XML Document Classification with Co-training

Xiaobing Jemma Wu

CSIRO ICT Centre, Australia

Abstract. This paper presents an algorithm of using Co-training with the precision/recall-driven decision-tree algorithm to handle the labeled-unlabeled problem of XML classification. The two views are generated using a predicate rewrite system mechanism which is built on a higher-order logic representation formalism. Experimental results show that this method performs well on classifying XML documents using only a few labeled examples and a large number of unlabeled examples.

Key words: Co-training, XML documents, predicate rewrite systems, decision-tree learning.

1 Introduction

One key weakness of the supervised learning is that it usually requires a large, often prohibitive, number of labeled training examples to learn accurately. However, example labelling has typically to be done by a human or sometimes an expert, and it is a painfully time-consuming task. On the contrary, unlabeled data are often abundant and easy to get. This is particularly true for online data sources such as XML documents. Early studies [14, 11, 16] have proved theoretically and empirically that unlabeled data is useful in helping improve the classification performance.

Blum and Mitchell [2] proposed the Co-training algorithm for data with two separate views. For example, with a web page classification problem, one view is the words in the web page itself, and the other is the words on the hyperlinks pointing to this page. The idea behind Co-training is that one classifier can add examples that are easy for it to classify and that provide useful information to improve the accuracy of the other classifier. The Co-training algorithm assumes that each of the two views contain information sufficient to classify the examples and these two views are conditionally independent given the classification. Co-training has been proved successful in making using of unlabeled data in classification [12, 8, 15].

The XML document classification is a natural application of the Co-training algorithm. An XML document consists of multiple substructures - elements, and these elements could and normally does contain redundancy information for classification. For example, in the Reuters dataset [1], element TITLE and element BODY can generate two redundantly sufficient views.

2 Knowledge representation for XML

The knowledge representation formalism used in this paper is based on a typed higher-order logic. Its basic principle is that an individual should be represented as a (closed) term [9]. This representation method is particularly suitable for structured data as all the information about an individual is stored in one place and the structure of the term provides strong guidance on the search for a suitable hypothesis.

2.1 Representation of XML documents

A well-formed XML document has a well-nested structure. We developed a complete representation for a well-formed XML document with the above formalism [19]. Here, we only give the representation of the top-level XML document as an example.

$$\mathit{type\ XML} = \mathit{XMLDecl} \times \mathit{Misclist} \times \mathit{DTD} \times \mathit{Misclist} \times \mathit{Element} \times \mathit{Misclist}$$

An XML document is represented as a 6-tuple type, with the first component representing the XML declaration and the second, fourth and sixth representing a list of miscellaneous stuff, and the third component representing the document type declaration, and the fifth representing the root element. All thesis six types are non-atomic types, but only type *Element* is recursive.

2.2 Predicate construction using predicate rewrite systems

A novel feature construction mechanism is presented in [9] accompanying the higher-order logic representation for individuals. Predicates are built up incrementally by composition of simpler functions called transformations. A transformation f is a function having signature $f : (\varrho_1 \rightarrow \Omega) \rightarrow \ldots \rightarrow (\varrho_k \rightarrow \Omega) \rightarrow S \rightarrow T$, where $\varrho_1, \ldots, \varrho_k, S$ and T are all types and $k \geq 0$. For example,

$$\mathit{projRootElement} : \mathit{XML} \rightarrow \mathit{Element}$$

is a transformation projecting an XML document onto its root element.

The predicate rewrite system is a mechanism to define and enumerate a set of predicates relevant to a particular application by using the transformations. A predicate rewrite system is a finite relation $\rightarrowtail$ on the set of all standard predicates. A predicate rewrite is in the form of $p \rightarrowtail q$ where p and q are two types and p is more general than q. In applications, to generate a search space of predicates, we start from some initial predicate p_0 (normally the weakest predicate *top*) and generate all the predicates that can be obtained by a predicate deviation step from p_0, then all the predicates that can be obtained from those by a predicate derivation step, and so on. A predicate rewrite system defines a predicate search space. For example, the following simple predicate rewrite system

$$\begin{aligned}&\mathit{top} \rightarrowtail \mathit{projRootElement} \circ \mathit{top}\\ &\mathit{top} \rightarrowtail \wedge_2\ (\mathit{projTagName} \circ \mathit{top}\ \mathit{projContents} \circ \mathit{top})\end{aligned}$$

could generate the search space of

$$\begin{aligned}&\mathit{top}\\ &\mathit{projRootElement} \circ \mathit{top}\\ &\mathit{projRootElement} \circ\ \wedge_2\ (\mathit{projTagName} \circ \mathit{top}\ \mathit{projContents} \circ \mathit{top})\end{aligned}$$

