

# Selective Sampling With Redundant Views

Ion Muslea, Steven Minton, and Craig A. Knoblock

Information Sciences Institute and Integrated Media Systems Center

University of Southern California

4676 Admiralty Way

Marina del Rey, CA 90292, USA

{muslea, minton, knoblock}@isi.edu

## Abstract

Selective sampling, a form of active learning, reduces the cost of labeling training data by asking only for the labels of the most informative unlabeled examples. We introduce a novel approach to selective sampling which we call *co-testing*. Co-testing can be applied to problems with *redundant views* (i.e., problems with multiple disjoint sets of attributes that can be used for learning). We analyze the most general algorithm in the co-testing family, *naive co-testing*, which can be used with virtually any type of learner. Naive co-testing simply selects at random an example on which the existing views disagree. We applied our algorithm to a variety of domains, including three real-world problems: wrapper induction, Web page classification, and discourse trees parsing. The empirical results show that besides reducing the number of labeled examples, naive co-testing may also boost the classification accuracy.

## Introduction

In order to learn a classifier, supervised learning algorithms need labeled training examples. In many applications, labeling the training examples is an expensive process because it requires human expertise and is a tedious, time consuming task. *Selective sampling*, a form of active learning, reduces the number of training examples that need to be labeled by examining unlabeled examples and selecting the most informative ones for the human to label. This paper introduces *co-testing*, which is a novel approach to selective sampling for domains with *redundant views*. A domain has redundant views if there are at least two mutually exclusive sets of features that can be used to learn the target concept. Our work was inspired by Blum and Mitchell (1998), who noted that there are many real world domains with multiple views. One example is Web page classification, where one can identify faculty home pages either based on the words on the page or based on the words in HTML anchors pointing to the page. Another example is perception learning with multiple sensors, where we

can determine a robot's position based on vision, sonar, or laser sensors.

Active learning techniques work by asking the user to label an example that maximizes the information conveyed to the learner (we refer to such selected examples as *queries*). In a standard, single-view learning scenario, this generally translates into finding an example that splits the version space in half, i.e., eliminating half of the hypotheses consistent with the training set. With redundant views, we can do much better. Co-testing simultaneously trains a separate classifier for each redundant view. Each classifier is applied to the pool of unlabeled examples, and the system selects a query based on the degree of disagreement among the learners. Because the target hypotheses in each view must agree, co-testing can reduce the hypothesis space faster than would otherwise be possible. To illustrate this, consider a learning problem where we have two views, **V1** and **V2**. For illustrative purposes, imagine an extreme case where there is an unlabeled example  $x$  that is classified as positive by a single hypothesis from the **V1** version space; furthermore, assume that  $x$  is classified as positive by all but one of the hypotheses from the **V2** version space. If the system asks for the label of this example, it will immediately converge to a single hypothesis in one of the spaces and no additional examples will be required.

In the real world, where noise and other effects intrude into the learning process, translating this simple intuition into an effective algorithm raises some interesting issues. In this paper we describe co-testing as a family of algorithms, and empirically analyze a simple implementation of the co-testing approach called *naive co-testing*. This paper begins with two in-depth illustrative examples that contrast co-testing with existing sampling approaches. Then we present the naive co-testing algorithm and discuss its application to both wrapper induction and traditional learning problems.

## Co-testing and uncertainty sampling

There are two major approaches to selective sampling: uncertainty and committee-based sampling. The former queries the unlabeled examples on which the learned classifier is the least confident; the later gener-

