Selective Sampling With Naive Co-Testing: Preliminary Results

Ion Muslea[1] and **Steven Minton**[2] and **Craig A. Knoblock**[3]

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.

1 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.

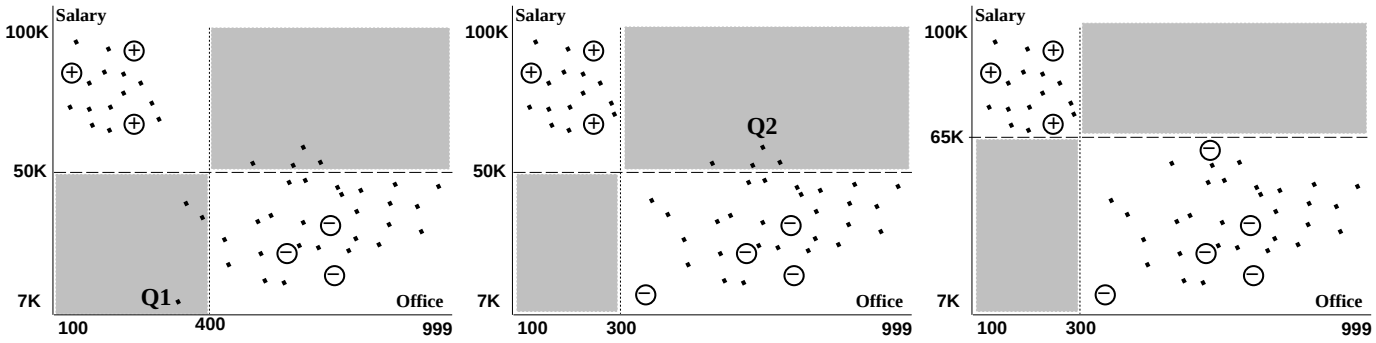
In the real world, where noise and other effects intrude into the learning process, translating this simple intuition into an effective algorithm raises some interesting issues. In this paper we describe co-testing as a family of algorithms, and empirically analyze a simple implementation of the co-testing approach called *naive co-testing*. We begin 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.

2 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 latter generates 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.

[1] Information Sciences Institute, Integrated Media Systems Center, and Computer Science Department, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90230, USA, email: muslea@isi.edu
[2] Information Sciences Institute, Integrated Media Systems Center, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90230, USA, email: minton@isi.edu
[3] Information Sciences Institute, Integrated Media Systems Center, and Computer Science Department, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90230, USA, email: knoblock@isi.edu



a) Initial hypotheses b) After query **Q1** c) After query **Q2**

Figure 1. Co-testing at work.

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 classifiers agree on all unlabeled examples, and co-testing stops.

As we already mentioned, the traditional approach in uncertainty sampling [5] 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.

3 CO-TESTING & COMMITTEE-BASED SAMPLING

Committee-based algorithms [8][1] 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: **V1**(a_1,a_2,...,a_{5000}) and **V2**(a_{5001},a_{5002},...,a_{10000}). Let us assume that the target concept has the following equivalent definitions:

- in **V1**: <t,t,t,?,?,...,?>;
- in **V2**: <f,f,f,?,?,...,?>;
- in **V1**∪**V2**: <t,t,t,?,?,...,?,f,f,f,?,?,...,?>.

The meaning of these concepts is straightforward: for example, in **V1**, a_1, a_2, and a_3 must be t, and the other attributes do not matter. Finally, let us further assume that the attribute a_4 has the value t for 99% of the instances (i.e., it rarely has the value f). The scarcity of examples with a_4=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. A typical problem with rare values is wrapper induction, which will be discussed at length later in this paper.

For this problem, we use the FIND-S learner [**?**], 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 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. As in the initial training set a_4 is unlikely to have the value f, all initial committee members will have a_4 set to t; this means that the queries are also unlikely to have a_4=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 (<t,t,t,t,?,...,?,f,f,f,?,?,...,?>) 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,

Given:

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

LOOP for k iterations

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

Figure 2. The Co-Testing Family of Algorithms.

after several queries, it learns the concepts `<t,t,t,t,?,...,?>` for **V1** and `<f,f,f,?,...,?>` for **V2**, which correspond to the concept `<t,t,t,t,?,...,?,f,f,f,?,?,...,?>` learned above. These two hypotheses disagree on all unlabeled examples that have a_4=`f` (**V1** labels them negative, while **V2** labels them positive) *and only on those*. Consequently, co-testing queries such an example and learns the correct hypotheses: `<t,t,t,?,...,?>` and `<f,f,f,?,...,?>`, respectively.