3 Class Probability Trees

Decision trees which output probabilistic decisions are called statistical decision-trees or class probability trees [3, 4, 10, 5]. Theoretically, supposing $|D|$ examples $\{d_1, \ldots, d_{|D|}\}$ which are assigned to $|C|$ classes $\{c_1, \ldots, c_{|C|}\}$, probabilistic trees return the posterior probability of each class given a new example d_{new} and a tree classifier T, i.e., $Pr(c_i|d_{new}, T)$, $i = 1, \ldots, |C|$.

The most obvious way of determining class probabilities of a decision tree is to compute them directly from the counts of training examples at the leaf nodes. That is, if an example that is being classified ends up at a leaf node which has 100 examples (90 positive ones and 10 negative ones) from the training set, then the probability of being positive is 0.9. The disadvantage of this approach is that the number of examples at each leaf is often too small to reliably compute the probability and the distribution of the training examples at the leaf can not reflect the distribution of the training set.

Many other approaches have been proposed to construct class probability trees. [17] proposes an approach to refine the class probability estimates in a greedily induced decision tree using local kernel density estimates. Johnson et al. [7] uses a tree-shrinking method [6] to get a probabilistic decision tree. A context tree weighting method is proposed in [18] which has been borrowed by other researchers for producing class probability trees [7].

Buntine [4] uses a Bayesian approach for tree smoothing to improve the naive probabilities obtained from the leaves only. Smoothing is to compute the class probabilities for a leaf by combining the leaf probabilities with the node probabilities on the path from the root down to this leaf. The posterior probability of class c_i given the new example d_{new} and the decision tree T takes the form of an average calculated along the branch traversed by the new example

$$Pr(c_i|d_{new}, T) = \sum_{N \in path(d_{new}, T)} Pr(node(N)|D, C, T) Pr(c_i|leaf(N); D, C, T),$$

where $Pr(node(N)|D, C, T)$ is the node probability of the node N, and $Pr(c_i|leaf(N); D, C, T)$ is the local probability of c_i in node N, i.e., the leaf probability. The node probability of each internal node is computed recursively from the root node. The leaf probability of each leaf is computed via the Laplace correction of the naive probability of the class probability in each leaf.

With the smoothing method, the probabilistic tree is obtained via two steps. The first step transfers a binary decision tree into an intermediate tree that computes the node probability and leaf probability for each internal node and the leaf probability for each leaf. The second step transfers the intermediate tree to the final class probability tree that has the final class probability for each class in each leaf.

The PRDT (Precision/recall-driven Decision Tree) algorithm [19] is modified using this smoothing method to get the *SmoothPRDT* algorithm which output probabilistic decisions. This new algorithm is shown in Figure 1.

```
function $SmoothPRDT(\mathcal{E}, \rightarrowtail)$; returns: a probabilistic decision tree;
inputs: $\mathcal{E}$, a set of examples;
        $\rightarrowtail$, a predicate rewrite system;


$T := PRDT(\mathcal{E}, \rightarrowtail)$;
transfer $T$ to an intermediate tree $T^{'}$ by computing the node probability and leaf
probability for each node in $T$;
transfer $T^{'}$ to the final tree $T^{''}$ by combining the leaf probability with the node
probability for each node on the path from the root down to the leaf;
return $T^{''}$;
```

Fig. 1. The probabilistic decision-tree algorithm by smoothing the PRDT tree

4 XML classification with Co-training

4.1 The Co-training framework

Co-training [2], invented by Blum and Mitchell, is a new strategy for using unlabeled data which has two seperate and redundant views. For example, with a web page classification problem, one view is the words in the web page itself, and the other is the words on the hyperlinks pointing to this page. Two classifiers induced from the two views are built incrementally through an iteration. Each classifier is initialised with its corresponding feature set of the labeled data only. In each iteration, each classifier labels the unlabeled data and add the most confidently labeled data into the training set. The two classifiers are rebuilt in each iteration with the updated training set.

