Journal of Artificial Intelligence Research 27 (2006) 203-233 Submitted 01/06; published 10/06

Active Learning with Multiple Views

Ion Muslea IMUSLEA@LANGUAGEWEAVER.COM
Language Weaver, Inc.
4640 Admiralty Way, Suite 1210
Marina del Rey, CA 90292

Steven Minton MINTON@FETCH.COM
Fetch Technologies, Inc.
2041 Rosecrans Ave., Suite 245
El Segundo, CA 90245

Craig A. Knoblock KNOBLOCK@ISI.EDU
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292

Abstract

Active learners alleviate the burden of labeling large amounts of data by detecting and asking the user to label only the most informative examples in the domain. We focus here on active learning for *multi-view* domains, in which there are several disjoint subsets of features (*views*), each of which is sufficient to learn the target concept. In this paper we make several contributions. First, we introduce Co-Testing, which is the first approach to multi-view active learning. Second, we extend the multi-view learning framework by also exploiting *weak views*, which are adequate only for learning a concept that is more general/specific than the target concept. Finally, we empirically show that Co-Testing outperforms existing active learners on a variety of real world domains such as wrapper induction, Web page classification, advertisement removal, and discourse tree parsing.

1. Introduction

Labeling the training data for a machine learning algorithm is a tedious, time consuming, error prone process; furthermore, in some application domains, the labeling of each example may also be extremely expensive (e.g., it may require running costly laboratory tests). *Active learning* algorithms (Cohn, Atlas, & Ladner, 1994; Roy & McCallum, 2001; Tong & Koller, 2001) cope with this problem by detecting and asking the user to label only the most informative examples in the domain, thus reducing the user's involvement in the data labeling process.

In this paper, we introduce Co-Testing, an active learning technique for *multi-view* learning tasks; i.e., tasks that have several disjoint subsets of features (*views*), each of which is sufficient to learn the concepts of interest. For instance, Web page classification is a multi-view task because Web pages can be classified based on the words that appear *either* in the documents *or* in the hyperlinks pointing to them (Blum & Mitchell, 1998); similarly, one can classify segments of televised broadcast based *either* on the video *or* on the audio information, or one can perform speech recognition based on *either* sound *or* lip motion features (de Sa & Ballard, 1998).

Co-Testing is a two-step iterative algorithm that requires as input a few labeled and many unlabeled examples. First, Co-Testing uses the few labeled examples to learn a hypothesis in each view. Then it applies the learned hypotheses to all unlabeled examples and detects the set of *contention points* (i.e., unlabeled examples on which the views predict a different label); finally, it *queries* (i.e., asks the user to label) one of the contention points, adds the newly labeled example to the training set, and repeats the whole process. Intuitively, Co-Testing relies on the following observation: if, for an unlabeled example, the hypotheses learned in each view predict a different label, at least one of them makes a mistake on that particular prediction. By asking the user to label such a contention point, Co-Testing is guaranteed to provide useful information for the view that made the mistake.

In this paper we make several contributions. First, we introduce Co-Testing, a family of active learners for multi-view learning tasks. Second, we extend the traditional multi-view learning framework by also allowing the use of *weak views*, in which one can adequately learn only a concept that is strictly more general or more specific than the target concept (all previous multi-view work makes the *strong view* assumption that each view is adequate for learning the target concept). Last but not least, we show that, in practice, Co-Testing clearly outperforms existing active learners on a variety of real world domains such as wrapper induction, Web page classification, advertisement removal, and discourse tree parsing.

Compared with previous work, Co-Testing is unique in several ways:

1. existing multi-view approaches (Blum & Mitchell, 1998; Collins & Singer, 1999; Pierce & Cardie, 2001), which also use a small set of labeled and a large set of unlabeled examples, are based on the idea of bootstrapping the views from each other. In contrast, Co-Testing is the first algorithm that exploits multiple views for active learning purposes. Furthermore, Co-Testing allows the simultaneous use of strong and weak views without additional data engineering costs.

2. existing active learners, which pool all domain features together, are typically designed to exploit some properties specific to a particular (type of) *base learner* (i.e., the algorithm used to learn the target concept); for example, *uncertainty reduction* methods assume that the base learner provides a reliable estimate of its confidence in each prediction. In contrast, Co-Testing uses the multiple views to detect the contention points, among which it chooses the next query. This approach has several advantages:

- it converges quickly to the target concept because it is based on the idea of *learning from mistakes* (remember that each contention point is guaranteed to represent a mistake in at least one of the views). In contrast, existing active learners often times query examples that are classified correctly, but with a low confidence.
- in its simplest form (i.e., Naive Co-Testing, which is described in section 4), it makes *no assumptions* about the properties of the base learner. More precisely, by simply querying an arbitrary contention point, Co-Testing is guaranteed to provide "the mistaken view" with a highly informative example.
- by considering only the contention points as query candidates, it allows the use of query selection heuristics that - computationally - are too expensive to be applied to the entire set of unlabeled examples.

The remainder of the paper is organized as follows. First, we introduce the concepts and notation, followed by a comprehensive survey of the literature on active and multi-view learning. Then we formally introduce the Co-Testing family of algorithms and we present our empirical evaluation on a variety of real-world domains.

2. Preliminaries: Terminology and Notation

For any given learning task, the set of all possible domain examples is called the *instance space* and is denoted by X. Any $x \in X$ represents a particular *example* or *instance*. In this paper we are concerned mostly with examples that are represented as *feature vectors* that store the values of the various *attributes* or *features* that describe the example.

The concept to be learned is called the *target concept*, and it can be seen as a function $c : X \rightarrow \{l_1, l_2, \ldots, l_N\}$ that classifies any instance x as a member of one of the N classes of interest $l_1, l_2, \ldots, l_N$. In order to learn the target concept, the user provides a set of *training examples*, each of which consists of an instance $x \in X$ and its label, $c(x)$. The notation $\langle x, c(x) \rangle$ denotes such a training example. The symbol L is used to denote the set of labeled training examples (also known as the *training set*).

Given a training set L for the target concept c, an inductive learning algorithm $\mathcal{L}$ searches for a function $h : X \rightarrow \{l_1, l_2, \ldots, l_N\}$ such that $\forall x \in X, h(x) = c(x)$. The learner $\mathcal{L}$ searches for h within the set H of all possible hypotheses, which is (typically) determined by the person who designs the learning algorithm. A hypothesis h is *consistent* with the training set L if and only if $\forall \langle x, c(x) \rangle \in L, h(x) = c(x)$. Finally, the *version space* $VS_{H,L}$ represents the subset of hypotheses in H that are consistent with the training set L.

By definition, a *passive learning* algorithm takes as input a randomly chosen training set L. In contrast, *active learning* algorithms have the ability to choose the examples in L. That is, they detect the most informative examples in the instance space X and ask the user to label only them; the examples that are chosen for labeling are called *queries*. In this paper we focus on *selective sampling* algorithms, which are active learners that choose the queries from a given *working set* of unlabeled examples U (we use the notation $\langle x, ? \rangle$ to denote an unlabeled examples). In this paper the terms *active learning* and *selective sampling* are used interchangeably.

In the traditional, *single-view* machine learning scenario, a learner has access to the entire set of domain features. By contrast, in the *multi-view* setting one can partition the domain's features in subsets (*views*) that are *sufficient* for learning the target concept. Existing multi-view learners are *semi-supervised* algorithms: they exploit unlabeled examples to boost the accuracy of the classifiers learned in each view by bootstrapping the views from each other.

In multi-view learning, an example x is described by a different set of features in each view. For example, in a domain with k views $V_1, V_2, \ldots V_k$, a labeled example can be seen as a tuple $\langle x_1, x_2, \ldots, x_k, l \rangle$, where l is its label, and $x_1, x_2, \ldots, x_k$ are its descriptions in the k views. Similarly, a k-view unlabeled example is denoted by $\langle x_1, x_2, \ldots, x_k, ? \rangle$. For any example x, $V_i(x)$ denotes the descriptions x_i of x in V_i. Similarly, $V_i(L)$ consists of the descriptions in V_i of all the examples in L.

3. Background on Active and Multi-view Learning

Active learning can be seen as a natural development from the earlier work on optimum experimental design (Fedorov, 1972). In the early 1980s, the machine learning community started recognizing the advantages of inductive systems that are capable of querying their instructors. For example, in order to detect errors in Prolog programs, the Algorithmic Debugging System (Shapiro, 1981, 1982) was allowed to ask the user several types of queries. Similarly, concept learning systems such as Marvin (Sammut & Banerji, 1986) and CAT (Gross, 1991) used queries as an integral part of their respective learning strategies.

Our literature review below is structured as follows. First, we discuss the early, mostly theoretical results on *query construction*. Then we focus on selective sampling algorithms, which select as the next query one of the unlabeled examples from the working set. Finally, we conclude by reviewing the existing multi-view learning algorithms.

3.1 Active Learning by Query Construction

The earliest approaches to formalizing active learning appeared in the seminal papers of Angluin (1982, 1988) and Valiant (1984), who focused on exact concept induction and learning in the PAC framework, respectively. This theoretic work focused on learning classes of concepts such as regular sets, monotone DNF expressions, and $\mu-$expressions. Besides *membership queries* such as "is this an example of the target concept?," Angluin also used more sophisticated types of queries such as *equivalence queries* ("is this concept equivalent with the target concept?") or *superset queries* ("is this concept a superset of the target concept?").

These early active learners took a *constructive* approach to query generation in the sense that each query is (*artificially*) constructed by setting the values of the attributes so that the query is as informative as possible. In practice, this may raise some serious problems; for example, consider a hand-writing recognizer that must discriminate between the 10 digits (Lang & Baum, 1992). In this scenario, an informative query may consist of an image that represents a "fusion" of two similarly-looking digits, such as "3" and "5." When presented with such an image, a user cannot label it properly because it does not represent a recognizable digit. Consequently, a query is "wasted" on a totally irrelevant image. Similar situations appear in many real world tasks such as text classification, information extraction, or speech recognition: whenever the active learner artificially builds a query for such a domain, it is highly unlikely that the newly created object has any meaning for the human user.