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.

4 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 **V** = $\{a_1, a_2, \ldots, a_N\}$ can be partitioned in two disjoint views **V1** and **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 latter 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, **V1** and **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.

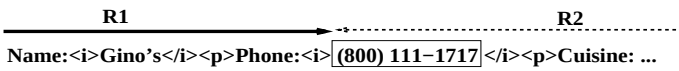


Figure 3. Extracting the phone number.

5 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 existing approaches, [10] and [9], 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 [7], are designed to learn high accuracy extraction rules for semi-structured documents. For instance, let us assume that we want to extract the phone numbers from a collection of documents that look similar to the Web-page fragment shown in Figure 3. To find the beginning of phone number, we can use the *start rule* **R1** = $SkipTo$(`Phone:<i>`). **R1** starts from the beginning of the page and ignores everything until it finds the string `Phone:<i>`. A similar rule can be used to find the end of the phone number.

An alternative way to find the start of the phone number is to use the rule **R2** = $SkipTo$(`Cuisine`)$SkipTo$(($Number$)), which is applied $backward$, from the end of the document, and has similar semantics: it ignores everything until it finds "`Cuisine`" and then, again, skips to the first number between parantheses.

The rules **R1** and **R2** are called $forward$ and $backward$ start rules, respectively. By simply reversing the direction in which a rule is applied, STALKER uses the same algorithm to learn both forward and backward rules. This duality allows us to create the two views in a straightforward manner. To learn a start rule, we can apply co-testing as follows: we use STALKER's learning algorithm, and the views **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.

Before analizing the empirical evaluation of co-testing on wrapper induction, we must provide a few additional details. First of all, in order to extract the items of interest, one must have $both$ a start rule and an end rule (each item is uniquely defined by its starting and ending points). Second, the end rules can be learned based on views similar to the ones described above: the sequences of characters that $precede$ and $follow$ the end of the item. Last but not least, in order to extract an item of interest, based on the various combinations of forward/backward start and end rules, one can identify three main classes of wrappers:

- the **FB** class: **Forward** start rule + **Backward** end rule. This is the type of wrapper learned by the STALKER algorithm for the work reported in [7]. The two rules are applied $independently$ of each other: the former starting from the beginning of the document, and the latter starting from the end of it.
- the **FF** class: **Forward** start rule + **Forward** end rule. The start rule is applied forward from the beginning of the document, while the latter is also applied forward, but from the point where the start rule

Table 1. Results on the 10 hardest tasks.

Task	Exs	STALKER			Co-Testing +STALKER		
		FB	FF	BB	FB	FF	BB
s2.2	491	51.7	75.3	49.5	89.8	82.7	94.0
s2.3	466	82.2	82.2	81.4	90.2	90.2	70.1
s3.4	174	98.1	64.7	98.2	99.1	85.9	99.1
s3.5	174	96.7	96.7	97.5	**100**	**100**	**100**
s6.1	26	92.3	97.3	95.0	**100**	**100**	**100**
s9.10	43	92.0	91.6	93.3	**100**	**100**	**100**
s11.3	71	67.1	93.1	67.1	47.5	93.3	49.5
s24.3	689	90.2	90.2	77.0	99.4	99.4	99.9
s26.3	376	97.9	97.1	97.9	**100**	**100**	**100**
s26.5	376	58.0	58.0	57.9	81.3	81.3	99.6
Mean		82.6	84.4	81.4	90.7	93.2	91.2

stopped.

- the **BB** class: **Backward** start rule + **Backward** end rule. The end rule is applied backward from the end of the document, while the start rule is also applied backward, but from the point where the end rule stopped.

Note that the **FB**, **FF**, and **BB** wrappers represent equally valid ways to extract an item. However, the three classes of wrappers are not equivalent. For example, given a particular extraction task, it is possible that no **FF** wrapper is 100% accurate, even though there are 100%-accurate **FB** and **BB** wrappers. Furthermore, even for tasks for which there are perfect wrappers of all three types, it may happen that one type of wrapper is more difficult to learn than the other ones (i.e., it requires more training examples).

