Active Learning with Strong and Weak Views: A Case Study on Wrapper Induction

Ion Muslea, Steven N. Minton, Craig A. Knoblock
muslea@isi.edu, minton@fetch.com, knoblock@isi.edu
U. of Southern California, Fetch Technologies, Inc., U. of Southern California,
4676 Admiralty Way
Marina del Rey, CA 90292

Abstract

Multi-view learners reduce the need for labeled data by exploiting disjoint sub-sets of features (*views*), each of which is sufficient for learning. Such algorithms assume that each view is a *strong view* (i.e., perfect learning is possible in each view). We extend the multi-view framework by introducing a novel algorithm, Aggressive Co-Testing, that exploits both strong and *weak views*; in a weak view, one can learn a concept that is strictly more general or specific than the target concept. Aggressive Co-Testing uses the weak views both for detecting the most informative examples in the domain and for improving the accuracy of the predictions. In a case study on 33 wrapper induction tasks, our algorithm requires significantly fewer labeled examples than existing state-of-the-art approaches.

1 Introduction

Labeling training data for learning algorithms is a tedious, error prone, time consuming process. Active learning addresses this issue by detecting and asking the user to label only the most informative examples in a domain. In this paper, we focus on Co-Testing [Muslea *et al.*, 2000], an active learning technique for domains with multiple *views*; i.e., domains with disjoint sub-sets of features, each of which is sufficient for learning. Co-Testing is a 2-step iterative algorithm that (1) uses the few available labeled examples to learn a hypothesis in each view and (2) queries (i.e., asks the user to label) examples on which the views predict a different label. Such queries are highly informative because they correct mistakes made by one of the views: whenever the views disagree, at least one of them must be wrong.

Co-Testing was successfully applied to wrapper induction [Muslea *et al.*, 2000], an industrially important application. In wrapper induction the goal is to learn rules that extract the relevant data from collections of Web pages that share the same underlying structure; e.g., extract the book titles and prices from `amazon.com`. For wrapper induction, Co-Testing uses two views: the sequences of tokens that precede and follow the extraction point, respectively. The extraction rules learned in these views are finite automata that consume an item's prefix or suffix within the page, respectively.

The main limitation of existing Co-Testing algorithms [Muslea *et al.*, 2000; 2002a] is that they are designed to use only views that are adequate for learning, thus being unable to also exploit imperfect views that would permit a faster convergence to the target concept. To address this problem, we extend the multi-view learning framework by introducing the idea of learning from *strong* and *weak* views. By definition, a *strong view* consists of features that are adequate for learning the target concept; in contrast, in a *weak view* one can only learn a concept that is more general or specific than the target concept. We introduce a novel algorithm, *Aggressive Co-Testing*, that exploits both strong and weak views without additional data engineering costs. We also describe a case study on wrapper induction, which shows that Aggressive Co-Testing clearly outperforms state-of-the-art algorithms.

To illustrate the idea of strong and weak views, consider the task of extracting fax numbers from a directory of restaurant Web pages such as Zagat. The two wrapper induction views described above are *strong views* because each of them is (typically) sufficient to extract the item of interest [Muslea *et al.*, 2000]. In addition to these two strong views, we can also exploit a view that consists of tokens *within* the item to be extracted. In this view, we learn the grammar "(*Number*) *Number* - *Number*" that describes the content of the fax numbers. This additional view is a *weak view* because the grammar above represents a concept *more general* than the target one; i.e., it cannot discriminate between fax and phone numbers that appear within the same Web page.

Aggressive Co-Testing for wrapper induction works as follows: first, it uses a few labeled examples to learn a rule in each view (i.e., one weak and two strong rules). Then it queries an unlabeled example on which the two strong rules extract *different* strings, *both* of which are inconsistent with the content-based grammar. Each such query is likely to represent a mistake not only in one, but in both strong views, thus leading to faster convergence. We use a collection of 33 difficult extraction tasks to show that using the weak view dramatically reduces the need for labeled data: compared with existing state of the art active learners, our novel algorithm requires between 45% and 81% fewer labeled examples.