Despite this practical applicability issue, the constructive approach to active learning leads to interesting theoretical insights about the merits of various types of queries. For example, researchers considered learning with:

- *incomplete queries*, for which the query's answer may be "`I don't know.`" (Angluin & Slonim, 1991; Goldman & Mathias, 1992; Sloan & Turan, 1994; Blum, Chalasani, Goldman, & Slonim, 1998);

- *malicious queries*, for which the answer to the queries may be erroneous (Angluin, Krikis, Sloan, & Turan, 1997; Angluin & Krikis, 1994; Angluin, 1994).

New learning problems were also considered, from unrestricted DNF expressions (Jackson, 1994; Blum, Furst, Jackson, Kearns, Mansour, & Rudich, 1994) and unions of boxes

(Goldberg, Goldman, & Mathias, 1994) to tree patterns (Amoth, Cull, & Tadepalli, 1998, 1999) and Horn clauses (Reddy & Tadepalli, 1997). Researchers also reported results on applying active learning to neural networks (Hwang, Choi, Oh, & Marks, 1991; Baum, 1991; Watkin & Rau, 1992; Hasenjager & Ritter, 1998) and for combining declarative bias (prior knowledge) and active learning (Tadepalli, 1993; Tadepalli & Russell, 1998).

3.2 Selective Sampling

Selective sampling represents an alternative active learning approach. It typically applies to *classification* tasks in which the learner has access to a large number of unlabeled examples. In this scenario, rather than constructing an informative query, the active learner asks the user to label one of the existing unlabeled examples. Depending on the *source* of unlabeled examples, there are two main types of sampling algorithms: stream- and pool- based. The former assumes that the active learner has access to an (infinite) stream of unlabeled examples (Freund, Seung, Shamir, & Tishby, 1997; Argamon-Engelson & Dagan, 1999; Dagan & Engelson, 1995); as successive examples are presented to it, the active learner must decide which of them should be labeled by the user. In contrast, in the pool-based scenario (Lewis & Gale, 1994; Lewis & Catlett, 1994; McCallum & Nigam, 1998; Muslea, Minton, & Knoblock, 2000, 2002a), the learner is presented with a *working set* of unlabeled examples; in order to make a query, the active learner goes through the entire pool and selects the example to be labeled next.

Based on the criterion used to select the next query, selective sampling algorithms fall under three main categories:

- *uncertainty reduction*: the system queries the example on which the current hypothesis makes the least confident prediction;

- *expected-error minimization*: the system queries the example that maximizes the *expected* reduction in classification error;

- *version space reduction*: the system queries the example that, once labeled, removes as much as possible of the version space.

The *uncertainty reduction* approach to selective sampling works as follows: first, one uses the labeled examples to learn a classifier; then the system queries the unlabeled example on which this classifier makes the *least confident* prediction. This straightforward idea can be applied to any base learner for which one can reliably estimate the confidence of its predictions. Confidence-estimation heuristics were proposed for a variety of base learners such as logistic regression (Lewis & Gale, 1994; Lewis & Catlett, 1994), partially hidden Markov Models (Scheffer & Wrobel, 2001), support vector machines (Schohn & Cohn, 2000; Campbell, Cristianini, & Smola, 2000), and inductive logic programming (Thompson, Califf, & Mooney, 1999).

The second, more sophisticated approach to selective sampling, *expected-error minimization*, is based on the *statistically optimal* solution to the active learning problem. In this scenario, the intuition is to query the unlabeled example that minimizes the error rate of the (future) classifier on the test set. Even though for some (extremely simple) base learners one can find such optimal queries (Cohn, Ghahramani, & Jordan, 1996), this is not true for most inductive learners. Consequently, researchers proposed methods to *estimate* the error reduction for various types of base learners. For example, Roy and McCallum (2001) use a sample estimation method for the Naive Bayes classifier; similar approaches were also described for parameter learning in Bayesian nets (Tong & Koller, 2000) and for nearest neighbor classifiers (Lindenbaum, Markovitch, & Rusakov, 2004).

The heuristic approach to *expected-error minimization* can be summarized as follows. First, one chooses a *loss function* (Roy & McCallum, 2001) that is used to estimate the future error rate. Then each unlabeled example x in the working set is considered as the possible next query, and the system estimates the expected reduction of the error rate for each possible label that x may take. Finally, the system queries the unlabeled example that leads to the largest estimated reduction in the error rate.

Finally, a typical *version space reduction* active learner works as follows: it generates a *committee* of several hypotheses, and it queries the unlabeled examples on which the disagreement within the committee is the greatest. In a two-class learning problem, this strategy translates into making queries that remove approximately half of the version space. Depending on the method used to generate the committee, one can distinguish several types of active learners:

- Query-by-Committee selects a committee by randomly sampling hypotheses from the version space. Query-by-Committee was applied to a variety of base learners such as perceptrons (Freund et al., 1997), Naive Bayes (McCallum & Nigam, 1998), and Winnow (Liere & Tadepalli, 1997). Furthermore, Argamon-Engelson and Dagan (1999, 1995) introduce an extension to Query-by-Committee for Bayesian learning. In the Bayesian framework, one can create the committee by sampling classifiers according to their posterior distributions; that is, the better a hypothesis explains the training data, the more likely it is to be sampled. The main limitation of Query-by-Committee is that it can be applied only to base learners for which it is feasible to randomly sample hypotheses from the version space.

- SG-net (Cohn et al., 1994) creates a 2-hypothesis committee that consists of a "*most-general*" and a "*most-specific*" classifier. These two hypotheses are generated by *modifying the base learner* so that it learns a classifier that labels as many as possible of the unlabeled examples in the working set as positive or negative, respectively. This approach has an obvious drawback: it requires the user to modify the base learner so that it can generate "*most-general*" and "*most-specific*" classifiers.

- Active-Decorate (Melville & Mooney, 2004) can be seen as both a generalization and an improvement of SG-net. It generates a large and diverse committee by successively augmenting the original training set with additional sets of artificially-generated examples. More precisely, it generates artificial examples in keeping with the distribution of the instance space; then it applies the current committee to each such example, and it labels the artificial example with the label that contradicts most of the committee's predictions. A new classifier is learned from this augmented dataset, and then the entire process is repeated until the desired committee size is reached. Active-Decorate was successfully used for domains with nominal and numeric features, but it is unclear how it could be applied to domains such as text classification or extraction, where generating the artificial examples may be problematic.

- Query-by-Bagging and Query-by-Boosting (Abe & Mamitsuka, 1998) create the committee by using the well-known bagging (Breiman, 1996) and boosting (Schapire, 1990) algorithms, respectively. These algorithms were introduced for the c4.5 base learner, for which both bagging and boosting are known to work extremely well.

In general, committee-based sampling tends to be associated with the *version space reduction* approach. However, for base learners such as support vector machines, one can use a *single hypothesis* to make queries that remove (approximately) half of the version space (Tong & Koller, 2001). Conversely, committee-based sampling can also be seen as relying on the *uncertainty reduction* principle: after all, the unlabeled example on which the disagreement within the committee is the greatest can be also seen as the example that has the least certain classification.

3.3 Multi-view, Semi-supervised Learning

As already mentioned, Blum and Mitchell (1998) provided the first formalization of learning in the multi-view framework. Previously, this topic was largely ignored, though the idea clearly shows up in applications such as word sense disambiguation (Yarowsky, 1995) and speech recognition (de Sa & Ballard, 1998). Blum and Mitchell proved that two independent, compatible views can be used to pac-learn (Valiant, 1984) a concept based on few labeled and many unlabeled examples. They also introduced Co-Training, which is the first general-purpose, multi-view algorithm.

Collins and Singer (1999) proposed a version of Co-Training that is biased towards learning hypotheses that predict the same label on most of the unlabeled examples. They introduce an explicit objective function that measures the compatibility of the learned hypotheses and use a boosting algorithm to optimize this objective function. In a related paper (Dasgupta, Littman, & McAllester, 2001), the authors provide pac-like guarantees for this novel Co-Training algorithm (the assumption is, again, that the views are both independent and compatible). Intuitively, Dasgupta et al. (2001) show that the ratio of contention points to unlabeled examples is an upper-bound on the error rate of the classifiers learned in the two views.

Abney (2002) extends the work of Dasgupta et al. by relaxing the view independence assumption. More precisely, the author shows that even with views that are *weakly dependent*, the ratio of contention points to unlabeled examples still represents an upper-bound on the two view's error rate. Unfortunately, this paper introduces just a theoretical definition for the *weak dependence* of the views, without providing an intuitive explanation of its practical consequences.

Researchers proposed two main types of extensions to the original Co-Training algorithm: modifications of the actual algorithm and changes aiming to extend its practical applicability. The former cover a wide variety of scenarios:

- Co-EM (Nigam & Ghani, 2000; Brefeld & Scheffer, 2004) uses Expectation Maximization (Dempster, Laird, & Rubin, 1977) for multi-view learning. Co-EM can be seen as the closest implementation of the theoretical framework proposed by Blum and Mitchell (1998).

- Ghani (2002) uses Error-Correcting Output Codes to allow Co-Training and Co-EM to scale up to problems with a large number of classes.

- Corrected Co-Training (Pierce & Cardie, 2001) asks the user to manually correct the labels of the bootstrapped examples. This approach is motivated by the observation that the quality of the bootstrapped data is crucial for Co-Training's convergence.

- Co-Boost (Collins & Singer, 1999) and Greedy Agreement (Abney, 2002) are Co-Training algorithms that explicitly aim to minimize the number of contention points.

The second group of extensions to Co-Training is motivated by the fact that, in practice, one also encounters many problems for which there is no straightforward way to split the features in two views. In order to cope with this problem, Goldman and Zhou (2000) advocate the use of *multiple biases* instead of multiple views. The authors introduce an algorithm similar to Co-Training, which bootstraps from each other hypotheses learned by two different base learners; this approach relies on the assumption that the base learners generate hypotheses that partition the instance space into equivalence classes. In a recent paper, Zhou and Goldman (2004) use the idea of a multi-biased committee for active learning; i.e., they use various types base learners to obtain a diverse committee, and then query the examples on which this committee disagree the most.