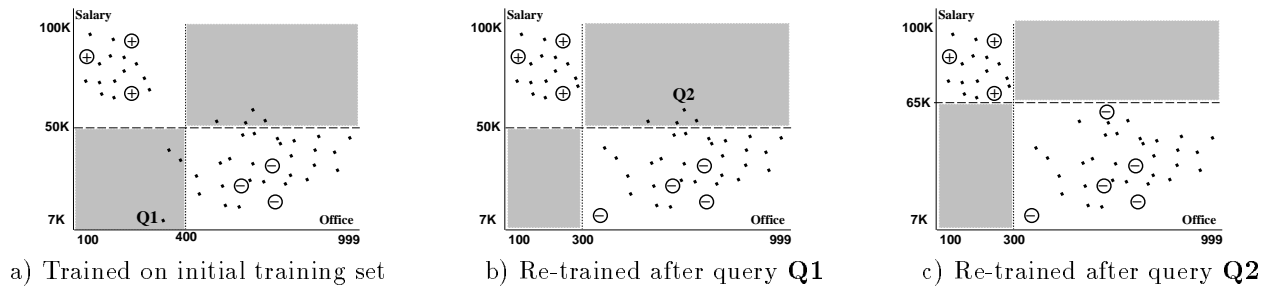


Figure 1: **Co-testing at work.**

ates a committee of several classifiers and selects the unlabeled examples on which the committee members disagree the most. In this section, we contrast co-testing with uncertainty sampling, and in the next section we compare our approach with committee-based sampling.

Let us consider the task of classifying the employees of a CS department in two categories: faculty and non-faculty. Let us assume that the classification can be done either by using a person’s salary (e.g., only faculty have salaries above \$65K) or office number (e.g., only faculty office numbers are below 300). In this case, the domain has two redundant views: one that uses only the salary, and another one that uses only the office number. In both views the target concept is a threshold value: \$65K for salary, and 300 for the office number. To learn the target concepts, we use for both views the following learner  $\mathcal{L}$ : first,  $\mathcal{L}$  identifies the pair of labeled examples that belong to different classes and have the closest attribute values; then  $\mathcal{L}$  sets the threshold to the mean of these two values.

Co-testing works as follows: initially, the user provides a few labeled examples, and a pool of unlabeled ones. In Figure 1a, the unlabeled examples are denoted by points, while the labeled ones appear as  $\oplus$  and  $\ominus$  (the former denotes faculty, and the latter represents non-faculty). We use the learner  $\mathcal{L}$  to create one classifier for each view (the classifiers are geometrically represented as the dotted and the dashed lines, respectively). Then we apply the classifiers to *all* unlabeled examples and determine the *contention points* – the examples that are labeled differently by the two classifiers. The contention points, which lay in the picture’s gray areas, are extremely informative because whenever the two classifiers disagree, at least one of them must be wrong. We select one of the contention points for labeling, add it to the training set, and repeat the whole process.

If the learner can evaluate the confidence of its classification, we can query the contention point on which *both* categorizers are *most confident*, which means that each query *maximally* improves at least one of the hypotheses. In each view from our example, we can measure the confidence level as the distances between the point and the threshold: the larger the distance, the higher the confidence in the classification. In Figure 1a co-testing asks for the label of the example **Q1**, which

is the contention point on which both categorizers are the most confident (i.e., the sum of the distances to the two thresholds is maximal). Once the example is labeled by the user, we re-train, find the new contention points (see Figure 1b), make the query **Q2**, and re-train again. As shown in Figure 1c, the new classifiers agree on all unlabeled examples, and co-testing stops.

As we already mentioned, the traditional approach in uncertainty sampling (Lewis & Gale 1994) consists of learning a single classifier and querying one of the points on which the classifier is the *least confident*. If we use just one of the views in the example above, the lowest confidence points are the ones that are the closest to the threshold. Consequently, uncertainty sampling makes queries that lead to *minimal* improvements of the hypothesis, and it takes more queries to find the correct classifier. In comparison, co-testing has two major advantages. First of all, combining evidence from several views allows us to make queries that lead to maximal improvements. Second, by querying only contention points, we are guaranteed to *always* select an example on which at least one of the classifiers is wrong.

### Co-testing & committee-based sampling

Committee-based algorithms (Seung, Oppen, & Sompolinski 1972)(Abe & Mamitsuka 1998) take a different approach. First, they generate several classifiers (the *committee*) that are consistent with the training set or sub-samples of it, respectively. Then they make the queries that are the most likely to eliminate half of the hypotheses that are consistent with the training set. More precisely, they apply all committee members to each unlabeled example and query the ones on which the committee vote is the most equally split.

