003)

3.3 Other Criteria

Apart from uncertainty-based and query-by-committee based selective sampling techniques, researchers have also explored other measures for selecting examples for presenting to the user. We discuss some of these techniques here:

Diversity and Representativeness Based Selective Sampling

Shen et al. (2004) apply active learning to the named entity recognition task using support vector machines (SVM) (Cortes and Vapnik, 1995). In this paper, they apply active learning methods to a simple and effective SVM model to recognize one class of names at a time, such as protein names, person names, etc. In addition to the uncertainty-based techniques for selective sampling based on distance of a word's feature vector from the hyper-plane, they use diversity criterion to maximize the training utility of a batch of examples. They demonstrate that a batch in which the examples have higher variance is more useful for active learning. They propose two methods for diversity sampling, global and local. For global consideration, they clusters (using k-means clustering (Kanungo et al., 2002)) based on similarity and select examples from different clusters. For local consideration, they select those examples which are most different from examples already in the pool.

Shen et al. (2004) also suggest that the most informative examples are those which are most representative of other examples. Representativeness of an example can be calculated as the number of examples similar to it. Examples with high representativeness are less likely to be an outlier and adding them to the training set will have an effect on a large number of unlabeled examples. If the examples were clustered together based on similarity, the centroids of the clusters would be the most representative examples. To measure similarity, they used cosine similarity and Dynamic Time Warping (DTW) algorithm (Rabiner et al., 1978) to find an optimal alignment between the words in the sequences which maximizes the accumulated similarity degree between the sequences.

Experiments and Results: For evaluation, they apply their techniques to recognize protein (PRT) names in biomedical domain using GENIA corpus V1.1 (Kim et al., 2003). The learner was bootstrapped with 10 examples sentences and each active learning iteration 50 examples were added to the training set. To achieve an F-score of 63.3, random sampling requires $83k$ sentences where as their strategy based on representativeness and uncertainty requires only $31k$ sentences (i.e. 39% fewer examples are needed).

4 Conclusion and Discussion

Active learning eases the burden of labeling data by selecting the most informative examples for the user to label. It is especially important in NLP where labeled data is usually limited. In this literature review, we present different techniques that have been used in active learning for NLP tasks such as named entity recognition, text categorization, semantic role labeling, grammar induction and parsing; and machine learning techniques such as SVM, CRFs, EM and perceptron algorithm. We show how active learning has helped these tasks in reducing the required labeled data significantly. In this review, we also discuss how the active learning approaches are evaluated. The main aim of active learning is to reduce the user annotation effort. Expected Number of User Actions (ENUA) (Kristjannson et al., 2004) is one direct measure of user effort. However, ENUA ignores the variation in effort involved in different user tasks such as annotating and reading text.

In structured outputs, Roth and Small (2006) discussed querying partial labels where the whole sentence is presented for context. It is an open question as to whether reading the whole sentence and labeling only one output is more efficient than reading and labeling the whole sentence. The latter is more beneficial for the learner since we get more labels. There is a need for effective user studies to analyze the real user effort involved which would help us direct our research in the correct direction. Also, in NLP all examples are not equal, a more difficult example would require more user effort. Hence, it is important to understand how to effectively measure the complexity of an example and select those examples which are beneficial for the classifier and also not so difficult to label.

In addition to selective sampling examples for user's annotation, semi-supervised techniques have been combined with active learning to make use of the large amount of unlabeled data usually available in NLP. This literature review presents some of the work where semi-supervised learning is combined with active learning.

A lot of interesting work has been done in active learning for NLP but as highlighted in this review, there is scope for improvement and there are some issues to address. Also, active learning can be applied to benefit more NLP problems. Each problem presents some interesting challenges that should be addressed for significant gain from active learning.

Acknowledgments

We are thankful to Noah Smith, Einat Minkov and Eric Nyberg whose guidance helped us in this literature review. We would also like to extend our thanks to Kevin Small, Dan Roth and D. Schuley for clearing our doubts in their work.

References

Aha, D. W. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, 36(2):267–287.