Within the multi-view framework, Nigam and Ghani (2000) show that, for "bag-of-words" text classification, one can create two views by arbitrarily splitting the original set of features into two sub-sets. Such an approach fits well the text classification domain, in which the features are abundant, but it is unlikely to work on other types of problems. An alternative solution is proposed by Raskutti, Ferra, and Kowalczyk (2002), where the authors create a second view that consists of a variety of features that measure the examples' similarity with the N largest clusters in the domain. Finally, Muslea, Minton, and Knoblock (2002b) propose a meta-learning approach that uses past experiences to predict whether the given views are appropriate for a new, unseen learning task.

4. The Co-Testing Family of Algorithms

In this section, we discuss in detail the Co-Testing family of algorithms. As we already mentioned, Co-Testing can be seen as a two-step iterative process: first, it uses a few labeled examples to learn a hypothesis in each view; then it queries an unlabeled example for which the views predict different labels. After adding the queried example to the training set, the entire process is repeated for a number of iterations.

The remainder of this section is organized as follows: first, we formally present the Co-Testing family of algorithms and we discuss several of its members. Then we introduce the concepts of strong and weak views, and we analyze how Co-Testing can exploit both types of views (previous multi-view learners could only use strong views). Finally, we compare and contrast Co-Testing to the related approaches.

4.1 The Family of Algorithms

Table 1 provides a formal description on the Co-Testing family of algorithms. The input consists of k views $V_1, V_2, \ldots, V_k$, a base learner $\mathcal{L}$, and the sets L and U of labeled and

Table 1: The Co-Testing family of algorithms: repeatedly learn a classifier in each view and query an example on which they predict different labels.

Given:

- a base learner $\mathcal{L}$
- a learning domain with features $V = \{a_1, a_2, \ldots, a_N\}$
- k views $V_1, V_2, \ldots, V_k$ such that $V = \bigcup_{i=1}^{k} V_i$ and $\forall i, j \in \{1, 2, \ldots, k\}, i \neq j, V_i \cap V_j = \emptyset$
- the sets L and U of labeled and unlabeled examples, respectively
- number N of queries to be made

- `LOOP` for N iterations
  - use $\mathcal{L}$ to learn the classifiers $h_1, h_2, \ldots, h_k$ in the views $V_1, V_2, \ldots, V_k$, respectively
  - let $ContentionPoints = \{ \ \langle x_1, x_2, \ldots, x_k, ?\rangle \in U \mid \exists i, j \ h_i(x_i) \neq h_j(x_j) \ \}$
  - let $\langle x_1, x_2, \ldots, x_k, ?\rangle =$ **SelectQuery**$(ContentionPoints)$
  - remove $\langle x_1, x_2, \ldots, x_k, ?\rangle$ from U and ask for its label l
  - add $\langle x_1, x_2, \ldots, x_k, l\rangle$ to L

- $h_{OUT} =$ **CreateOutputHypothesis**($h_1, h_2, \ldots, h_k$)

unlabeled examples, respectively. Co-Testing algorithms work as follows: first, they learn the classifiers $h_1, h_2, \ldots, h_k$ by applying the algorithm $\mathcal{L}$ to the projection of the examples in L onto each view. Then they apply $h_1, h_2, \ldots, h_k$ to all unlabeled examples in U and create the set of contention points, which consists of all unlabeled examples for which at least two of these hypotheses predict a different label. Finally, they query one of the contention points and then repeat the whole process for a number of iterations. After making the allowed number of queries, Co-Testing creates an *output hypothesis* that is used to make the actual predictions.

The various members of the Co-Testing family differ from each other with two respects: the strategy used to select the next query, and the manner in which the output hypothesis is constructed. In other words, each Co-Testing algorithm is uniquely defined by the choice of the functions **SelectQuery()** and **CreateOutputHypothesis()**.

In this paper we consider three types of query selection strategies:

- *naive*: choose at random one of the contention points. This straightforward strategy is appropriate for base learners that lack the capability of reliably estimating the confidence of their predictions. As this naive query selection strategy is independent of both the domain and the base learner properties, it follows that it can be used for solving any multi-view learning task.

- *aggressive*: choose as query the contention point Q on which the least confident of the hypotheses $h_1, h_2, \ldots, h_k$ makes the most confident prediction; more formally,

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

Aggressive Co-Testing is designed for high accuracy domains, in which there is little or no noise. On such domains, discovering unlabeled examples that are misclassified "with high confidence" translates into queries that remove significantly more than half of the version space.

- *conservative*: choose the contention point on which the confidence of the predictions made by $h_1, h_2, \ldots, h_k$ is as close as possible (ideally, they would be equally confident in predicting different labels); that is,

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

Conservative Co-Testing is appropriate for noisy domains, where the *aggressive* strategy may end up querying mostly noisy examples.

Creating the output hypothesis also allows the user to choose from a variety of alternatives, such as:

- *weighted vote*: combines the vote of each hypothesis, weighted by the confidence of their respective predictions.

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

- *majority vote*: Co-Testing chooses the label that was predicted by most of the hypotheses learned in the k views.

$$h_{OUT}(x) = \underset{l \in Labels}{\arg\max} \sum_{\substack{g \in \{h_1,\ldots,h_k\} \\ g(x) = l}} 1$$

This strategy is appropriate when there are at least three views, and the base learner cannot reliably estimate the confidence of its predictions.

- *winner-takes-all*: the output hypothesis is the one learned in the view that makes the smallest number of mistakes over the N queries. This is the most obvious solution for 2-view learning tasks in which the base learner cannot (reliably) estimate the confidence of its predictions. If we denote by $Mistakes(h_1), Mistakes(h_2), \ldots, Mistakes(h_k)$ the number of mistakes made by the hypotheses learned in the k views on the N queries, then

$$h_{OUT}(x) = \underset{g \in \{h_1,\ldots,h_k\}}{\arg\min} Mistakes(g)$$

4.2 Learning with Strong and Weak Views

In the original multi-view setting (Blum & Mitchell, 1998; Muslea et al., 2000), one makes the *strong views* assumption that each view is sufficient to learn the target concept. However, in practice, one also encounters views in which one can accurately learn only a concept that is strictly *more general* or *more specific* than the concept of interest (Muslea, Minton, & Knoblock, 2003). This is often the case in domains that involve *hierarchical* classification, such as information extraction or email classification. For example, it may be extremely easy to discriminate (with a high accuracy) between work and personal emails based solely on the email's sender; however, this same information may be insufficient for predicting the work or personal *sub-folder* in which the email should be stored.

We introduce now the notion of a *weak view*, in which one can accurately learn only a concept that is strictly *more general* or *more specific* than the target concept. Note that learning in a weak view is qualitatively different from learning "an approximation" of the target concept: the latter represents learning with imperfect features, while the former typically refers to a (easily) learnable concept that is a strict generalization/specialization of the target concept (note that, in the real world, imperfect features and noisy labels affect learning in both strong and weak views).

In the context of learning with strong and weak views, we redefine *contention points* as the unlabeled examples on which the *strong views* predict a different label. This is a necessary step because of two reasons: first, as the weak view is inadequate for learning the target concept, it typically disagrees with the strong views on a large number of unlabeled examples; in turn, this would increase the number of contention points and skew their distribution. Second, we are not interested in fixing the mistakes made by a weak view, but rather in using this view as an additional information source that allows faster learning in the strong views (i.e., from fewer examples).

Even though the weak views are inadequate for learning the target concept, they can be exploited by Co-Testing both in the **SelectQuery()** and **CreateOutputHypothesis()** functions. In particular, weak views are extremely useful for domains that have only two strong views:

- the weak view can be used in **CreateOutputHypothesis()** as a *tie-breaker* when the two strong views predict a different label.

- **SelectQuery()** can be designed so that, ideally, each query would represent a mistake in both strong views. This can be done by first detecting the contention points - if any - on which the weak view disagrees with both strong views; among these, the next query is the one on which the weak view makes the most confident prediction. Such queries are likely to represent a mistake in *both* strong views, rather than in just one of them; in turn, this implies simultaneous large cuts in *both* strong version spaces, thus leading to faster convergence.

In section 5.2.2 we describe a Co-Testing algorithm that exploits strong and weak views for wrapper induction domains (Muslea et al., 2003; Muslea, Minton, & Knoblock, 2001). Note that learning from strong and weak views clearly extends beyond wrapper induction tasks: for example, the idea of exploiting complementary information sources (i.e., different types of features) appears in the two multi-strategy learners (Kushmerick, Johnston, & McGuinness, 2001; Nahm & Mooney, 2000) that are discussed in section 4.3.2.

4.3 Co-Testing *vs.* Related Approaches

As we already mentioned in section 3.3, existing multi-view approaches are typically semi-supervised learners that bootstrap the views from each other. The two exceptions (Muslea et al., 2002a; Jones, Ghani, Mitchell, & Riloff, 2003) interleave Co-Testing and Co-EM (Nigam & Ghani, 2000), thus combining the best of both worlds: semi-supervised learning provides the active learner with more accurate hypotheses, which lead to more informative queries; active learning provides the semi-supervised learner with a more informative training set, thus leading to faster convergence.

4.3.1 Co-Testing *vs.* Existing Active Learners

Intuitively, Co-Testing can be seen as a committee-based active learner that generates a committee that consists of one hypothesis in each view. Also note that Co-Testing can be combined with virtually any of the existing single-view active learners: among the contention points, Co-Testing can select the next query based on any of the heuristics discussed in section 3.2.

There are two main differences between Co-Testing and other active learners:

- except for Co-Testing and its variants (Muslea et al., 2002a; Jones et al., 2003), all other active learners work in the single-view framework (i.e., they pool together all the domain features).
- single-view active learners are typically designed for a particular (class of) base learner(s). For example, Query-by-Committee (Seung, Opper, & Sompolinski, 1992) assumes that one can randomly sample hypotheses from the version space, while Uncertainty Sampling (Lewis & Gale, 1994; Lewis & Catlett, 1994) relies on the base learner's ability to reliably evaluate the confidence of its predictions. In contrast, the basic idea of Co-Testing (i.e., querying contention points) applies to any multi-view problem, independently of the base learner to be used.