Despite their advantages, committee-based algorithms have difficulties on some types of problems. For example, consider the problem  $\mathcal{P}$  of learning *conjunctive concepts* in an instance space with 10,000 binary attributes that can be split into two redundant views:  $\mathbf{V1}(a_1, a_2, \dots, a_{5000})$  and  $\mathbf{V2}(a_{5001}, a_{5002}, \dots, a_{10000})$ . Let us assume that the target concept has the following equivalent definitions:

- in  $\mathbf{V1}$ :  $\langle t, t, t, ?, ?, \dots, ? \rangle$ ;
- in  $\mathbf{V2}$ :  $\langle f, f, f, ?, ?, \dots, ? \rangle$ ;
- in  $\mathbf{V1} \cup \mathbf{V2}$ :  $\langle t, t, t, ?, ?, \dots, ?, f, f, f, ?, ?, \dots, ? \rangle$ .

Given:

- a problem  $\mathcal{P}$  with features  $\mathbf{V} = \{a_1, a_2, \dots, a_N\}$
- a learning algorithm  $\mathcal{L}$
- two views  $\mathbf{V1}$  and  $\mathbf{V2}$  ( $\mathbf{V} = \mathbf{V1} \cup \mathbf{V2}$  and  $\mathbf{V1} \cap \mathbf{V2} = \emptyset$ )
- the sets  $T$  and  $U$  of labeled and unlabeled examples

LOOP for  $k$  iterations

- use  $\mathcal{L}$ ,  $\mathbf{V1}(T)$ , and  $\mathbf{V2}(T)$  to create classifiers  $h_1$  and  $h_2$
- let  $ContentionPoints = \{x \in U, h_1(x) \neq h_2(x)\}$
- let  $x = \mathbf{SelectQuery}(ContentionPoints)$
- remove  $x$  from  $U$ , ask for its label, and add it to  $T$

Figure 2: **The Co-Testing Family of Algorithms.**

The meaning of these concepts is straightforward: for example, in  $\mathbf{V1}$ ,  $a_1$ ,  $a_2$ , and  $a_3$  must be  $\mathbf{t}$ , and the other attributes do not matter. Finally, let us further assume that the attribute  $a_4$  has the value  $\mathbf{t}$  for 99% of the instances (i.e., it rarely has the value  $\mathbf{f}$ ). The scarcity of examples with  $a_4 = \mathbf{f}$  makes the target concept difficult to learn because it is highly improbable that a random training set includes such an examples. For domains like this one, the challenge consists of identifying these rare and informative examples.<sup>1</sup>

For this problem, we use the FIND-S learner (Mitchell 1997), which generates *the most specific* hypothesis that is consistent with all positive examples. We chose FIND-S because boolean conjunctions are PAC-learnable by FIND-S (i.e., with a high probability, the target concept can be learned in polynomial time based on a polynomial number of *randomly* chosen examples).

Now let us assume that we apply a committee-based approach to the 10,000-attribute instance space. In the initial training set  $a_4$  is unlikely to have the value  $\mathbf{f}$ , in which case *all* initial committee members will have  $a_4$  set to  $\mathbf{t}$ ; this means that the queries are also unlikely to have  $a_4 = \mathbf{f}$  because such examples are classified as negative by all committee members (remember that queries are made only on examples on which the committee is split). After several queries, all the committee members become identical ( $\langle \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, ?, \dots, ?, \mathbf{f}, \mathbf{f}, \mathbf{f}, ?, ?, \dots, ? \rangle$ ) and learning stops. Consequently, even though the target concept is PAC-learnable, with high probability the learned concept will *not* be the correct one.