To evaluate *naive co-testing* for wrapper induction, we applied it on the 23 extraction tasks on which STALKER failed to generate perfect **FB** wrappers [7]. For the purpose of this discussion, we split the 23 tasks in two groups:

- the 10 tasks on which, based on random examples, STALKER fails to learn perfect wrappers of any type (see Table 1);
- the 13 tasks on which, based on random examples, STALKER fails to learn perfect **FB** wrappers, but manages to learn either a perfect **FF** or a perfect **BB** wrapper (see Table 2).

The second group is obviously less interesting than the first one, but it is worth being analized because one does not know apriori whether or not one class of wrappers is better fit than the other ones for a particular task.

For all tasks, we followed the experimental setup from [7], where STALKER was successively trained on $randomly$ chosen training sets of sizes 1, 2, …, 10, and the reported accuracy was averaged over 20 runs. We use such small training sets because, in practice, the learning curves tend to flatten even before reaching 10 examples.

Compared with stand-alone STALKER,[4] over the 23 tasks, co-testing improved the average accuracies of the three classes of wrappers as follows:

- **FB**: from 85.7% to 94.2% (error rate reduced by 59.5%);
- **FF**: from 92.9% to 97.0% (error rate reduced by 57.8%);
- **BB**:from 85.3% to 94.5% (error rate reduced by 62.6%).

Note that for all three classes of wrappers, co-testing reduced the overall average error rate by more than 57%!

[4] 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 [8] [3] [1].

Table 2. Results on other 13 difficult tasks.

Task	Exs	STALKER			Co-Testing + STALKER		
		FB	FF	BB	FB	FF	BB
s1.1	403	86.7	**100**	86.7	98.7	100	98.7
s1.2	403	94.4	**100**	93.3	97.1	100	97.1
s2.1	500	98.3	**100**	95.1	100	100	100
s6.9	26	97.6	**100**	97.6	100	100	100
s6.10	15	96.6	96.6	**100**	100	100	100
s9.7	43	93.6	**100**	93.6	100	100	100
s9.11	38	96.0	**100**	98.8	100	100	100
s9.le2	30	99.5	99.5	**100**	99.0	99.0	100
s11.0	90	93.7	**100**	93.7	99.6	100	99.6
s11.1	90	97.3	97.3	**100**	100	100	100
s11.2	90	69.0	**100**	69.0	71.6	100	71.6
s24.1	423	93.0	**100**	90.9	96.8	100	96.8
s26.4	376	30.1	**100**	30.1	98.0	99.7	98.0
Mean		88.1	99.4	88.3	96.9	99.9	97.0

The results above deserve a few comments. First, on the 10 most difficult tasks presented in Table 1, in four cases co-testing learns 100% accurate wrappers of *all three types*. Furthermore, for all these four tasks, the perfect wrappers are learned based on less than 9 queries: five for s6.1, six for s11.3 and s26.3, and eight for s3.5. Second, on all 10 tasks co-testing improves the accuracy of the best STALKER wrapper. Third, only on two tasks, s2.3 and s11.3, the accuracy of the $worst$ of the three wrappers decreases. Finally, the results in Table 2 were quite expected: as for each of these 13 tasks *at least* one class of wrappers could be learned with 100% accuracy based on $random$ examples, this "easy-to-learn" class behaves like an "oracle" that detects all the mistakes of the other two classes of wrappers. We can conclude that applying co-testing to STALKER leads to a dramatic improvement in accuracy without having to label more training data.

6 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 [1] because these are techniques where performance has been reported on several well-studied UCI domains.[5] 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 $\mathcal{MLC}$++ library [4], and we used as learner MC4, which is the $\mathcal{MLC}$++ implementation of C4.5.

We present here[6] the results of co-testing on two real world domains for which there is an intuitive way to create the two views: Ad [**?**] and Transfer-Few [6]. 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

[5] http://www.ics.uci.edu/∼mlearn/MLRepository.html

[6] More results on applying co-testing to UCI domains are presented in [**?**]