Baker, J. (1979). Trainable grammars for speech recognition. In *Speech communication papers presented at the 97th Meeting of the Acoustical Society*, pages 547–550.

Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT' 98: Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, New York, NY, USA. ACM.

Califf, M. E. (1998). *Relational learning techniques for natural language information extraction*. PhD thesis, The University of Texas at Austin. Supervisor-Raymond J. Mooney.

Carreras, X. and Màrques, L. (2004). Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97. Boston, MA, USA.

Carreras, X. and Màrquez, L. (2003). Online learning via global feedback for phrase recognition. In *NIPS*.

Cesa-Bianchi, N., Gentile, C., and Zaniboni, L. (2006). Worst-case analysis of selective sampling for linear classification. *Journal Machine Learning Research*, 7:1205–1230.

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA.

Charniak, E. (1996). Tree-bank grammars. In *AAAI/IAAI, Vol. 2*, pages 1031–1036.

Collins, M. (2002). Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *EMNLP '02: Proceedings of the ACL-02 conference on Empirical methods in natural language processing*, pages 1–8, Morristown, NJ, USA. Association for Computational Linguistics.

Collins, M. (2003). Head-driven statistical models for natural language parsing. *Comput. Linguist.*, 29(4):589–637.

Cormack, G. and Lynam, T. (2005). Trec 2005 spam track overview.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3):273–297.

Culotta, A., Kristjansson, T., McCallum, A., and Viola, P. (2006). Corrective feedback and persistent learning for information extraction. *Artificial Intelligence*, 170(14):1101–1122.

Culotta, A. and McCallum, A. (2004). Confidence estimation for information extraction. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, Boston, MA.

Culotta, A. and McCallum, A. (2005). Reducing labeling effort for structured prediction tasks. In *Twentieth Conference of the American Association for Artificial Intelligence (AAAI 2005)*, pages 746–751.

Dagan, I. and Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In *International Conference on Machine Learning*, pages 150–157.

Dempster, A., Laird, N., , and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, Series B:1–38.

Freund, Y., Seung, H. S., Shamir, E., and Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

Gentile, C. (2002). A new approximate maximal margin classification algorithm. *J. Mach. Learn. Res.*, 2:213–242.

Gold, E. M. (1967). Language identification in the limit. *Information and Control*, 10(5):447–474.

Har-Peled, S., Roth, D., and Zimak, D. (2002). Constraint classification: A new approach to multiclass classification. In *ALT '02: Proceedings of the 13th International Conference on Algorithmic Learning Theory*, pages 365–379, London, UK. Springer-Verlag.

Helmbold, D. and Panizza, S. (1997). Some label efficient learning results. In *COLT '97: Proceedings of the tenth annual conference on Computational learning theory*, pages 218–230, New York, NY, USA. ACM.

Hwa, R. (1998). An empirical evaluation of probabilistic lexicalized tree insertion grammars. In *COLING-ACL*, pages 557–563.

Hwa, R. (1999). Supervised grammar induction using training data with limited constituent information. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 73–79, Morristown, NJ, USA. Association for Computational Linguistics.

Hwa, R. (2000). Sample selection for statistical grammar induction. In *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora*, pages 45–52, Morristown, NJ, USA. Association for Computational Linguistics.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2002). An efficient k-means clustering algorithm: analysis and implementation.

Khardon, R. and Wachman, G. (2007). Noise tolerant variants of the perceptron algorithm. *J. Mach. Learn. Res.*, 8:227–248.

Kim, J. D., Ohta, T., Tateisi, Y., and Tsujii, J. (2003). Genia corpus–semantically annotated corpus for bio-textmining. *Bioinformatics*, 19 Suppl 1.

Kristjannson, T., Culotta, A., Viola, P., and Callum, A. M. (2004). Interactive information extraction with constrained conditional random fields. In *Nineteenth Conference of the American Association for Artificial Intelligence (AAAI 2004)*, San Jose, CA.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *AGENTS '99: Proceedings of the third annual conference on Autonomous Agents*, pages 175–181, New York, NY, USA. ACM.