2 Related work

The idea of exploiting complementary information sources (i.e., types of features) appears in various multi-strategy

learners. Of particular interest are two recent papers [Kushmerick *et al.*, 2001; Nahm and Mooney, 2000] in which the authors use sets of features that clearly do *not* have the same expressive power. This work can be seen as learning from strong and weak views, even though it was not formalized as such, and it was not used for active learning.

Kushmerick *et al.* [2001] focus on classifying 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. The order of the text lines cannot be used by itself to accurately classify the lines. However, 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].

Another algorithm that can be seen as learning from strong and weak views is DISCOTEX [Nahm and Mooney, 2000], which extracts 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 and 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. For example, it may discover that "`IF the job requirements include 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 the weak view can be used to learn only concepts that are *more specific* than the target concept. Nahm and Mooney show that these mined rules improve the extraction accuracy by capturing information that complements the RAPIER extraction rules.

3 Preliminaries

In this section we first explain the main idea behind Co-Testing algorithms [Muslea *et al.*, 2000; Muslea, 2002], and then we describe the strong and weak views that we use for wrapper induction.

3.1 Background: the Co-Testing approach

Figure 1 provides a formal description of the Co-Testing family of algorithms. Given a base learner $\mathcal{L}$, a set L of labeled examples, and a set U of unlabeled ones, Co-Testing works as follows: first, it learns the classifiers h_1 and h_2 by applying the algorithm $\mathcal{L}$ to the projection of the examples in L onto the two views, **V1** and **V2**. Then it applies h_1 and h_2 to all unlabeled examples in U and detects the set of *contention points*, which are unlabeled examples for which h_1 and h_2 predict a different label. Finally, it asks the user to label one of the contention points and repeats the whole process.

Given: - a base learner $\mathcal{L}$
- a learning problem with features **V**=$\{a_1, a_2, \ldots, a_N\}$
- two views **V1** and **V2**, where **V=V1**$\cup$**V2** and **V1**$\cap$**V2**=$\emptyset$
- the sets L and U of labeled and unlabeled examples
- number N of queries to be made

LOOP for N iterations
- use $\mathcal{L}$, **V1**(L), and **V2**(L) to create classifiers h_1 and h_2
- let $ContentionPoints$ = $\{ u \in U \mid h_1(u) \neq h_2(u) \}$
- IF $ContentionPoints$ is empty THEN quit
- let $Query$ = **SelectQuery**($ContentionPoints$)
- remove $Query$ from U and ask for its label l
- add labeled $Query$ to L

- **CreateOutputHypothesis**(h_1, h_2)

Figure 1: Co-Testing algorithms repeatedly query examples for which the two views make a different prediction.

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*[1] is constructed. In other words, each Co-Testing algorithm is uniquely defined by the choice of the heuristics **SelectQuery()** and **CreateOutputHypothesis()**. In turn, these two heuristics depend on the properties of both the application domain and the base learner $\mathcal{L}$.

We consider here two types of query selection strategies:

- *random*: randomly choose a contention point. This strategy is appropriate for base learners that lack the capability of estimating the confidence of their predictions.
- *max-confidence*: choose the contention point on which both h_1 and h_2 make the most confident prediction. This strategy is appropriate for high accuracy domains (e.g., wrapper induction), in which there is little or no noise. On such tasks, discovering examples that are misclassified "with high confidence" translates into queries that "fix big mistakes," thus leading to fast convergence.

We also consider two "output hypothesis" heuristics:

- *winner-takes-all*: the output hypothesis is the one learned in the view that makes the smallest number of mistakes over the N queries.
- *majority vote*: examples are labeled according to the prediction of most views (requires at least three views).

3.2 Wrapper induction: the strong views

In wrapper induction, each item of interest is described by three strings of variable length: the item's content, together

[1] Once training is completed, the *output hypothesis* is used to predict the label of all new, unseen examples.