By contrast, co-testing easily learns the correct concept. First, after several queries, it generates the concepts  $\langle \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, ?, \dots, ? \rangle$  for  $\mathbf{V1}$  and  $\langle \mathbf{f}, \mathbf{f}, \mathbf{f}, ?, \dots, ? \rangle$  for  $\mathbf{V2}$ , which correspond to the concept  $\langle \mathbf{t}, \mathbf{t}, \mathbf{t}, \mathbf{t}, ?, \dots, ?, \mathbf{f}, \mathbf{f}, \mathbf{f}, ?, ?, \dots, ? \rangle$  learned above. These two hypotheses disagree on *all* unlabeled examples that have  $a_4 = \mathbf{f}$  ( $\mathbf{V1}$  labels them negative, while  $\mathbf{V2}$  labels them positive) *and only on those*. Consequently, co-testing queries such an example and learns the correct hypotheses:  $\langle \mathbf{t}, \mathbf{t}, \mathbf{t}, ?, \dots, ? \rangle$  and  $\langle \mathbf{f}, \mathbf{f}, \mathbf{f}, ?, \dots, ? \rangle$ , respectively.

<sup>1</sup>Later in this paper we will discuss wrapper induction, which is a typical example of problem with rare values.

In order to make the problem more realistic, let us now assume that there are two attributes with rare values. In case they both fall within the same view, the argument above remains valid, and co-testing is guaranteed to find the correct hypothesis. If the two attributes belong to different views, co-testing still finds the perfect hypothesis unless both rare values *always* appear together in all unlabeled examples (which is highly unlikely). A similar argument holds for an arbitrary number of independent attributes with rare values.

## The co-testing algorithms

In this section we present a formal description of the co-testing family of algorithms, which was designed for problems with *redundant views*. By definition, a learning problem  $\mathcal{P}$  is said to have redundant views if its set of attributes  $\mathbf{V} = \{a_1, a_2, \dots, a_N\}$  can be partitioned in two disjoint views  $\mathbf{V1}$  and  $\mathbf{V2}$ , and either view is sufficient to learn a classifier for  $\mathcal{P}$ . Ideally, the two views should be able to reach the same classification accuracy, but we will see later that in practice this is *not* a necessary condition.

Given a learner  $\mathcal{L}$ , a set  $T$  of labeled examples, and a set  $U$  of unlabeled ones, co-testing (see Figure 2) works as follows: first, it uses  $\mathcal{L}$  to learn two classifiers  $h_1$  and  $h_2$  based on the projections of the examples in  $T$  onto the two views,  $\mathbf{V1}$  and  $\mathbf{V2}$ . Then it applies  $h_1$  and  $h_2$  to all unlabeled examples and creates the list *ContentionPoints* of all unlabeled examples on which they disagree. The difference between the members of the co-testing family comes from the manner in which they select the next query. *Naive co-testing*, on which we will focus in the remaining sections, is the most straightforward member of the family: it *randomly* queries one of the contention points. Naive co-testing is also the most general member of the family because it can be applied to virtually any type of learner (the more sophisticated version discussed in the second section is applicable only to learners that can reliably estimate the confidence of their classification). Despite its simplicity, the empirical results show that naive co-testing is a powerful selective sampling algorithm. We believe that more sophisticated versions of co-testing should lead to faster convergence, but this is a topic that we are still investigating.

## Naive co-testing for wrapper induction

A plethora of applications are using data extracted from collections of on-line documents. To avoid hand-writing a large number of extraction rules, researchers focused on learning the rules based on labeled examples. As labeling such examples is an extremely tedious and time consuming task, active learning can play a crucial role in reducing the user's burden. However, relatively little attention has been paid to applying active learning to information extraction. The only approaches that we know about, (Thompson, Califf, & Mooney 1999) and (Soderland 1999), are not general-purpose algorithms

because they select the queries based on heuristics specific to their respective learners, RAPIER and WHISK.