The Co-Testing approach to active learning has both advantages and disadvantages. On one hand, Co-Testing cannot be applied to problems that do not have at least two views. On the other hand, for any multi-view problem, Co-Testing can be used with the best base learner for that particular task. In contrast, in the single-view framework, one often must either create a new active learning method for a particular base learner or, even worse, modify an existing base learner so that it can be used in conjunction with an existing sampling algorithm.

To illustrate this last point, let us briefly consider learning for *information extraction*, where the goal is to use machine learning for extracting relevant strings from a collection of documents (e.g., extract the perpetrators, weapons, and victims from a corpus of news stories on terrorist attacks). As information extraction is different in nature from a typical classification task, existing active learners cannot be applied in a straightforward manner:

- for *information extraction from free text* (IE), the existing active learners (Thompson et al., 1999; Soderland, 1999; Scheffer & Wrobel, 2001) are crafted based on heuristics specific to their respective base learners, rapier, whisk, and Partially Hidden Markov Models. An alternative is discussed by Finn and Kushmerick (2003), who explore a variety of IE-*specific* heuristics that can be used for active learning purposes and analyze the trade-offs related to using these heuristics.

- for *wrapper induction*, where the goal is to extract data from Web pages that share the same underlying structure, there are no reported results for applying (single-view) active learning. This is because typical wrapper induction algorithms (Muslea et al., 2001; Kushmerick, 2000; Hsu & Dung, 1998) are base learners that lack the properties exploited by the single-view active learners reviewed in section 3.2.: they are *determinist* learners that are *noise sensitive*, provide *no confidence* in their predictions, and make *no mistakes* on the training set.

In contrast, Co-Testing applies naturally to both wrapper induction (Muslea et al., 2000, 2003) and information extraction from free text (Jones et al., 2003). This is due to the fact that Co-Testing does *not* rely on the base learner's properties to identify its highly informative set of candidate queries; instead, it focuses on the contention points, which, by definition, are *guaranteed* to represent mistakes in some of the views.

4.3.2 Exploiting Weak Views

We briefly discuss now two learning tasks that can be seen as learning from strong and weak views, even though they were not formalized as such, and the views were not used for active learning. An additional application domain with strong and weak views, wrapper induction, is discussed at length in section 5.2.

The discotex (Nahm & Mooney, 2000) system was designed to extract job titles, salaries, locations, etc from computer science job postings to the newsgroup `austin.jobs`. discotex proceeds in four steps: first, it uses rapier (Califf & Mooney, 1999) to learn extraction rules for each item of interest. Second, it applies the learned rules to a large, unlabeled corpus of job postings and creates a database that is populated with the extracted data. Third, by text mining this database, discotex learns to predict the value of each item based on the values of the other fields; e.g., it may discover that "`IF the job requires` C++ `and` CORBA `THEN the development platforms include Windows`." Finally, when the system is deployed and the rapier rules fail to extract an item, the mined rules are used to predict the item's content.

In this scenario, the rapier rules represent the *strong view* because they are sufficient for extracting the data of interest. In contrast, the mined rules represent the *weak view* because they cannot be learned or used by themselves. Furthermore, as discotex discards all but the most accurate of the mined rules, which are highly-specific, it follows that this weak view is used to learn concepts that are *more specific* than the target concept. Nahm and Mooney (2000) show that the mined rules improve the extraction accuracy by capturing information that complements the rapier extraction rules.

Another domain with strong and weak views is presented by Kushmerick et al. (2001). The learning task here is to classify the lines of text on a business card as a person's name, affiliation, address, phone number, etc. In this domain, the strong view consists of the words that appear on each line, based on which a Naive Bayes text classifier is learned. In the weak view, one can exploit the relative order of the lines on the card by learning a

Hidden Markov Model that predicts the probability of a particular ordering of the lines on the business card (e.g., name followed by address, followed by phone number).

This weak view defines a class of concepts that is *more general* than the target concept: all line orderings are possible, even though they are not equally probable. Even though the order of the text lines cannot be used by itself to accurately classify the lines, when combined with the strong view, the ordering information leads to a classifier that clearly outperforms the stand-alone strong view (Kushmerick et al., 2001).

Note that both approaches above use the strong and weak views for passive, rather than active learning. That is, given a fixed set of labeled and no unlabeled examples, these algorithms learn one weak and one strong hypothesis that are then used to craft a *domain-specific* predictor that outperforms each individual hypothesis. In contrast, Co-Testing is an active learner that seamlessly integrates weak and strong hypotheses without requiring additional, domain-specific data engineering.

5. Empirical Validation

In this section we empirically compare Co-Testing with other state of the art learners. Our goal is to test the following hypothesis: given a multi-view learning problem, Co-Testing converges faster than its single-view counterparts.

We begin by presenting the results on three real-world *classification* domains: Web-page classification, discourse tree parsings, and advertisement removal. Then we focus on an important industrial application, *wrapper induction* (Muslea et al., 2001; Kushmerick, 2000; Hsu & Dung, 1998), in which the goal is to learn rules that extract the relevant data from a collection of documents (e.g., extract book titles and prices from a Web site).

The results for classification and wrapper induction are analyzed separately because:

- for each of the three classification tasks, there are only two strong views that are available; in contrast, for wrapper induction we have two strong and one weak views, which allows us to explore a wider range of options.

- for each classification domain, there is exactly one available dataset. In contrast, for wrapper induction we use a testbed of 33 distinct tasks. This imbalance in the number of available datasets requires different presentation styles for the results.

- in contrast to typical classification, a major requirement for wrapper induction is to learn (close to) 100%-accurate extraction rules from just a handful of examples (Muslea, 2002, pages 3-6). This requirement leads to significant differences in both the experimental setup and the interpretation of the results (e.g., results that are excellent for most classification tasks may be unacceptable for wrapper induction).

5.1 Co-Testing for Classification

We begin our empirical study by using three classification tasks to compare Co-Testing with existing active learners. We first introduce these three domains and their respective views; then we discuss the learners used in the evaluation and analyze the experimental results.

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* / *conservative*	*weighted vote*	√	√	√	√	√

Table 2: The algorithms used for classification. The last five columns denote Query-by-Committee/-Bagging/-Boosting, Uncertainty Sampling and Random Sampling.

5.1.1 The Views used by Co-Testing

We applied Co-Testing to three real-world classification domains for which there is a natural, intuitive way to create two views:

- AD (Kushmerick, 1999) is a classification problem with two classes, 1500 attributes, and 3279 examples. In AD, images that appear in Web pages are classified into `ads` and `non-ads`. The view V_1 consists of all textual features that describe the image; e.g., 1-grams and 2-grams from the caption, from the URL of the page that contains the image, from the URL of the page the image points to, etc. In turn, V_2 describes the properties of the image itself: length, width, aspect ratio, and "origin" (i.e., are the image and the page that contains it coming from the same Web server?).

- COURSES (Blum & Mitchell, 1998) is a domain with two classes, 2206 features, and 1042 examples. The learning task consists of classifying Web pages as `course homepages` and `other pages`. In COURSES the two views consist of words that appear in the page itself and words that appear in hyperlinks pointing to them, respectively.

- TF (Marcu, Carlson, & Watanabe, 2000) is a classification problem with seven classes, 99 features and 11,193 examples. In the context of a machine translation system, it uses the shift-reduce parsing paradigm to learn how to rewrite Japanese discourse trees as English-like discourse trees. In this case, V_1 uses features specific to the shift-reduce parser: the elements in the input list and the partial trees in the stack. V_2 consists of features specific to the Japanese tree given as input.

5.1.2 The Algorithms used in the Evaluation

Table 2 shows the learners used in this empirical comparison. We have implemented all active learners as extensions of the $\mathcal{MLC}$++ library (Kohavi, Sommerfield, & Dougherty, 1997). For each domain, we choose the base learner as follows: after applying all primitive learners in $\mathcal{MLC}$++ on the dataset (10-fold cross-validation), we select the one that obtains the best performance. More precisely, we are using the following base learners: IB (Aha, 1992) for AD, Naive Bayes (Blum & Mitchell, 1998) for COURSES, and MC4, which is an implementation of C4.5, for TF.

The five single-view algorithms from Table 2 use all available features (i.e., $V_1 \cup V_2$) to learn the target concept.[1] On all three domains, Random Sampling (**Rnd**) is used as strawman; Query-by-Bagging and -Boosting, denoted by **qBag** and **qBst**, are also run on all three domains. In contrast, Uncertainty Sampling (**US**) is applied only on AD and COURSES because MC4, which is the base learner for TF, does not provide an estimate of the confidence of its prediction.

As there is no known method for *randomly* sampling from the IB or MC4 version spaces, Query-by-Committee (**QBC**) is not applied to AD and TF. However, we apply **QBC** to COURSES by borrowing an idea from McCallum and Nigam (1998): we create the committee by sampling hypotheses according to the (Gamma) distribution of the Naive Bayes parameters estimated from the training set L.

For Query-by-Committee we use a typical 2-hypothesis committee. For Query-by-Bagging and -Boosting, we use a relatively small 5-hypothesis committees because of the CPU constraints: the running time increases linearly with the number of learned hypotheses, and, in some domains, it takes more than 50 CPU hours to complete the experiments even with the 5-hypothesis committees.

Because of the limitations of their respective base learners (i.e., the above-mentioned issue of estimating the confidence in each prediction), for AD and TF we use Naive Co-Testing with a *winner-takes-all* output hypothesis; that is, each query is randomly selected among the contention points, and the output hypothesis is the one learned in the view that makes the fewest mistakes on the queries. In contrast, for COURSES we follow the methodology from the original Co-Training paper (Blum & Mitchell, 1998), where the output hypothesis consists of the *weighted vote* of the classifiers learned in each view.

On COURSES we investigate two of the Co-Testing query selection strategies: *naive* and *conservative*. The third, *aggressive* query selection strategy is not appropriate for COURSES because the "hyperlink view" is significantly less accurate than the other one (after all, one rarely encounters more than a handful of words in a hyperlink). Consequently, most of the high-confidence contention points are "unfixable mistakes" in the hyperlink view, which means that even after seeing the correct label, they cannot be classified correctly in that view.