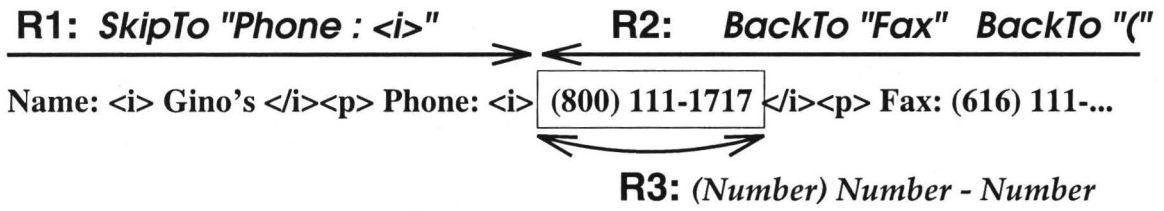

Figure 2: The *forward* and *backward* strong rules (i.e., **R1** and **R2**) find the beginning of the phone number by consuming its suffix or prefix, respectively. **R3** is a content-based grammar that describes the structure of the item to be extracted.

with its prefix and suffix within the document. As this is not a typical machine learning representation in which an example's description in each view consists of a fixed set of features , we describe here in detail how Co-Testing can be applied to wrapper induction. As a first step, we introduce the basic ideas in STALKER [Muslea *et al.*, 2001], which is the wrapper induction algorithm that we use as base learner.

Consider the illustrative task of extracting phone numbers from Web pages similar to the one shown in Figure 2. In STALKER, an *extraction rule* consists of a *start rule* and an *end rule* that identify the beginning and the end of the item, respectively. Given that start and end rules are extremely similar, we describe here only the former. In order to find the beginning of the phone number, one can use the start rule

R1 = SkipTo "`Phone :<i>`"

This rule is applied *forward*, from the beginning of the document, and it ignores everything until it finds the string `Phone :<i>`. For a slightly more complicated extraction task, in which toll-free numbers appear in italics and the other ones in bold, one can use a disjunctive start rule such as

R1' = *either* SkipTo "`Phone :<i>`"
or SkipTo "`Phone :<b>`"

An alternative way to detect the beginning of the phone number is to use the start rule

R2 = BackTo "`Fax`" BackTo "`(`"

which is applied *backward*, from the *end* of the document. **R2** ignores everything until it finds "`Fax`" and then, again, skips back to the first open parenthesis.

As shown in [Muslea *et al.*, 2001], the above extraction rules can be learned based on user-provided examples of items to be extracted. Note that **R1** and **R2** represent descriptions of the *same concept* (i.e., start of phone number) in two *different views*. That is, the views **V1** (*forward view*) and **V2** (*backward view*) consist of the sequences of tokens that *precede* and *follow* the beginning of the item, respectively.

Note that both **V1** and **V2** represent *strong views*: as the Web pages to be wrapped share the same underlying structure, STALKER can be seen as uncovering and exploiting this underlying structure for extraction purposes. Consequently, both the forward and backward rules are expected to extract the relevant data from any page.

3.3 Wrapper induction: the weak view

Besides the two strong views above, one can also use a third, content-based view, which describes the actual item to be extracted. For example, when extracting phone numbers, one may exploit the fact that they can be described by a simple grammar: "(*Number*) *Number* - *Number*". Similarly, when extracting URLs, one can take advantage of the fact that a typical URL starts with the string "`http://www.`", ends with the string "`.html`", and contains no HTML tags.

In this paper, we use the following features to describe the content of each item to be extracted:

- the *length range* (in tokens) of the seen examples. For instance, phone numbers in the format "(*Number*) *Number* - *Number*" consist of six tokens (i.e., the three numbers, the dash, and the two parentheses).
- the *token types* that appear in the training examples. This feature consists of the set of the most specific *wildcards* (e.g., *Number*, *AllCaps*, etc) that match the tokens encountered in the item to be extracted. For example, in the phone number case, this list consists of two wildcards: *Number* and *Punctuation*. The complete hierarchy of wildcards is described in Figure 3.
- a *start-pattern* such as "`http://www.`" or "(*Number*)", which describes the beginning of the item of interest.
- an *end-pattern* such as "*AlphaNum*`.html`" or "*Number* - *Number*", which describes the end of the item.