Wrapper induction algorithms, like STALKER (Muslea, Minton, & Knoblock 2000), are designed to learn high accuracy extraction rules for semi-structured documents. For instance, let us assume that we want to extract the restaurant names from a collection of documents that look similar to the Web-page fragment shown in Figure 3. To find the beginning of the restaurant name, we can use the *start rule* **R1** = *SkipTo(Cuisine:)**SkipTo(<p>)*. **R1** starts from the beginning of the page and ignores everything until it finds the string “Cuisine:”; then, again, it ignores everything until it finds “<p>”. A similar *end rule* can be used to find the end of the name within the document.

An alternative way to find the start of the name is to use the rule **R2** = *SkipTo(Phone)**SkipTo(Capitalized)*, which is applied *backward*, from the end of the document, and has similar semantics: it ignores everything until it finds “Phone” and then, again, skips to the first capitalized word. In this paper, we call **R1** and **R2** *forward* and *backward* start rules, respectively. As STALKER can learn both forward and backward rules, we can create the two views in a straightforward manner: **V1** and **V2** consist of the sequences of characters that *precede* and *follow* the beginning of the item, respectively. More precisely, in **V1** we learn forward rules, while in **V2** we learn backward rules. Finally, to apply co-testing to wrapper induction, we use STALKER’s learning algorithm and the two views described above.

Note that by combining forward/backward start and end rules, one can obtain three types of wrappers: **FB** (Forward start and Backward end rules), **FF** (Forward start and Forward end rules), and **BB** (Backward start and Backward end rules). Out of the 206 extraction tasks described in (Muslea, Minton, & Knoblock 2000), we applied *naive co-testing* on the 10 tasks on which, based on random examples, STALKER failed to generate perfect wrappers of *all three types*. STALKER was successively trained on random training sets of sizes 1, 2, ..., 10; the extraction accuracy was averaged over 20 runs. For co-testing, we started with one random labeled example and made nine queries. We used such small training sets because the learning curves tend to flatten even before reaching 10 examples.

Because of the space constraints, we present here only an overview of the empirical results. Over the 10 tasks, stand-alone STALKER reached the following average accuracies: 82.6% (**FB**), 84.4% (**FF**), and 81.4% (**BB**). By applying co-testing,<sup>2</sup> we obtained significantly higher accuracies for all three classes of wrappers: 90.7% (**FB**), 93.2% (**FF**), and 91.2% (**BB**). These

<sup>2</sup>We compared co-testing only with STALKER because there is no other active learning algorithm for wrapper induction. Furthermore, STALKER can not be used in a straightforward manner in conjunction with existing general-purpose selective sampling algorithms (Seung, Opper, & Sompolinski 1972) (Cohn, Atlas, & Ladner 1994) (Abe & Mamitsuka 1998).

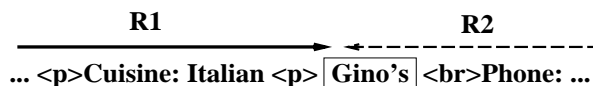


Figure 3: Extracting the restaurant name.

results deserve several comments. First, for all three classes of wrappers, co-testing reduced the error rate by 47%, 57%, and 53%, respectively. Second, for four of the 10 tasks, co-testing learned 100% accurate wrappers of *all three types*. Furthermore, these perfect wrappers were learned based on as few as five or six queries. Third, on all 10 tasks co-testing *improved* the accuracy of the *most accurate* of the three types of wrappers. Finally, on eight tasks co-testing also improved the accuracy of the *least accurate* of the wrappers. We can conclude that applying co-testing to STALKER leads to a dramatic improvement in accuracy without having to label more training data.

### Beyond wrapper induction

In order to contrast naive co-testing with state-of-the-art sampling algorithms, we applied it to more traditional machine learning domains. In this paper, we compared naive co-testing with query-by-bagging and -boosting (Abe & Mamitsuka 1998) because these are techniques where performance has been reported on several well-studied UCI domains.<sup>3</sup> These two algorithms are also the most general selective sampling approaches in terms of practical applicability (i.e., similarly to co-testing, they can use a large variety of learners). We implemented all three algorithms based on the *MCC++* library (Kohavi, Sommerfield, & Dougherty 1997), and we used as learner MC4, which is the *MCC++* implementation of C4.5.