5.1.3 The Experimental Results

The learners' performance is evaluated based on 10-fold, stratified cross validation. On AD, each algorithm starts with 150 randomly chosen examples and makes 10 queries after each of the 40 learning episodes, for a total of 550 labeled examples. On COURSES, the algorithms start with 6 randomly chosen examples and make one query after each of the 175 learning episodes. Finally, on TF the algorithms start with 110 randomly chosen examples and make 20 queries after each of the 100 learning episodes.

Figures 1 and 2 display the learning curves of the various algorithms on AD, TF, and COURSE. On all three domains, Co-Testing reaches the highest accuracy (i.e., smallest error rate). Table 3 summarizes the statistical significance results (`t`-test confidence of at least

1. In a preliminary experiment, we have also ran the algorithms of the individual views. The results on V_1 and V_2 were either worse then those on $V_1 \cup V_2$ or the differences were statistically insignificant. Consequently, for sake of simplicity, we decided to show here only the single-view results for $V_1 \cup V_2$.

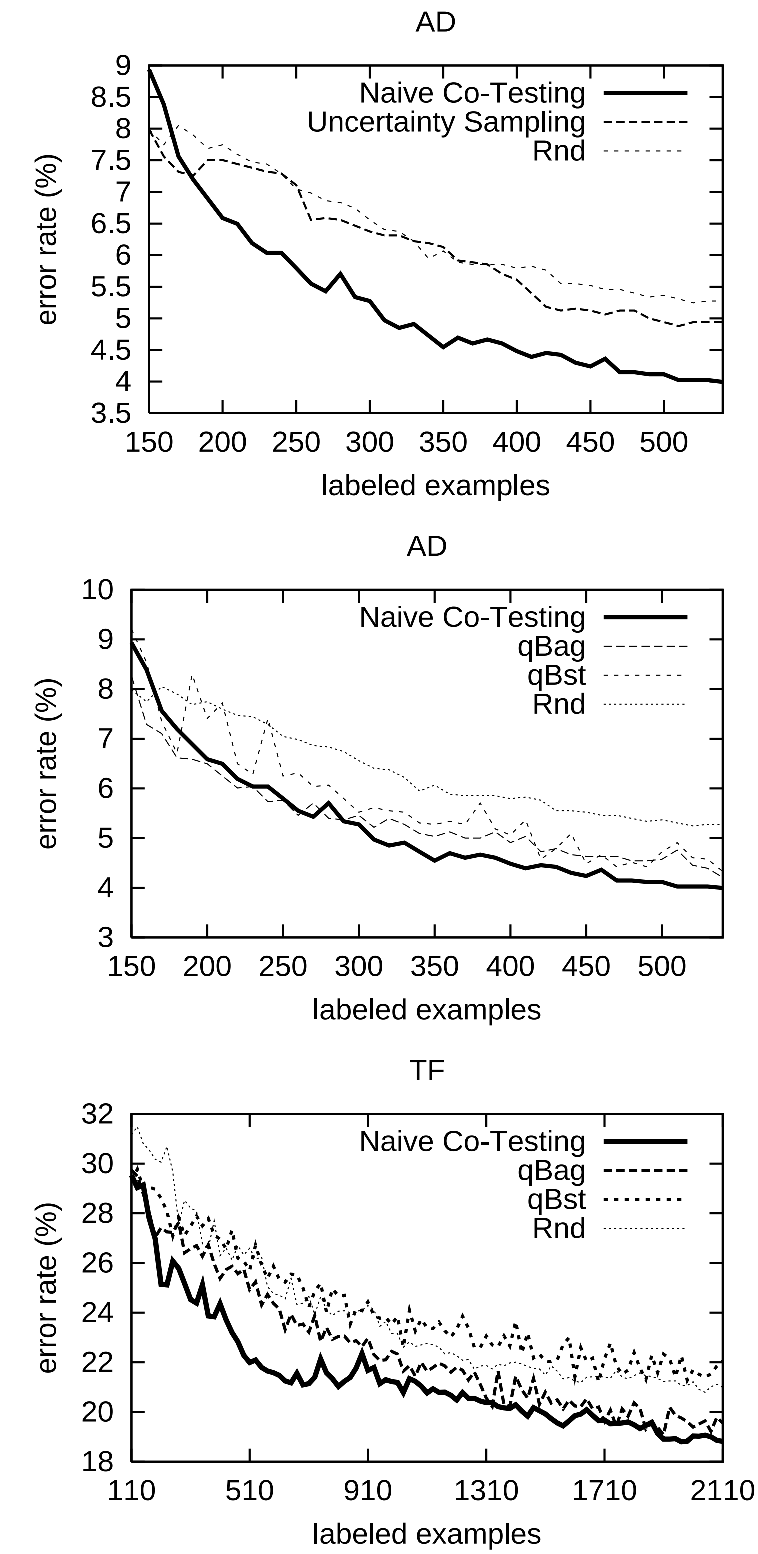

Figure 1: Empirical results on the AD and TF problems

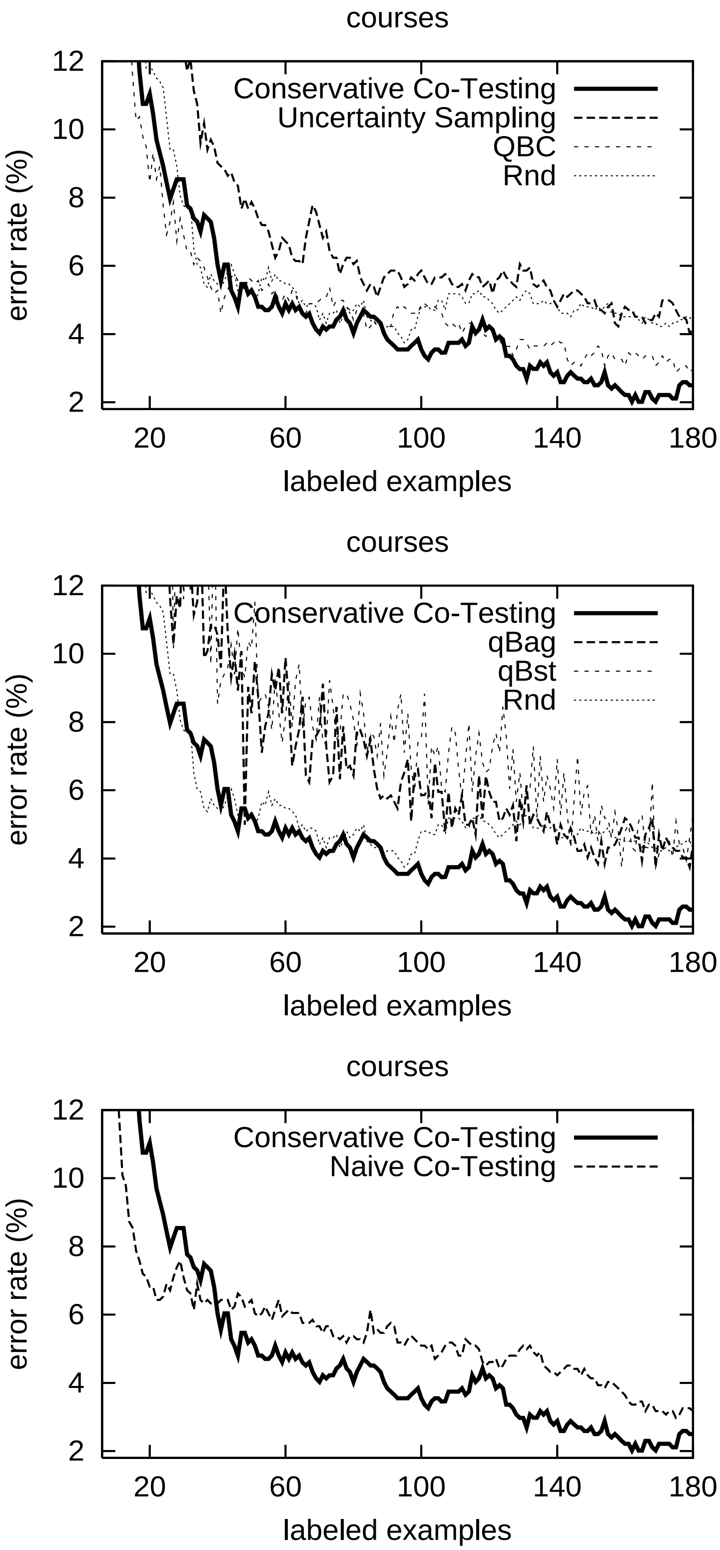

Figure 2: Empirical results on the COURSES problem

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

Table 3: Statistical significance results in the empirical (pair-wise) comparison of the various algorithms on the three domains.

95%) obtained in a pair-wise comparison of the various algorithms. These comparisons are performed on the right-most half of each learning curve (i.e., towards convergence). The best way to explain the results in Table 3 is via examples: the results of comparing Naive Co-Testing and Random Sampling on AD appear in the first three columns of the first row. The three numbers (i.e., 0, 0, and 19) mean that on (all) 19 comparison points Naive Co-Testing outperforms Random Sampling in a statistically significant manner. Similarly, comparing Naive and Conservative Co-Testing on COURSES (the last three columns on the last row) leads to the following results: on 28 of the comparison points Conservative Co-Testing outperforms Naive Co-Testing in a statistically significant manner; on 21 other points the differences are statistically insignificant; finally, on *no* comparison point Naive Co-Testing outperforms its Conservative counterpart.

The results in Table 3 can be summarized as follows. First of all, *no single-view algorithm* outperforms Co-Testing in a statistically significant manner on *any* of the comparison points. Furthermore, except for the comparison with Query-by-Bagging and -Boosting on AD, where the difference in accuracy is statistically insignificant on almost all comparison points, Co-Testing clearly outperform all algorithms on all domains.

Finally, let us briefly comment on applying multi-view, semi-supervised learners to the three tasks above. As mentioned in section 3.3, such algorithms bootstrap the views from each other by training each view on the examples labeled with high-confidence by the other view. For AD and TF, we could not use multi-view, semi-supervised learning because the base learners IB and MC4 do not provide a (reliable) estimate of the confidence in their predictions. More precisely, MC4 provides no estimate at all, while IB's estimates are extremely poor when the training data is scarce (e.g., see the poor performance of Uncertainty Sampling on AD, where it barely outperforms Random Sampling).