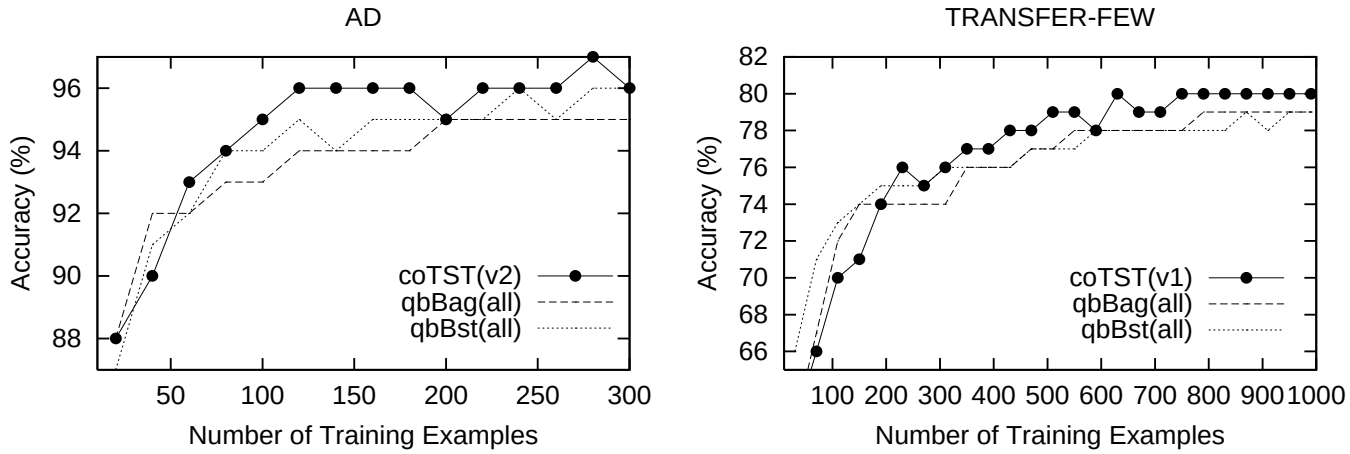


Figure 4. Co-Testing on Traditional Machine Learning Domains.

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.

As the size of these 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 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).

7 DISCUSSION

In [2], the authors showed that redundant views can provide an important source of information for supervised learning algorithms. Previously, this topic was largely ignored, though the idea clearly shows up in many unsupervised applications using techniques like EM[**?**]. However, rather than considering active learning methods, the authors 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 application domains discussed above, 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.

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 (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.

8 CONCLUSION

This paper introduced co-testing, a family of selective sampling algorithms. We focused on a simple member of the family, *naive co-testing*, which randomly selects one of the contention points among multiple, redundant views. 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 or better than 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 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.

ACKNOWLEDGEMENTS

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

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

[2] A. Blum and T. Mitchell, 'Combining labeled and unlabeled data with co-training', in *Proceedings of the 1988 Conference on Computational Learning Theory*, pp. 92–100, (1998).
[3] L. Atlas D. Cohn and R. Ladner, 'Improving generalization with active learning', *Machine Learning*, **15**, 201–221, (1994).
[4] A. Dempster, N. Laird, and D. Rubin, 'Maximum likelihood from incomplete data vie the EM algorithm', *Journal of Royal Statistical Society*, **39**, 1–38, (1977).
[5] R. Kohavi, D. Sommerfield, and J. Dougherty, 'Data mining using mlc++, a machine learning library in c++', *International Journal of Artificial Intelligence Tools*, **6(4)**, 537–566, (1997).
[6] N. Kushmerick, 'Learning to remove internet advetisements', in *Proceedings of the Third International Conference on Autonomous Agents (Seattle 1999)*, pp. 175–181, (1999).
[7] D. Lewis and W. Gale, 'A sequential algorithm for training text classifiers', in *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12, (1994).
[8] D. Marcu, 'A decision-based approach to rhetorical parsing', in *The 37th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pp. 365–372, (1999).
[9] T. Mitchell, 'Machine learning'. McGraw-Hill, (1997).
[10] I. Muslea, S. Minton, and C. Knoblock, 'Hierarchical wrapper induction for semistructured information sources', *Journal of Autonomous Agents & Multi-Agent Systems (in press)*, (2000).
[11] I. Muslea, S. Minton, and C. Knoblock, 'Selective sampling with redundant views', in *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-2000)*, (2000).
[12] H. Seung, M. Opper, and H. Sompolinski, 'Query by committee', in *Proceedings of the 5th Workshop on Computational Learning Theory*, pp. 287–294, (1972).
[13] S. Soderland, 'Learning extraction rules for semi-structured and free text', *Machine Learning*, **34**, 233–272, (1999).
[14] C. Thompson, M. Califf, and R. Mooney, 'Active learning for natural language parsing and information extraction', in *Proceedings of the 16th International Conference on Machine Learning (ICML-99)*, pp. 406–414, (1999).