First, we applied co-testing on two real world domains for which there is a natural, intuitive way to create the two views: **Ad** (Kushmerick 1999) and **Transfer-Few** (Marcu, Carlson, & Watanabe 2000). The former is a Web classification problem with two classes, 1500 attributes, and 3279 examples. It classifies Web images into ads and non-ads and has the following views: **V1** describes the image itself (geometry, words in the image’s URL and caption), while **V2** contains all other features (e.g., words from the URL of the page that contains the image, and words from the URL of the page the image points to). The second domain, **Transfer-Few**, has seven classes, 99 features and 11,193 examples. It uses a shift-reduce parsing paradigm in order to learn to rewrite Japanese discourse trees as English-like discourse trees in the context of a machine translation system. In this case, **V1** describes features specific to a shift-reduce parsing paradigm: the elements in the input list and the partial trees in the stack. **V2** describes features specific to the Japanese tree given as input.

<sup>3</sup><http://www.ics.uci.edu/~mllearn/MLRepository.html>

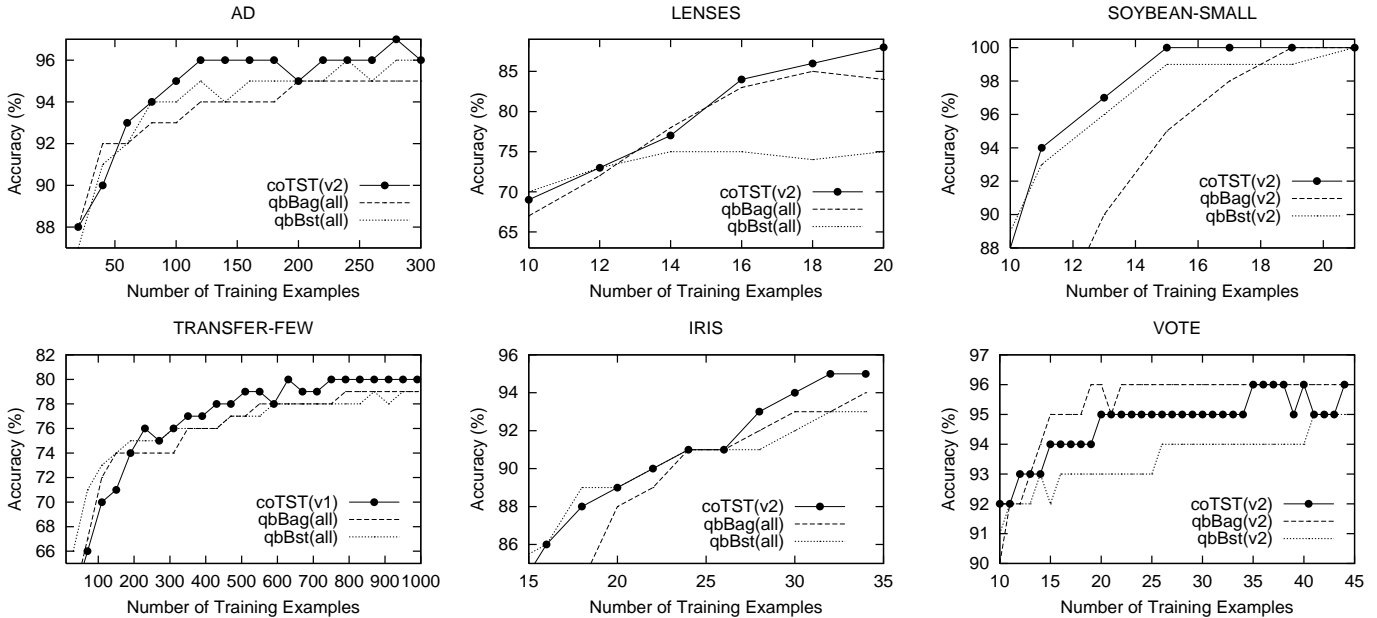


Figure 4: Co-Testing on Traditional Machine Learning Domains.