On COURSES, we have applied both Co-Training and Co-EM in conjunction with the Naive Bayes base learner. Both these multi-view learners reach their maximum accuracy (close to 95%) based on solely 12 labeled and 933 unlabeled examples, after which their

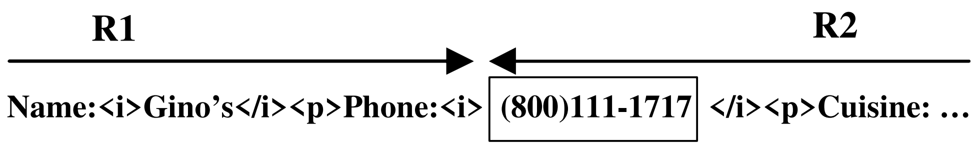

Figure 3: Both the *forward* and *backward* rules detect the beginning of the phone number.

performance does not improve in a statistically significant manner.[2] As shown in Figure 2, when the training data is scarce (i.e., under 40 labeled examples), Co-Testing's accuracy is less than 95%; however, after making additional queries, Co-Testing reaches 98% accuracy, while Co-Training and Co-EM remain at 95% even when trained on 180 labeled and 765 unlabeled examples. These results are consistent with the different goals of active and semi-supervised learning: the former focuses on learning the *perfect* target concept from a minimal amount of labeled data, while the latter uses unlabeled examples to boost the accuracy of a hypothesis learned from just a handful of labeled examples.

5.2 Co-Testing for Wrapper Induction

We focus now on a different type of learning application, *wrapper induction* (Muslea et al., 2001; Kushmerick, 2000), in which the goal is to learn rules that extract relevant sub-strings from a collection of documents. Wrapper induction is a key component of commercial systems that integrate data from a variety of Web-based information sources.

5.2.1 The Views used by Co-Testing

Consider the illustrative task of extracting phone numbers from documents similar to the fragment in Figure 3. To find where the phone number begins,[3] one can use the rule

R1 = *SkipTo*(`Phone:<i>`)

This rule is applied *forward*, from the beginning of the page, and it ignores everything until it finds the string `Phone:<i>`. Note that such forward-going rules do not represent the only way to detect where the phone number begins: an alternative approach is to use the rule

R2 = *BackTo*(`Cuisine`) *BackTo*((*Number*))

which is applied *backward*, from the *end* of the document. **R2** ignores everything until it finds "`Cuisine`" and then, again, skips to the first number between parentheses.

Forward and backward rules such as **R1** and **R2** can be learned from user-provided examples by the state of the art wrapper induction system stalker (Muslea et al., 2001), which we use as base learner for Co-Testing. Intuitively, stalker creates a forward or a

2. A recent paper (Brefeld & Scheffer, 2004) shows that - for text classification - svm is more appropriate than Naive Bayes as base learner for Co-EM, though not necessarily for Co-Training. As the $\mathcal{MLC}$++ library does not provide svm as base learner, we could not compare our results with those by Brefeld and Scheffer (2004), where Co-EM + svm reaches 99% accuracy based on 12 labeled and 933 unlabeled examples. However, in all fairness, it is unlikely that Co-Testing could lead to an even faster convergence.
3. As shown by Muslea et al. (2001), the end of the phone number can be found in a similar manner.

backward rule that consumes all the tokens that precede or follow the extraction point, respectively. It follows that rules such as **R1** and **R2** represent descriptions of the same concept (i.e., beginning of phone number) that are learned in two different views: the sequences of tokens that *precede* and *follow* the beginning of the item, respectively. These views are *strong views* because each of them is sufficient to accurately extract the items of interest (Muslea et al., 2001, 2000).

In addition to these two views, which rely mostly on the *context* of the item to be extracted (i.e., the text surrounding the item), one can use a third view that describes the content of the item to be extracted. For example, phone numbers can be described by the simple grammar: "`(` *Number* `)` *Number* `-` *Number*"; similarly, most URLs start with "`http://www.`", end with "`.html`", and contain no HTML tags.

Such a content-based view is a *weak view* because it often represents a concept *more general* than the target one. For example, the phone number grammar above cannot discriminate between the home, office, cell, and fax numbers that appear within the same Web page; similarly, the URL grammar cannot distinguish between the URLs of interest (e.g., a product's review) and all the other ones (e.g., advertisements).

In this weak view, we use as base learner a version of DataPro (Lerman, Minton, & Knoblock, 2003) that is described elsewhere (Muslea et al., 2003). DataPro learns - from positives examples only - the "prototypes" of the items to be extracted; i.e., it finds statistically significant sequences of tokens that (1) are highly unlikely to have been generated by chance and (2) describe the content of many of the positive examples. The features used by this base learner consist of the *length range* (in tokens) of the seen examples, the *token types* that appear in the training set (e.g., *Number*, *AllCaps*, etc), and the *start*- and *end*-pattern (e.g., "`http://www.`" and "*AlphaNum* `.html`", respectively).

5.2.2 The Algorithms used in the Evaluation

As the extraction rules learned in the two strong views do not provide any estimate of the confidence of the extractions, the only Co-Testing algorithm that can be implemented based solely on the forward and backward views is *Naive Co-Testing* with a *winner-takes-all* output hypothesis:

- each query is randomly chosen (*Naive Co-Testing*) among the contention points, which are the documents from which the learned rules extract different strings.

- the output hypothesis is the rule learned in the view that makes the fewest mistakes over the allowed number of queries (i.e., *winner-takes-all*).

Given the additional, content-based view, we can also implement an *aggressive* version of Co-Testing for wrapper induction:

- the contention points are, again, the unlabeled examples on which the rules learned in the *strong* views do not extract the same string.

- the *aggressive query selection strategy* works by selecting the contention point for which the hypothesis learned in the weak view is maximally confident that *both* STALKER rules are extracting incorrect strings. More formally, for each contention point, let s_1 and s_2 be the strings extracted by the strong views; let us also denote by n_1 and n_2

the number of constraints learned in the weak views that are violated by s_1 and s_2. Using this notation, the next query is the contention point for which $min(n_1, n_2)$ has the largest value.

- the output hypothesis is obtained by the following *majority voting* scheme: if the strings extracted by the strong views are identical, then they represent the extracted item; otherwise the result is the one of these two strings that violates fewer of the constraints learned in the weak view.

In the empirical evaluation below, we compare these two Co-Testing algorithms with Random Sampling and Query-by-Bagging. The former is used as strawman, while the latter is the only general-purpose active learner that can be applied in a straightforward manner to wrapper induction (for details, see the discussion in section 4.3.1). Finally, existing multi-view, semi-supervised learners cannot be used for wrapper induction because the base learners do not provide an estimate in the confidence of each extraction; and even if such an estimate could be obtained, wrapper induction algorithms are extremely sensitive to mislabeled examples, which would make the bootstrapping process unacceptably brittle.

In this paper, the implementation of Random Sampling is identical with that of Naive Co-Testing with *winner takes all*, except that it randomly queries one of the unlabeled examples from the working set. For Query-by-Bagging, the committee of hypotheses is created by repeatedly re-sampling (with substitution) the examples in the original training set L. We use a relatively small committee (i.e., 10 extraction rules) because when learning from a handful of examples, re-sampling with replacement leads to just a few distinct training sets. In order to make a fair comparison with Co-Testing, we run Query-by-Bagging once in each strong view and report the best of the obtained results.

5.2.3 The Experimental Results

In our empirical comparison, we use the 33 most difficult wrapper induction tasks from the testbed introduced by Kushmerick (1998, 2000). These tasks, which were previously used in the literature (Muslea et al., 2003; Muslea, 2002), are briefly described in Table 4. We use 20-fold cross-validation to compare the performance of Naive and Aggressive Co-Testing, Random Sampling, and Query-by-Bagging on the 33 tasks. Each algorithm starts with two randomly chosen examples and then makes 18 successive queries.

The results below can be summarized as follows: for 12 tasks, only the two Co-Testing algorithms learn 100% accurate rules; for another 18 tasks, Co-Testing and *at least* another algorithm reach 100% accuracy, but Co-Testing requires the smallest number of queries. Finally, on the remaining three tasks, no algorithm learns a 100% accurate rule.

Figure 4 shows the aggregate performance of the four algorithms over the 33 tasks. In each of the six graphs, the `X` axis shows the number of queries made by the algorithm, while the `Y` axis shows the number of tasks for which a 100% accurate rule was learned based on exactly `X` queries. As mentioned earlier, all algorithms start with 2 random examples and make 18 additional queries, for a total of 20 labeled examples. *By convention*, the rightmost point on the `X` axis, which is labeled "`19 queries`", represents the number of tasks that require more than the allowed 18 queries to learn a 100% accurate rule. This additional "`19 queries`" data-point allows us to summarize the results without dramatically extending the `X` axis beyond 18 queries: as for some of the extraction tasks Random Sampling and

Task ID	Source name	Item name	Nmb exs
S1-0	Computer ESP	Price	404
S1-1		URL	404
S1-2		Item	404
S2-0	CNN/Time AllPolitics	URL	501
S2-1		Source	501
S2-2		Title	499
S2-3		Date	492
S3-0	Film.com Search	URL	175
S3-1		Name	175
S3-3		Size	175
S3-4		Date	175
S3-5		Time	175
S6-1	PharmaWeb	University	27
S9-10	Internet Travel Net.	Arrival Time	44
S9-11		Availability	39
S11-1	Internet Address	Email	91
S11-2		Update	91

Task ID	Source name	Item name	Nmb exs
S11-3	Finder	Organization	72
S15-1	NewJour	Name	355
S19-1	Shops.Net	Score	201
S19-3		Item Name	201
S20-3	Democratic Party Online	Score	91
S20-5		File Type	328
S24-0	Foreign Languages for Travelers	Language	690
S24-1		URL	424
S24-3		Translation	690
S25-0	US Tax Code	URL	328
S26-3	CD Club Web Server	Price	377
S26-4		Artist	377
S26-5		Album	377
S28-0	Cyberider Cycling WWW	URL	751
S28-1		Relevance	751
S30-1	Congress Quarterly	Person Name	30

Table 4: The 33 wrapper induction tasks used in the empirical evaluation.