The basic idea behind the co-training framework is to exploit the compatibility between different views on an example. In the traditional supervised learning environment, we have an individual space X, a target function $f : X \to Y$, and labeled examples $\{< x_i, f(x_i) >\}$. In the co-training setting, we assume such a supervised learning problem in which the individuals are drawn from X according to some fixed probability distribution. We further assume the individual space $X = X_1 \times X_2$, where X_1 and X_2 correspond to two different "views" of an example, and C_1 and C_2 be concept classes defined over X_1 and X_2. That is, each example x in X can be factored into two sets of features (x_1, x_2). The co-training paradigm requires that X_1 and X_2 each contain information sufficient to classify the example and these two views are compatible. By compatible, it means that there exists some function $f_1 \in C_1$ and some function $f_2 \in C_2$ such that for all $x \in X$, $f(x) = f_1(x) = f_2(x)$. If the above constraints on X_1 and X_2 are satisfied, then X_1 and X_2 are said to be *redundantly sufficient* to classify X with respect to f. It also requires that x_1 and x_2 are conditionally independent given the classification. In a word, two assumptions exist for the co-training framework: there are two distinct "views" of an example, each of which is sufficient for classification, and the two views are conditionally independent given the class label.

There are many applications which have natural feature splits, i.e., multiple "views". In [2, 11], Blum and Mitchell are interested in classification of web pages, and they suggest to describe a web page from two different views: the words on the page itself, and the words in all hyperlinks pointing to this page. Other applications include classifying noun phrases into semantic classes in which one view is the noun phrase itself and the other is the linguistic context of this phrase, and the problem of object recognition in multimedia data in which the two views are the video and audio signals.

Under the above theoretical setting, the goal of the Co-training model is to boost the performance of a weak classifier using the unlabeled data (Figure 2).

input: a sample $D = D_l \cup D_u$ where

$D = \{(x_i = (x_{1,i}, x_{2,i}), y_i) | 1 \leq i \leq m\}$
$D_{l,1} = \{(x_{1,i}, y_i) | 1 \leq i \leq m\}$
$D_{l,2} = \{(x_{2,i}, y_i) | 1 \leq i \leq m\}$
$D_u = \{x_i = (x_{1,i}, x_{2,i}) | m + 1 \leq i \leq n\}$

output: a function $h = (h_1, h_2)$ such that:

1. $h_1(x_{1,i}) = h_2(x_{2,i}) = y_i$ for $i = 1, \ldots, m$;
2. $h_1(x_{1,i}) = h_2(x_{2,i})$ for $i = m + 1, \ldots, n$.

Fig. 2. The objective of Cotraining model

The intuition behind the Co-training algorithm is that it may be easier for one learner to identify an example and this example may provide useful information to the other learner. Therefore, the basic idea of the Co-training algorithm is one learner inclemently trains on the other learner's classification of unlabelled examples. The Co-training algorithm works in an iterative manner. Given a small set of labeled data, D_l, and a large set of unlabeled data, D_u, the learner updates a set of labeled data, D_w, and two hypothesis h_1 and h_2 on X_1 and X_2, respectively. Initially, $D_w = D_l$. In each iteration, train hypothesis h_1 on the X_1 of D_w and hypothesis h_2 on the X_2 of D_w. Use h_1 to label the unlabeled data in D_u and select the examples that most confidently labeled and add them into D_w. Same operations with h_2. This process repeats until certain conditions are satisfied or all the data in D_u has been moved to D_w. The final hypothesis is the combination of h_1 and h_2.

4.2 Generate views using predicate rewrite systems

Now we can take the advantage of the flexible control of the feature space via the predicate rewrite system and create multiple feature subsets. $S_{\rightarrowtail}$ is defined as the set of all predicates that can be obtained from a predicate rewrite system $\rightarrowtail$, starting from some initial predicate. Figure 3 shows the view generating process via predicate rewrite systems.