In order to learn the content-based description of an item, we use as base learner a simplified version of the DataPro algorithm [Lerman and Minton, 2000]. After tokenizing each of the user-provided examples of strings to be extracted, the weak-view learner proceeds as follows:

- the *length range* is determined by finding the examples that contain the largest and the smallest number of tokens;
- the *token types* are obtained by going through the tokens that appear in the labeled examples and adding to the set of "seen types" the most specific wildcard that covers it.
- a start-pattern of *length one* consists of the *most specific* wildcard that covers the first token in *all* labeled examples; if all examples start with the same token, such as "`(`" in the phone number example, the actual token is preferred to the most specific wildcard. A start-pattern of length k is generated by repeating the procedure above for the first, second, ..., up to k-th position.
- the end-pattern is learned in the same manner as the start pattern, but using the k tokens at the end of the item.

Note that, unlike the forward and backward views, the content-based view is a *weak view* because, for many extraction tasks, this view does not uniquely define the item of inter-

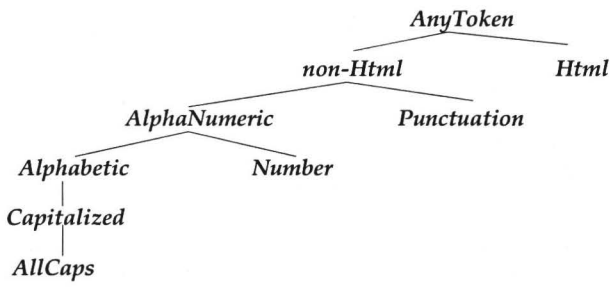

Figure 3: The hierarchy of wildcards used for wrapper induction. The parent-child relationship denotes the *IsMoreGeneralThen* relationship. For example, the most general wildcard is *AnyToken*, which matches all possible tokens. *non-Html*, which is a child of *AnyToken*, denotes all tokens than are not HTML tags (i.e., alphanumeric tokens and punctuation signs).

est. This is a consequence of the fact the view uses only features that describe the content of each item. For Web pages that contain several items with similar descriptions, such as multiple email addresses, phone numbers, URLs, or names, the content-based grammar cannot discriminate between the various items with similar descriptions.

4 Aggressive Co-Testing

We introduce now Aggressive Co-Testing, which provides a framework for naturally exploiting both strong and weak views. For Aggressive Co-Testing, the contention points are defined as unlabeled examples on which the *strong views* predict a different label. For the wrapper induction problem, Aggressive Co-Testing uses the two strong and one weak views described earlier (i.e., the forward, backward, and content-based views). Consequently, the contention points are unlabeled documents from which the forward and backward rules extract different strings. Aggressive Co-Testing uses the labeled examples to learn one hypothesis in each view, detects the contention points, and then uses the following heuristics:

- **SelectQuery()** returns the contention point on which both strong rules violate the largest number of constraints learned in the weak view; e.g., the extracted strings are longer than the seen examples, the start- and end- patterns do not match, etc. This is a *max-confidence* querying strategy because the content-based view is *maximally confident* that the strong rules extract incorrect strings.
- **CreateOutputHypothesis()** uses the three views for *majority voting*. That is, given a new, unseen document, both strong rules are applied to it; if they extract the same string, this string is returned as the answer. Otherwise the "winner" is the strong rule that violates the fewest constraints learned in the weak view. Note that this flexible approach allows Co-Testing to use the most appropriate strong rule for each document in the dataset.

To better understand how Aggressive Co-Testing works, we contrast it now with Naive Co-Testing [Muslea *et al.*, 2000], which uses only the two strong views. Both algorithms detect the contention points in the same manner, but they use different query selection strategies and output hypotheses. More precisely, Naive Co-Testing *randomly* queries one of the contention points and generates a *winner-takes-all* output hypothesis (i.e., the rule that makes the fewest mistakes on the queries extracts the data from *all* documents).