As the size of these two domains leads to high computational costs, we used 2-fold cross validation and averaged the results over 5 random runs. In both domains we started with a randomly chosen training set of 10 examples, and we made 10 and 20 queries after each learning episode, respectively. As shown in the first column of Figure 4, co-testing clearly outperforms query-by-bagging and -boosting on both domains. We must emphasize that these results were obtained despite the fact that the classifier used by co-testing is less powerful than the other two (i.e., a single decision tree *vs* 20 bagged/boosted decision trees).

To better understand how naive co-testing can outperform such powerful algorithms, we continued our empirical investigation on small size UCI domains. As in most of these domains there is no natural way to create two views, we artificially created the views by splitting the attributes in half<sup>4</sup> (i.e., first and last 50% of them). Finally, in order to have a fair comparison, we selected the domains on which bagging and boosting have accuracies as close as possible to MC4: **Lenses**, **Soybean-Small**, **Iris**, and **Vote**. For all these domains, we used 10-fold cross validation and averaged the results over 20 random runs. For each domain, we started with a random training set of five examples, and we made one query after each learning episode.

<sup>4</sup>The only exception is the *iris* domain (4 attributes), where there is a large difference in the accuracy of the views  $\{a_1, a_2\}$  and  $\{a_3, a_4\}$  (72% and 94%, respectively). By using the views  $\{a_1, a_2, a_3\}$  and  $\{a_4\}$ , we obtained closer accuracies: 94% and 95%, respectively. Note that the view  $\{a_4\}$  is *not* restricted to single-level decision trees because  $a_4$  has continuous values.

Figure 4 shows the learning curves for both the UCI and the real world domains. For co-testing, we always present the results on the best of the two views. For query-by-bagging and -boosting, we show the best learning curve among **V1**, **V2**, and **V1∪V2** (denoted by **v1**, **v2**, and **all** in the legends from Figure 4). Note that on **soybean-small** and **vote**, the best accuracy of query-by-bagging and -boosting was obtained on **V2**, not on all features!

The empirical results require a few comments. First of all, co-testing outperforms query-by-boosting on all six domains. Second, co-testing does better than query-by-bagging on five domains; on the sixth one, **Vote**, both reach the same final accuracies, but query-by-bagging is capable of doing it based on fewer queries. The explanation is quite simple: on this domain, bagging consistently outperforms MC4 by 1%, which means that it reaches the final accuracy based on fewer examples. Third, on all domains except **Soybean-Small** and **Vote**, the best accuracy of query-by-bagging and -boosting is obtained on **V1∪V2**. This means that co-testing outperforms even more clearly the other two algorithms on the individual views. Last but not least, we found it interesting that the UCI domains contain so much redundancy that our arbitrary views do not lead to any loss of accuracy.

## Discussion

In (Blum & Mitchell 1998), the authors showed that redundant views can provide an important source of information for supervised machine learning algorithms. Previously, this topic was largely ignored, though the idea clearly shows up in many unsupervised applica-

tions using techniques like EM(Dempster, Laird, & Rubin 1977). However, rather than considering active learning methods, Blum and Mitchell use the two views to learn hypotheses that feed each other with the unlabeled examples on which their classification is the most confident.

Our empirical results show that co-testing is a potentially powerful approach for active learning. In the wrapper induction, **Ad**, and **Transfer-Few** domains, all of which have natural redundant views, naive co-testing clearly improves upon the current state of the art. We believe that co-testing works so well in these domains because it can identify the rarely occurring cases that are relevant, as described in the third section. We note that all these three domains have large number of features, so finding relevant but rare feature-values contributes significantly to performance.