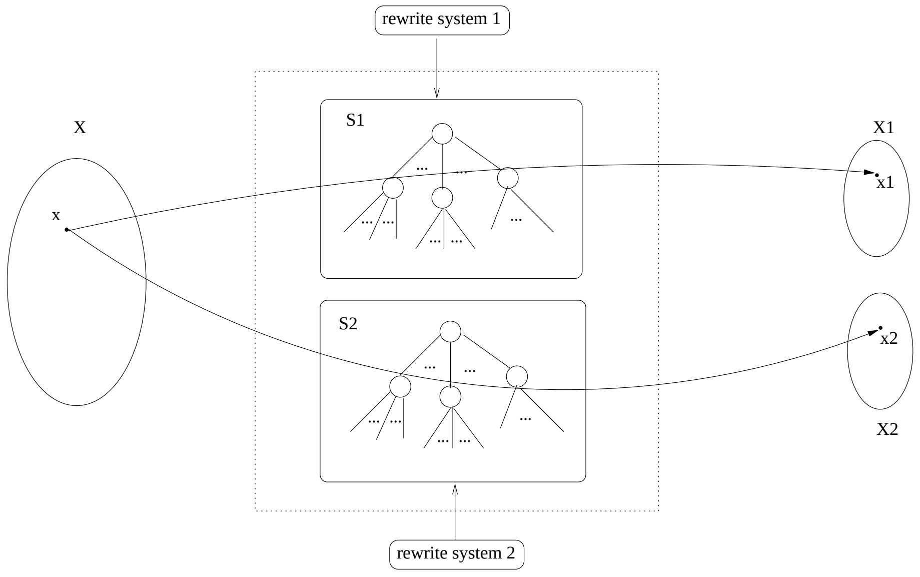

Fig. 3. Two predicate rewrite systems map the individual space onto two views

The system is given two different predicate rewrite systems $\rightarrowtail_1$ and $\rightarrowtail_2$ which generate two predicate search spaces $S_{\rightarrowtail_1}$ and $S_{\rightarrowtail_2}$, respectively. The individual space X is mapped onto two subspaces X_1 and X_2 by $S_{\rightarrowtail_1}$ and $S_{\rightarrowtail_2}$, respectively, and an individual $x \in X$ is mapped onto two views $(x_1, x_2) \in$

$X_1 \times X_2$ correspondingly. X_1 and X_2 can be easily changed by slightly modifying $\rightarrowtail_1$ and $\rightarrowtail_2$. If x is an XML document, then x_1 and x_2 are two substructures withdrawn by $S_{\rightarrowtail_1}$ and $S_{\rightarrowtail_2}$.

Unlike the previous investigation of Co-training [13, 12, 8] in which the two views are fixed and set up before running, the two views (x_1, x_2) here are only created on the run during learning. Predicate rewrite systems $\rightarrowtail_1$ and $\rightarrowtail_2$ can be easily modified to change the two views X_1 and X_2 without going deep inside the data and dig out different splits of features.

How to choose the two predicate rewrite systems is a key point to the success of the Co-training. It is an open question how to generate two conditionally independent views for an individual. The general method is first generate a complete predicate rewrite system $\rightarrowtail$ for a particular application. Then identify two parallel substructures (elements) E_1 and E_2 that could contain sufficient information for classification. Generate predicate rewrite system $\rightarrowtail_1$ by removing the predicate rewrite that generates predicates on E_1. Similarly, generate $\rightarrowtail_2$ by removing the predicate rewrite that generates predicates on E_2. In this way, we actually screen E_1 in X_1 and E_2 in X_2. This idea is best illustrated with an example.

In the XML version of Reuters dataset, a document contains about 12 elements. The following two predicate rewrite systems generate two separate feature spaces corresponding to the text contents in element TITLE and element BODY.

The first predicate rewrite system $\rightarrowtail_1$ is as follows.

$top \rightarrowtail projRootElement \circ top$

$top \rightarrowtail \wedge_2\ (projTagName \circ top\ projContents \circ\ top$

$top \rightarrowtail listToSet \circ top$

$top \rightarrowtail setExists1\,(\wedge_2(isElement\ projContentElement \circ top))$

$top \rightarrowtail setExists1\,(\wedge_2(isFeature\ projContentFeature \circ top))$

$top \rightarrowtail setExists1\,(\wedge_2(proFeaturePos \circ top\ projFeatureWeight \circ top))$