5 Empirical Evaluation

The algorithms in the experimental comparison

In this empirical evaluation we compare the following algorithms: Aggressive Co-Testing, Naive Co-Testing, Query-by-Bagging, and Random Sampling. The first two algorithms were described in the previous section; Random Sampling, which is used as strawman, is identical with Naive Co-Testing, except that it randomly queries one of the unlabeled examples instead of one of the contention points.

Query-by-Bagging [Abe and Mamitsuka, 1998] is the only single-view active learner that can be used in a straightforward manner with STALKER and, more generally, for wrapper induction.[2] Even Query-by-Boosting [Abe and Mamitsuka, 1998], which is similar to Query-by-Bagging, cannot use STALKER as a base learner: as STALKER rarely - if ever - makes mistakes on small training sets, it eliminates the ability of the boosting algorithm to generate a diverse committee.

Query-by-Bagging is based on the idea of creating a *committee* of extraction rules and then querying the example on which the committee is the most split (i.e., the rules in the committee extract the largest number of distinct strings); the algorithm's actual predictions are made by *majority voting* the committee of rules. Query-by-Bagging generates a committee of 10 extraction rules, each of which is learned by training STALKER with examples obtained by *re-sampling with replacement* the original training set. We are forced to use such a small committee because of the scarcity of the training data: as STALKER is expected to train on a handful of examples, sampling-with-replacement from a few examples leads to few distinct training sets for creating the committee. Query-by-Bagging is run once in each view, and we report only the best results, which are obtained in the forward view.

The datasets

In order to empirically compare the algorithms above, we use the wrapper induction testbed introduced by Kushmerick [2000]. It consists of 206 extraction tasks from 30 Web-based information sources.[3] As shown in [Muslea *et al.*, 2001], on most of these 206 tasks STALKER learns 100% accurate rules from just one or two randomly-chosen labeled examples. We consider here the 33 most difficult tasks in the testbed, which were also used in previous work on multi-view learning [Muslea *et al.*, 2000; 2002b]:

- the 28 tasks on which 20 random examples are insufficient for learning 100%-accurate rules in both strong views;

[2]Typical wrapper induction algorithms do *not* have the properties that active learners require of their base learners; e.g., the ability to evaluate the confidence of each prediction [Lewis and Gale, 1994], or to randomly sample hypotheses from the version space [Seung *et al.*, 1992], or to generate *most specific* and *most general* extraction rules [Cohn *et al.*, 1994].

[3]These datasets can be obtained from the RISE repository: `http://www.isi.edu/~muslea/RISE/index.html`.

- the five additional tasks on which, in order to learn 100%-accurate rules in both strong views, STALKER requires a large number of random examples [Muslea, 2002].

The empirical results

For each of these 33 tasks, we use 20-fold cross-validation to compare the performance of the algorithms above. Within each fold, the algorithms start with *the same* two randomly-chosen examples and then make a succession of queries. In the end, the error rate is averaged over the 20 folds.

Figure 4 summarizes the algorithms' performance over the 33 tasks. In each graph, 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 after exactly `X` queries. Each algorithm is allowed to make 18 queries, for a total of 20 labeled examples. *By convention*, the "`19 queries`" data point denotes tasks for which a 100% accurate rule is *not* learned even after 18 queries.

Aggressive Co-Testing clearly outperforms the other algorithms: it makes an average of 2.43 queries over the 30 tasks that are solved with 100% accuracy; furthermore, on 11 of these 30 tasks, a single query is sufficient to learn the correct rule. In contrast, Naive Co-Testing, which comes second, makes an average of 4.4 queries per task and converges in a single query on just four of the 33 tasks. Also note that Aggressive Co-Testing solves correctly two of the five tasks that cannot be solved by Naive Co-Testing; the other two algorithms fail to solve 23 and 26 of the 33 tasks, respectively.

Even though Aggressive Co-Testing makes 45% fewer queries that Naive Co-Testing, at first glance the difference between 2.43 and 4.4 queries per task may seem small. However, one must take into account that wrapper induction is used in information agents [Knoblock *et al.*, 2001], which typically use hundreds of extraction rules; in this context, Aggressive Co-Testing makes a tremendous difference.