Query-by-Bagging need hundreds of queries to learn the correct rules, the histograms would become difficult to read if the entire `X` axis were shown.

As shown in Figure 4, the two Co-Testing algorithms clearly outperform their single-view counterparts, with Aggressive Co-Testing doing significantly better than Naive Co-Testing (the results are statistically significant with a confidence of at least 99%). Aggressive Co-Testing learns 100%-accurate rules on 30 of the 33 tasks; for all these tasks, the extraction rules are learned from at most seven queries. Naive Co-Testing learns 100% accurate rules on 28 of the 33 tasks. On 26 of these 28 tasks, the extraction rules are learned based on at most six queries. In contrast, Random Sampling and Query-by-Bagging learn 100% accurate rules for only seven and twelve of the tasks, respectively. In other words, both Co-Testing algorithms learn the correct target concept for more than twice as many tasks than Query-by-Bagging or Random Sampling.

We must emphasize the power of Aggressive Co-Testing on high-accuracy tasks such as wrapper induction: for 11 of the 33 tasks, a single, "aggressively-chosen" query is sufficient to learn the correct extraction rule. In contrast, Naive Co-Testing converges in a single query on just four of the 33 tasks, while the other two learners never converge in a single query.

For the three tasks on which Aggressive Co-Testing does not learn 100% accurate rules, the failure is due to the fact that one of the views is significantly less accurate than the other one. This leads to a majority of contention points that are mislabeled by the "bad view," which - in turn - skews the distribution of the queries towards mistakes of the "bad view." Consequently, Co-Testing's performance suffers because such queries are uninformative for

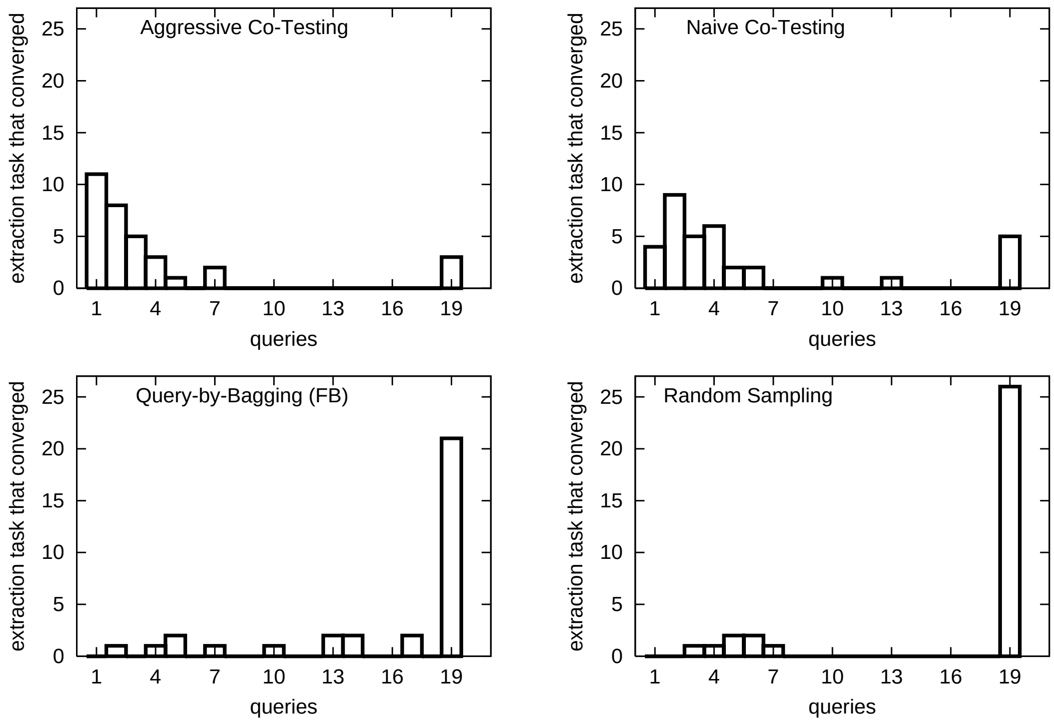

Figure 4: Convergence results for the 33 wrapper induction tasks.

both views: the "good view" makes the correct prediction on them, while the "bad view" is inadequate to learn the target concept. In order to cope with this problem, we introduced a view validation algorithm (Muslea et al., 2002b) that predicts whether the views are appropriate for a particular task.

Finally, let us briefly compare the results above with the ones obtained by WIEN (Kushmerick, 2000), which is the only wrapper induction system that was evaluated on all the extraction tasks used here. As the two experimental setups are not identical (i.e., cross-validation vs. random splits) this is just an *informal* comparison; however, it puts our results into perspective by contrasting Co-Testing with another state of the art approach to wrapper induction.

The results can be summarized as follows: WIEN fails on 18 of the 33 task; these 18 tasks include the three for which Aggressive and Naive Co-Testing failed to learn perfect rules. On the remaining 15 tasks, WIEN requires between 25 and 90 examples[4] to learn the correct rule. For the same 15 tasks, both Aggressive and Naive Co-Testing learn 100% accurate rules based on at most eight examples (two random plus at most six queries).

4. In the WIEN framework, an example consists of a document in which *all* items of interest are labeled. For example, a page that contains a list of 100 names, all labeled, represents a single labeled example. In contrast, for STALKER the same labeled document represents 100 distinct labeled examples. In order to compare the WIEN and STALKER results, we convert the WIEN data to STALKER-like data by multiplying the number of labeled WIEN pages by the average number of item occurrences in each page.

6. Conclusion

In this paper we introduce Co-Testing, which is an active learning technique for multi-view learning tasks. This novel approach to active learning is based on the idea of *learning from mistakes*; i.e., Co-Testing queries unlabeled examples on which the views predict a different label (such contention points are guaranteed to represent mistakes made in one of the views). We have analyzed several members of the Co-Testing family (e.g., Naive, Conservative and Aggressive Co-Testing). We have also introduced and evaluated a Co-Testing algorithm that simultaneously exploits both strong and weak views.

Our empirical results show that Co-Testing is a powerful approach to active learning. Our experiments use four extremely different base learners (i.e., STALKER, IB, Naive Bayes, and MC4) on four different types of domains: wrapper induction, text classification (COURSES), ad removal (AD), and discourse tree parsing (TF). In all these scenarios, Co-Testing clearly outperforms the single-view, state of the art active learning algorithms. Furthermore, except for Query-by-Bagging, Co-Testing is the only algorithm that can be applied to all the problems considered in the empirical evaluation. In contrast to Query-by-Bagging, which has a poor performance on COURSES and wrapper induction, Co-Testing obtains the highest accuracy among the considered algorithms.

Co-Testing's success is due to its ability to discover the mistakes made in each view. As each contention point represents a mistake (i.e., an erroneous prediction) in at least one of the views, it follows that each query is extremely informative for the view that misclassified that example; that is, mistakes are more informative than correctly labeled examples. This is particularly true for base learners such as STALKER, which do not improve the current hypothesis unless they are provided with examples of misclassified instances.

As a limitation, Co-Testing can be applied only to multi-view tasks; that is, unless the user can provide two views, Co-Testing cannot be used at all. However, researchers have shown that besides the four problems above, multiple views exist in a variety of real world problems, such as named entity classification (Collins & Singer, 1999), statistical parsing (Sarkar, 2001), speech recognition (de Sa & Ballard, 1998), word sense disambiguation (Yarowsky, 1995), or base noun phrase bracketing (Pierce & Cardie, 2001).

The other concern about Co-Testing is related to the potential violations of the two multi-view assumptions, which require that the views are both uncorrelated and compatible. For example, in the case of correlated views, the hypotheses learned in each view may be so similar that there are no contention points among which to select the next query. In terms of view incompatibility, remember that, for three of the 33 wrapper induction tasks, one of the views was so inaccurate that the Co-Testing could not outperform Random Sampling. In two companion papers (Muslea et al., 2002a, 2002b) we have proposed practical solutions for both these problems.

Acknowledgments

This research is based upon work supported in part by the National Science Foundation under Award No. IIS-0324955 and grant number 0090978, in part by the Defense Advanced Research Projects Agency (DARPA), through the Department of the Interior, NBC, Acquisition Services Division, under Contract No. NBCHD030010, and in part by the Air Force

Office of Scientific Research under grant number FA9550-04-1-0105. The U.S.Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copy right annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

Abe, N., & Mamitsuka, H. (1998). Query learning using boosting and bagging. In *Proceedings of the 15th International Conference on Machine Learning (ICML-98)*, pp. 1–10.

Abney, S. (2002). Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 360–367.

Aha, D. (1992). Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *International Journal of Man-Machine Studies*, *36*(1), 267–287.

Amoth, T., Cull, P., & Tadepalli, P. (1998). Exact learning of tree patterns from queries and counterexamples. In *Proceedings of the Conference on Computational Learing Theory*, pp. 175–186.

Amoth, T., Cull, P., & Tadepalli, P. (1999). Exact learning of unordered tree patterns from queries. In *Proceedings of the Conference on Computational Learing Theory*, pp. 323–332.

Angluin, D. (1982). A note on the number of queries needed to identify regular languages. *Information and Control*, *51*, 76–87.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Angluin, D. (1994). Exact learning of $\mu-$DNF formulas with malicious membership queries. Tech. rep. YALEU/DCS/TR-1020, Yale University.

Angluin, D., & Krikis, M. (1994). Malicious membership queries and exceptions. Tech. rep. YALEU/DCS/TR-1019, Yale University.

Angluin, D., Krikis, M., Sloan, R., & Turan, G. (1997). Malicious omissions and errors in answers to membership queries. *Machine Learning*, *28*, 211–255.

Angluin, D., & Slonim, D. (1991). Randomly fallible teachers: learning monotone DNF with an incomplete membership oracle. *Machine Learning*, *14*(1), 7–26.