$top \rightarrowtail (= REUTERS)$

$\dots$

$top \rightarrowtail (= TITLE)$

$top \rightarrowtail (= 0)$

$\dots$

$top \rightarrowtail (= 299)$

$top \rightarrowtail (\geq 0.0)$

$\dots$

$top \rightarrowtail (\leq 1.0)$

The second predicate rewrite system $\rightarrowtail_2$ is obtained by replacing $top \rightarrowtail (= \mathit{TITLE})$ by $top \rightarrowtail (= \mathit{BODY})$.

$$
\begin{aligned}
&top \rightarrowtail projRootElement \circ top\\
&top \rightarrowtail \wedge_2\ (projTagName \circ top\ projContents \circ\ top\\
&top \rightarrowtail listToSet \circ top\\
&top \rightarrowtail setExists1\,(\wedge_2(isElement\ projContentElement \circ top))\\
&top \rightarrowtail setExists1\,(\wedge_2(isFeature\ projContentFeature \circ top))\\
&top \rightarrowtail setExists1\,(\wedge_2(proFeaturePos \circ top\ projFeatureWeight \circ top))\\
&top \rightarrowtail (= REUTERS)\\
&\ldots\\
&top \rightarrowtail (= BODY)\\
&top \rightarrowtail (= \mathit{0})\\
&\ldots\\
&top \rightarrowtail (= \mathit{299})\\
&top \rightarrowtail (\geq \mathit{0.0})\\
&\ldots\\
&top \rightarrowtail (\leq \mathit{1.0})
\end{aligned}
$$

4.3 The CotrainPRDT algorithm

Now we introduce our CotrainPRDT algorithm which is a combination of the Co-training algorithm and the PRDT algorithm [19]. The PRDT algorithm is an decision-tree algorithm driven by precision and recall criteria. It is specially designed for XML document classification. The PRDT algorithm takes two parameters, a set of training examples and a predicate rewrite system $\rightarrowtail$, as inputs and outputs a decision-tree. Since the Co-training framework requires decision confidence on unlabeled data, we replace the PRDT algorithm with its probabilistic version of *SmoothPRDT* which is described in Section 3.

Figure 4 gives the CotrainPRDT algorithm.

Given a set D^l of labeled examples, a set D^U of unlabeled examples, and two different predicate rewrite systems $\rightarrowtail_1$ and $\rightarrowtail_2$, the algorithm first creates a smaller pool D^u by randomly selecting u examples from D^U. Blum and Mitchell [2] found that the Co-training algorithm can obtain better results when using a smaller pool D^u from D^U, and we adopt this idea here. The algorithm then enters an iteration. First, use SmoothPRDT algorithm to learn a tree T_1 with D^l and $\rightarrowtail_1$, and a tree T_2 with D^l and $\rightarrowtail_2$. Second, allow each of these two classifiers to label the unlabeled examples in D^u and select the p examples it most confidently labels as positive, and the n examples it most confidently labels as negative. Normally, p and n are set to match the ratio of positive to negative examples in the underlying data distribution. These two sets of $p+n$ newly labeled examples could be contradictably labeled by the two classifiers, and these contradictably labeled examples will surely hurt the classifiers performance if added into the labeled data. Therefore only the non-contradictably newly labeled examples are added, along with the label assigned, into D^l, and removed from D^u. Finally, the pool D^u is replenished to u examples by randomly drawing examples from D^U.

```
**function** $CotrainPRDT(D^l, D^u, \rightarrowtail_1, \rightarrowtail_2)$; **returns**: two decision trees;
**inputs**: $D^l$, a set of labeled examples;
        $D^U$, a set of unlabeled examples;
        $\rightarrowtail_1$ and $\rightarrowtail_2$, two independent predicate rewrite systems;

Create a smaller pool $D^u$ by randomly select a certain number of unlabeled exam-
ples from $D^U$;
**while** $D^u \neq \phi$ **do**
    $T_1 := SmoothPRDT(D^l, \rightarrowtail_1)$;
    $T_2 := SmoothPRDT(D^l, \rightarrowtail_2)$;
    use $T_1$ to label the unlabeled data in $D^u$;
    use $T_2$ to label the unlabeled data in $D^u$;
    select $p$ positive and $n$ negative most confidently labeled data by $T_1$ and
    $T_2$ and add the non-contradictable ones to $D^l$;
    refill $D^u$ by random data from $D^U$;

return two decision trees $T_1$ and $T_2$ whose predications are combined when
classifying new data;
```

Fig. 4. The CotrainPRDT algorithm

5 Experimental Results

We have evaluated our CotrainPRDT algorithm on two XML datasets. One is a semi-artificial one that satisfies both of the Co-training assumptions. The other one is a real-world dataset that partly satisfies the Co-training assumptions.