To put our work into a larger context, we briefly compare the results above with the ones obtained by WIEN [Kushmerick, 2000], which is the only wrapper induction system for which there are published results for all the extraction tasks used here. As the two experimental setups are not identical,[4] this is just an *informal* comparison that contrasts Co-Testing with a state-of-the-art approach to wrapper induction.

The results in [Kushmerick, 2000] can be summarized as follows: WIEN, which uses random sampling, learns the correct extraction rule on 15 of the 33 task. On these 15 tasks, WIEN requires between 25 and 90 labeled examples[5] to learn the correct rule. For the same 15 tasks, both Aggressive and Naive Co-Testing learn 100% accurate rules from at most eight labeled examples (two random plus at most six queries).

[4]Instead of using cross-validation, WIEN repeatedly splits the dataset into randomly chosen training and test sets.

[5]For WIEN, an example consists of a document in which *all* items of interest are labeled; e.g., a page with 15 labeled names represents a *single* example. In contrast, STALKER counts the 15 labeled strings as 15 examples. We convert the WIEN results into equivalent STALKER-like ones by multiplying the number of WIEN labeled pages by the average number of item occurrences per page.

Discussion

The empirical results deserve several comments. First of all, the experiments illustrate the benefits of a framework that naturally integrates strong and weak views: Aggressive Co-Testing exploits the strengths and mitigates the weaknesses of each individual view. For example, we do *not* use the weak view to identify the contention points because its mistakes may be "unfixable" (remember that in a weak view one learns a concept more general/specific than the one of interest). On the other hand, we use the weak view both to detect the highly informative contention points and to find the most appropriate strong view for each prediction.

In contrast to Aggressive Co-Testing, existing multi-view learners [Blum and Mitchell, 1998; Muslea *et al.*, 2000] can use only the strong views, thus losing an important source of information. Similarly, single-view learners must either pool all features together or simply ignore all but one view. Note that, in practice, pooling the features together may not be a straightforward task:

- in DISCOTEX [Nahm and Mooney, 2000], the text mining features (the weak view) are the extracted items, which become available only after the extraction rules are learned and applied to the unlabeled corpus.
- the main contribution of [Kushmerick *et al.*, 2001] consists of a novel algorithm that exploits features from both the strong and weak views (i.e., the words in each text line, and the lines's order within the business card).

Second, we ran an additional experiment to determine the usefulness of the weak view with respect to each heuristic (i.e., query selection and output hypothesis). We considered two hybrid algorithms between Aggressive and Naive Co-Testing: one uses the *random* and *majority vote* heuristics, while the other uses *max-confidence* and *winner-takes-all*; i.e., each hybrid exploits the weak view in only one of the two heuristics. Because of space constraints, we just summarize our findings: the two hybrids outperform Naive and underperform Aggressive Co-Testing; more precisely, they make an average of 3.0 and 3.9 queries per task, respectively. In other words, the weak view improves both the query selection and the output hypothesis.

Finally, note that on three of the 33 tasks, Aggressive Co-Testing fails to learn 100%-accurate rules; in fact, on these tasks Query-by-Bagging and Random Sampling obtain more accurate rules than both Aggressive and Naive Co-Testing. This happens because, on these three tasks, the *backward* view is significantly less accurate than the *forward* one. Consequently, the distribution of the queries is skewed towards mistakes of the "bad view", which are not informative for either view: the "good view" extracts the correct string anyway, while the "bad view" is inadequate to learn the target concept. To cope with this problem, we plan to use view validation [Muslea *et al.*, 2002b], which predicts whether or not the strong views are appropriate for a particular task.