Naive co-testing's good performance on the UCI domains was more surprising, especially since we derived the views by splitting the features arbitrarily. We conjecture that splitting the problem into two views is profitable because it effectively produces committees that are independent, so that the hypotheses produced by one view are quite different than those produced in the other view. Perhaps any committee-based technique that encourages such variation within its committee would do as well.

Whether or not co-testing turns out to do well on traditional single-view domains, we believe that it will have practical value because many large real-world domains do have redundant views. We note that the views are not required to lead to the same accuracy, which makes the constraint easier to fulfill (in fact, none of our domains above had equally accurate views). Clearly, more work needs to be done here, both in exploring the space of co-testing algorithms as well as analyzing the theoretical underpinnings of the approach. Nevertheless, we believe this study presents a step towards an interesting new approach to active learning.

## Conclusion

This paper introduced co-testing, a family of selective sampling algorithms. Co-testing queries one of the contention points among multiple, redundant views. We focused on a simple member of the family, *naive co-testing*, which randomly selects one of the contention points. We provided empirical evidence that on domains like wrapper induction, where other sampling methods cannot be naturally applied, *co-testing* leads to significant improvements of the classification accuracy. We also applied *naive co-testing* to traditional machine learning domains, and we showed that its query selection strategy is comparable to the more sophisticated ones used in query-by-bagging and -boosting.

We plan to continue our work on co-testing by following several research directions. First, we will continue studying the various members of the family in order to fully understand both its advantages and its weaknesses. Second, we plan to provide theoretical guar-

antees for the most interesting members of the family. Third, we will search a formal way to detect the redundant views within a given domain. Last but not least, we will perform a large-scale empirical evaluation by applying co-testing to various real world problems such as Web classification and natural language processing.

## Acknowledgments

This work was supported in part by USC's Integrated Media Systems Center (IMSC) - an NSF Engineering Research Center, by the National Science Foundation under grant number IRI-9610014, by the U.S. Air Force under contract number F49620-98-1-0046, by the Defense Logistics Agency, DARPA, and Fort Huachuca under contract number DABT63-96-C-0066, and by research grants from NCR and General Dynamics Information Systems. The views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion or policy of any of the above organizations or any person connected with them.

## References

- Abe, N., and Mamitsuka, H. 1998. Query learning using boosting and bagging. In *Proc. of ICML*, 1-10.
- Blum, A., and Mitchell, T. 1998. Combining labeled and unlabeled data with co-training. In *Proc. of the 1988 Conf. Computational Learning Theory*, 92-100.
- Cohn, D.; Atlas, L.; and Ladner, R. 1994. Improving generalization with active learning. *ML* 15:201-221.
- Dempster, A.; Laird, N.; and Rubin, D. 1977. Maximum likelihood from incomplete data via the EM algorithm. *J. of Royal Statistical Society* 39:1-38.
- Kohavi, R.; Sommerfield, D.; and Dougherty, J. 1997. Data mining using MLC++, a machine learning library in C++. *Intl. J. of AI Tools* 6(4):537-566.
- Kushmerick, N. 1999. Learning to remove internet advertisements. In *Proc. of Auton. Agents-99*, 175-181.
- Lewis, D., and Gale, W. 1994. A sequential algorithm for training text classifiers. In *Proc. of Research and Development in Information Retrieval*, 3-12.
- Marcu, D.; Carlson, L.; and Watanabe, M. 2000. The automatic translation of discourse structures.
- Mitchell, T. 1997. Machine learning. McGraw-Hill.
- Muslea, I.; Minton, S.; and Knoblock, C. 2000. Hierarchical wrapper induction for semistructured information sources. *J. Autonom. Agents & Multi-Agent Sys.*
- Seung, H.; Opper, M.; and Sompolinski, H. 1972. Query by committee. In *Proc. of COLT-72*, 287-294.
- Soderland, S. 1999. Learning extraction rules for semistructured and free text. *ML* 34:233-272.
- Thompson, C.; Califf, M.; and Mooney, R. 1999. Active learning for natural language parsing and information extraction. In *Proc. of ICML-99*, 406-414.