5.1 The Reuters 2x2 dataset

The Reuters 2x2 dataset is a modified subset of Reuters collection [1], created to satisfy the two assumptions of the Co-training setting, with the similar method in [12].

Four classes of documents that have high performance with PRDT algorithm were selected from the Reuters collection, which are Earn, Acq, Grain and Crude.

We used the positive documents of Grain and Crude to create the positive documents in the new semi-artificial dataset, and the positive documents of Earn and Acq to create the negative documents in the new semi-artificial dataset. In a newly created XML document, the contents of element TITLE and element BODY come from two XML documents of two different classes correspondingly. Thus, the two elements are conditionally independent.

There are 1872 XML documents in total being created, with 385 in the positive class and 1487 in the negative class. Each of the two views generated by $\rightarrowtail_1$ and $\rightarrowtail_2$ is sufficient to get a good classifier (with 96.1% and 96.5% at BEP, respectively). Therefore, Reuters 2x2 satisfies the Co-training requirements.

Five fold experiments were done and the average results are reported here. In each fold, we randomly select 468 examples as test set, 12 (3 positive and 9 negative) as labeled set (D^l), the rest 1392 as unlabeled set (D^U). At the start of each experiment, 75 unlabeled documents are withdrawn randomly from D^U to form the smaller pool D^u.

Figure 5 shows the averaged results on Reuters 2x2 using CotrainPRDT with different numbers of iterations. Figure 6 shows the behaviour of the two classifiers and the combined classifier in one fold experiment. The two classifiers boost each other during the iteration and the performance of the combined classifier increases as that of the two classifiers increases.

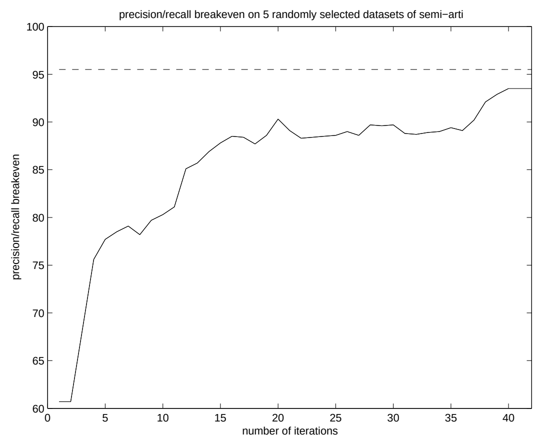

Fig. 5. the performance of CotrainPRDT on the Reuters 2x2 dataset

5.2 The real-world Reuters dataset

The real-world dataset is the XML version of the Reuters dataset [1]. We did experiments on three classes: Earn, Acq and Grain. Five fold experiments were

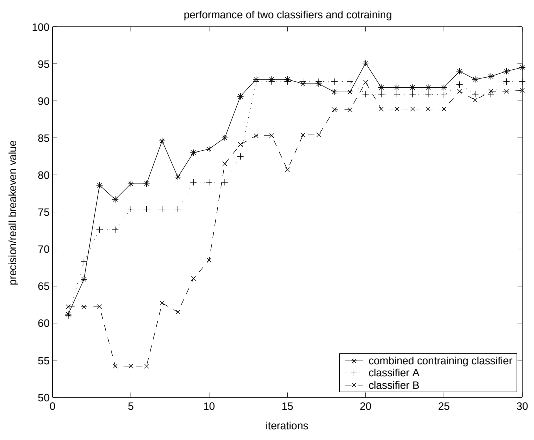

Fig. 6. the performance of the two classifiers from the two views and the combined classifier in one fold experiment on Reuters 2x2 dataset.

done on each class. In each fold, we randomly select 329 examples as test set, 12 (3 positive and 9 negative) as labeled set D^l, and the rest 9591 as unlabeled set D^U. D^u is set to 75 random examples from D^U.

In the experiments, the predicate rewrite systems $\rightarrowtail_1$ and $\rightarrowtail_1$ generate two views corresponding to element TITLE and element BODY. However, unlike the Reuters 2x2 dataset, the two views here are not conditionally independent given the class.