Argamon-Engelson, S., & Dagan, I. (1999). Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, *11*, 335–360.

Baum, E. (1991). Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, *2*, 5–19.

Blum, A., Chalasani, P., Goldman, S., & Slonim, D. (1998). Learning with unreliable boundary queries. *Journal of Computer and System Sciences*, *56*(2), 209–222.

Blum, A., Furst, M., Jackson, J., Kearns, M., Mansour, Y., & Rudich, S. (1994). Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the 26th ACM Symposium on the Theory of Computing*, pp. 253–262.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 1988 Conference on Computational Learning Theory*, pp. 92–100.

Brefeld, U., & Scheffer, T. (2004). Co-EM support vector learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML-2004)*, pp. 121–128.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24(2)*, 123–140.

Califf, M. E., & Mooney, R. (1999). Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pp. 328–334.

Campbell, C., Cristianini, N., & Smola, A. (2000). Query learning with large margin classifiers. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pp. 111–118.

Cohn, D., Atlas, L., & Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, *15*, 201–221.

Cohn, D., Ghahramani, Z., & Jordan, M. (1996). Active learning with statistical models. In *Advances in Neural Information Processing Systems*, Vol. 9, pp. 705–712.

Collins, M., & Singer, Y. (1999). Unsupervised models for named entity classification. In *Proceedings of the Empirical NLP and Very Large Corpora Conference*, pp. 100–110.

Dagan, I., & Engelson, S. (1995). Committee-based sampling for training probabilistic classifiers. In *Proceedings of the 12th International Conference on Machine Learning*, pp. 150–157.

Dasgupta, S., Littman, M., & McAllester, D. (2001). PAC generalization bounds for co-training. In *Neural Information Processing Systems*, pp. 375–382.

de Sa, V., & Ballard, D. (1998). Category learning from multi-modality. *Neural Computation*, *10*(5), 1097–1117.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data vie the EM algorithm. *Journal of Royal Statistical Society*, *39*, 1–38.

Fedorov, V. V. (1972). *Theory of optimal experiment*. Academic Press.

Finn, A., & Kushmerick, N. (2003). Active learning selection strategies for information extraction. In *Proceedings of the ECML-2004 Workshop on Adaptive Text Extraction and Mining (ATEM-2003)*.

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

Ghani, R. (2002). Combining labeled and unlabeled data for multiclass text classification. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pp. 187–194.

Goldberg, P., Goldman, S., & Mathias, D. (1994). Learning unions of boxes with membership and equivalence queries. In *Proceedings of the Conference on Computational Learing Theory*, pp. 198–207.

Goldman, S., & Mathias, D. (1992). Learning k -term DNF formulas with an incomplete membership oracle. In *Proceedings of the Conference on Computational Learing Theory*, pp. 77–84.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pp. 327–334.

Gross, K. (1991). *Concept acquisition through attribute evolution and experiment selection*. Ph.D. thesis, School of Computer Science, Carnegie Mellon University.

Hasenjager, M., & Ritter, H. (1998). Active learning with local models. *Neural Processing Letters*, *7*, 107–117.

Hsu, C.-N., & Dung, M.-T. (1998). Generating finite-state transducers for semi-structured data extraction from the web. *Journal of Information Systems*, *23(8)*, 521–538.

Hwang, J.-N., Choi, J., Oh, S., & Marks, R. (1991). Query-based learning applied to partially trained multilayer perceptrons. *IEEE Transactions on Neural Networks*, *2*, 131 –136.

Jackson, J. (1994). An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pp. 42–53.

Jones, R., Ghani, R., Mitchell, T., & Riloff, E. (2003). Active learning for information extraction with multiple view feature sets. In *Proceedings of the ECML-2004 Workshop on Adaptive Text Extraction and Mining (ATEM-2003)*.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1997). Data mining using MLC++, a machine learning library in C++. *International Journal of AI Tools*, *6(4)*, 537–566.

Kushmerick, N. (1999). Learning to remove internet advertisements. In *Proceedings of the Third International Conference on Autonomous Agents (Agents-99)*, pp. 175–181.

Kushmerick, N. (2000). Wrapper induction: efficiency and expressiveness. *Artificial Intelligence Journal*, *118*(1-2), 15–68.

Kushmerick, N., Johnston, E., & McGuinness, S. (2001). Information extraction by text classification. In *The IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*.

Lang, K., & Baum, E. (1992). Query learning can work poorly when a human oracle is used. In *Proceedings of the IEEE International Joint Conference on Neural Networks*.

Lerman, K., Minton, S., & Knoblock, C. (2003). Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, *18*, 149–181.

Lewis, D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the 11th International Conference on Machine Learning (ICML-94)*, pp. 148–156.

Lewis, D., & Gale, W. (1994). A sequential algorithm for training text classifiers. In *Proceedings of Research and Development in Information Retrieval*, pp. 3–12.

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. In *The 14th National Conference on Artificial Intelligence (AAAI-97)*, pp. 591–596.

Lindenbaum, M., Markovitch, S., & Rusakov, D. (2004). Selective sampling for nearest neighbor classifiers. *Machine Learning*, *54*(2), 125–152.

Marcu, D., Carlson, L., & Watanabe, M. (2000). The automatic translation of discourse structures. In *Proceedings of the 1st Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*, pp. 9–17.

McCallum, A., & Nigam, K. (1998). Employing EM in pool-based active learning for text classification. In *Proceedings of the 15th International Conference on Machine Learning*, pp. 359–367.

Melville, P., & Mooney, R. J. (2004). Diverse ensembles for active learning. In *Proceedings of the International Conference on Machine Learning*, pp. 584–591.

Muslea, I. (2002). *Active Learning with Multiple Views*. Ph.D. thesis, Department of Computer Science, University of Southern California.

Muslea, I., Minton, S., & Knoblock, C. (2000). Selective sampling with redundant views. In *Proceedings of National Conference on Artificial Intelligence (AAAI-2000)*, pp. 621–626.

Muslea, I., Minton, S., & Knoblock, C. (2001). Hierarchical wrapper induction for semistructured sources. *Journal of Autonomous Agents and Multi-Agent Systems*, *4*, 93–114.

Muslea, I., Minton, S., & Knoblock, C. (2002a). Active + Semi-supervised Learning = Robust Multi-view Learning. In *The 19th International Conference on Machine Learning (ICML-2002)*, pp. 435–442.

Muslea, I., Minton, S., & Knoblock, C. (2002b). Adaptive view validation: A first step towards automatic view detection. In *The 19th International Conference on Machine Learning (ICML-2002)*, pp. 443–450.

Muslea, I., Minton, S., & Knoblock, C. (2003). Active learning with strong and weak views: a case study on wrapper induction. In *Proceedings of International Joint Conference on Atificial Intelligence (IJCAI-2003)*, pp. 415–420.

Nahm, U.-Y., & Mooney, R. (2000). A mutually beneficial integration of data mining and information extraction. In *The 17th National Conference on Artificial Intelligence (AAAI-2000)*, pp. 627–632.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. In *Proceedings of Information and Knowledge Management*, pp. 86–93.

Pierce, D., & Cardie, C. (2001). Limitations of co-training for natural language learning from large datasets. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP-2001)*, pp. 1–10.

Raskutti, B., Ferra, H., & Kowalczyk, A. (2002). Combining clustering and co-training to enhance text classification using unlabeled data. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 620–625.

Reddy, C., & Tadepalli, P. (1997). Learning horn definitions with equivalence and membership queries. In *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pp. 243–255.

Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the 18th International Conference on Machine Learning (ICML-2001)*, pp. 441–448.

Sammut, C., & Banerji, R. B. (1986). Learning concepts by asking questions. In Carbonell, R. S. M., Carbonell, J., & 2), T. M. M. V. (Eds.), *Machine Learning: An Artificial Intelligence Approach*, pp. 167–192. Morgan Kaufmann.

Sarkar, A. (2001). Applying co-training methods to statistical parsing. In *Proceedings of the 2nd Annual Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, pp. 175–182.

Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5(2)*, 197–227.

Scheffer, T., & Wrobel, S. (2001). Active learning of partially hidden Markov models. In *Proceedings of the ECML/PKDD-2001 Workshop on Active Learning, Database Sampling, Experimental Design: Views on Instance Selection*.

Schohn, G., & Cohn, D. (2000). Less is more: Active learning with support vector machines. In *Proceedings of the 17th International Conference on Machine Learning (ICML-2000)*, pp. 839–846.

Seung, H. S., Opper, M., & Sompolinski, H. (1992). Query by committee. In *Proceedings of the 1988 Conference on Computational Learning Theory (COLT-72)*, pp. 287–294.

Shapiro, E. (1981). A general incremental algorithm that infers theories from facts. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pp. 446–451.

Shapiro, E. (1982). Algorithmic program diagnosis. In *Proceedings of the 9th ACM Symposium on Principles of Programming Languages*, pp. 299–308.

Sloan, R., & Turan, G. (1994). Learning with queries but incomplete information (extended abstract). In *Proceedings of the Conference on Computational Learing Theory*, pp. 237–245.

Soderland, S. (1999). Learning extraction rules for semi-structured and free text. *Machine Learning*, *34*, 233–272.

Tadepalli, P. (1993). Learning from queries and examples with tree-structured bias. In *Proceedings of the 10th International Conference on Machine Learning (ICML-93)*, pp. 322–329.

Tadepalli, P., & Russell, S. (1998). Learning from queries and examples with structured determinations. *Machine Learning*, 245–295.

Thompson, C., Califf, M. E., & Mooney, R. (1999). Active learning for natural language parsing and information extraction. In *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pp. 406–414.

Tong, S., & Koller, D. (2000). Active learning for parameter estimation in Bayesian networks. In *Advances in Neural Information Processing Systems*, Vol. 13, pp. 647–653.

Tong, S., & Koller, D. (2001). Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, *2*, 45–66.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.

Watkin, T., & Rau, A. (1992). Selecting examples for perceptrons. *Journal of Physics A: Mathematical and General*, *25*(1), 113–121.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting of the Association of Computational Linguistics*, pp. 189–196.

Zhou, Y., & Goldman, S. (2004). Democratic co-learning. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 594–602.