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*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Liere, R. and Tadepalli, P. (1997). Active learning with committees for text categorization. In *Proceedings of 14th Conference of the American Association for Artificial Intelligence (AAAI)*, pages 591–596, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Marcu, D., Carlson, L., and Watanabe, M. (2000). The automatic translation of discourse structures. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*, pages 9–17, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: the penn treebank. *Computational Linguistics*, 19(2):313–330.

McCallum, A. and Nigam, K. (1998). Employing em and pool-based active learning for text classification. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 350–358, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Muggleton, S. and Raedt, L. D. (1994). Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679.

Muslea, I., Minton, S., and Knoblock, C. A. (2006). Active learning with multiple views. *Journal of Artificial Intelligence Resesearch (JAIR)*, 27:203–233.

Muslea, I. A. (2002). *Active learning with multiple views.* PhD thesis, University of Southern California, Los Angeles, CA, USA. Adviser-Craig Knoblock.

Pereira, F., Tishby, N., and Lee, L. (1993). Distributional clustering of english words. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 183–190, Morristown, NJ, USA. Association for Computational Linguistics.

Pierce, D. and Cardie, C. (2001). Limitations of co-training for natural language learning from large datasets. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP-2001)*.

Platt, J. (2000). Probabilistic outputs for support vector machines and comparison to regularize likelihood methods. In Smola, A., Bartlett, P., Schoelkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 61–74.

Platt, J. C. (1998). Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Microsoft Research, Redmond, Washington.

Punyakanok, V., Roth, D., tau Yih, W., and Zimak, D. (2005). Learning and inference over constrained output. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-2005)*, pages 1124–1129, Edinburgh, Scotland, UK.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, L. R., Rosenberg, A. E., and Levinson, S. E. (1978). Considerations in dynamic time warping algorithms for discrete word recognition. In *IEEE Transactions on acoustics, speech and signal processing.*

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

Roth, D. and Small, K. (2006). Active learning with perceptron for structured output. In *ICML 06: Workshop on Learning in Structured Output Spaces.*

Sarkar, A. (2002). *Statistical Parsing Algorithms for Lexicalized tree adjoining grammar.* PhD thesis, University of Pennsylvania, CA, USA.

Schabes, Y. and Waters, R. (1993). Stochastic lexicalized context-free grammar. In *3rd International Workshop on Parsing Technologies (IWPT'93)*, pages 257–266, Tilburg, The Netherlands.

Scholkopf, B. and Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA.

Schulley, D., Wachman, G., and Brodley, C. (2006). Spam filtering using inexact string matching in explicit feature space with on-line linear classifiers. In *The Fifteenth Text REtrieval Conference (TREC).*

Sculley, D. (2007). Online active learning methods for fast label-efficient spam filtering. In *CEAS 2007: Proceedings of the Fourth Conference on Email and Anti-Spam.*

Sculley, D. and Wachman, G. M. (2007). Relaxed online svms for spam filtering. In *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 415–422, New York, NY, USA. ACM.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141, Morristown, NJ, USA. Association for Computational Linguistics.

Shen, D., Zhang, J., Su, J., Zhou, G., and Tan, C.-L. (2004). Multi-criteria-based active learning for named entity recognition. In *ACL '04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 589, Morristown, NJ, USA. Association for Computational Linguistics.

Steedman, M., Hwa, R., Clark, S., Osborne, M., Sarkar, A., Hockenmaier, J., Ruhlen, P., Baker, S., and Crim, J. (2003). Example selection for bootstrapping statistical parsers. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 157–164, Morristown, NJ, USA. Association for Computational Linguistics.

Thompson, C. A., Califf, M. E., and Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. In *Proceedings of 16th International Confernce on Machine Learning (ICML-1999)*, pages 406–414. Morgan Kaufmann, San Francisco, CA.

Zelle, J. and Mooney, R. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the 14th Conference of the American Association for Artificial Intelligence (AAAI 1996)*, pages 1050–1055, Portland, OR. AAAI Press/MIT Press.