6 Conclusion and Future Work

In this paper we introduce the concepts of strong and weak views and present a novel active learner that naturally integrates and exploits both types of views. In a case study on

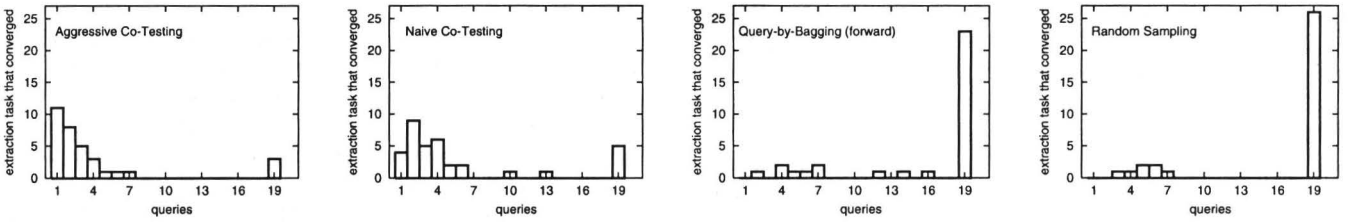

Figure 4: Convergence results on the 33 wrapper induction tasks.

wrapper induction, we show that weak views represent a powerful source of information that can be used both to detect highly informative examples and to improve the algorithm's predictions. On a set of 33 difficult extraction tasks, our novel algorithm converges by making up to 81% fewer queries than other state of the art active learners.

We intend to continue our work along several directions. First, we plan to investigate the use of weak views for semi-supervised multi-view learning [Blum and Mitchell, 1998]. Second, we intend to extend view validation [Muslea *et al.*, 2002b] so that it also accounts for weak views. Finally, we are interested in a theoretical analysis of learning from strong and weak views.

Acknowledgments

This material is based upon work supported in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197 and F30602-00-1-0504, in part by the Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and F49620-02-1-0270, in part by the United States Air Force under contract number F49620-02-C-0103, and in part by a gift from the Microsoft Corporation. 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 and Mamitsuka, 1998] Naoki Abe and Hiroshi Mamitsuka. Query learning using boosting and bagging. In *Proceedings of ICML-98*, pages 1–10, 1998.

[Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of COLT-98*, pages 92–100, 1998.

[Califf and Mooney, 1999] Mary Elaine Califf and Raymond Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of AAAI-99*, pages 328–334, 1999.

[Cohn *et al.*, 1994] David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15:201–221, 1994.

[Knoblock *et al.*, 2001] Craig Knoblock, Steven Minton, Jose-Luis Ambite, Naveen Ashish, Ion Muslea, and Andrew Philpot. The Ariadne approach to Web-based Information Integration. *International Journal of Cooperative Information Sources*, 10(1/2):145–169, 2001.

[Kushmerick *et al.*, 2001] Nicholas Kushmerick, Edward Johnston, and Stephen McGuinness. Information extraction by text classification. In *IJCAI-2001 Workshop on Adaptive Text Extraction and Mining*, 2001.

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

[Lerman and Minton, 2000] Kristina Lerman and Steven Minton. Learning the common structure of data. In *Proceedings of AAAI-2000*, pages 609–614, 2000.

[Lewis and Gale, 1994] David Lewis and William Gale. A sequential algorithm for training text classifiers. In *Proceedings of Research and Development in Information Retrieval*, pages 3–12, 1994.

[Muslea *et al.*, 2000] Ion Muslea, Steven Minton, and Craig Knoblock. Selective sampling with redundant views. In *Proccedings of AAAI-2000*, pages 621–626, 2000.

[Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig Knoblock. Hierarchical wrapper induction for semistructured sources. *Journal of Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.

[Muslea *et al.*, 2002a] Ion Muslea, Steven Minton, and Craig Knoblock. Active + Semi-supervised Learning = Robust Multi-view Learning. In *Proceedings of ICML-2002*, pages 435–442, 2002.

[Muslea *et al.*, 2002b] Ion Muslea, Steven Minton, and Craig Knoblock. Adaptive view validation: A first step towards automatic view detection. In *Proceedings of ICML-2002*, pages 443–450, 2002.

[Muslea, 2002] Ion Muslea. *Active Learning with Multiple Views*. PhD thesis, Department of Computer Science, University of Southern California, 2002.

[Nahm and Mooney, 2000] Un-Yong Nahm and Raymond Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of AAAI-2000*, pages 627–632, 2000.

[Seung *et al.*, 1992] H. Sebastian Seung, Manfred Opper, and Haim Sompolinski. Query by committee. In *Proceedings of COLT-1992*, pages 287–294, 1992.