Figure 7 shows the curves of the performance on the three classes. The CotrainPRDT works very efficient on class Earn, but not very well on class Acq and Grain. We executed PRDT on each of the two views for the three classes, and found that both views of class Earn are strong enough to get a good classifier, with 86.4% and 94.3% at BEP, respectively. However, one of the two views of both class Acq and Grain is not strong enough to get a good classifier. For class Acq, the BEP values one two views are 63.0% and 85.5%, respectively, and for class Grain, they are 69.7% and 86.9%, respectively.

6 Conclusions

This paper has presented an algorithm which combines the Co-training strategy with a decision-tree algorithm driven by precision/recall breakeven point. The two views of the Co-training are created using two different predicate rewrite systems that are built on a novel higher-order logic representation method for XML documents. Experimental results show that the CotrainPRDT algorithm can be used to successfully classification XML documents using a few labeled examples together with a large number of unlabeled example, provided that there are two sufficient views with the documents.

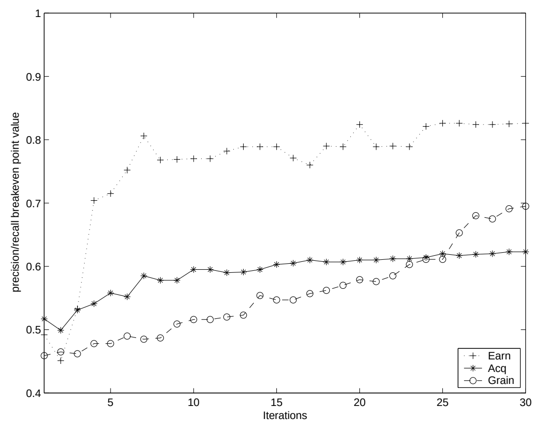

Fig. 7. The curves of performance of CotrainPRDT on 3 classes of Reuters XML dataset vs iterations

References

1. *Reuters-21578.* http://www.daviddlewis.com/resources/testcollections/reuters21578/.
2. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*, pages 92–100. Morgan Kaufmann Publishers, 1998.
3. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth International Group, 1984.
4. W. Buntine. Learning classification trees. *Statistics and Computing*, 2:63–73, 1992.
5. O. Gur-Ali and W.A. Wallace. Induction of rules subject to a quality constraint: Probabilistic inductive learning. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):979–984, December 1993.
6. T. Hastie and D. Pregibon. Shrinking trees. *AT&T technical memorandum*, 1990.
7. D.E. Johnson, F.J. Oles, T.Zhang, and T.Goetz. A decision-tree-based symbolic rule induction system for text categorization. *IBM Systems Journal*, 41(3), 2002.
8. S. Kiritchenko and S. Matwin. Email classification with co-training. In *Proceedings of the 2001 conference of the Centre for Advanced Studies on Collaborative research*, 2001.
9. J. W. Lloyd. *Logic for Learning: Learning Comprehensible Theories from Structured Data.* Springer-Verlag, 2003.
10. D. M. Magerman. *Natural language parsing as statistical pattern recognition.* PhD thesis, Stanford University, 1994.
11. T. Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the Sixth International Colloquium of Cognitive Science.*, San Sebastian, Spain, 1999.
12. K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the Ninth International Conference on Information and Knowledge Management*, pages 86–93, McLean, Virginia, United States, 2000.

13. K. Nigam and R. Ghani. Understanding the behavior of co-training. In *Proceedings of KDD*, 2000.
14. K. P. Nigam. *Using Unlabeled Data to Improve Text Classification.* PhD thesis, Carnegie Mellon University, May 2001.
15. B. Raskutti, H. Ferra, and A. Kowalczyk. Combining clustering and co-training to enhance text classification using unlabelled data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 620 – 625, Edmonton, Alberta, Canada, 2002. ACM Press.
16. M. Seeger. Learning with labeled and unlabeled data. Technical report, Institute for Adaptive and Neural Computation, University of Edinburgh, 2001.
17. P. Smyth, A. Gray, and U. M. Fayyad. Retrofitting decision tree classifiers using kernel density estimation. In *Proceedings of the 1995 Conference on Machine Learning.* Morgan Kaufman, 1995.
18. F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalkens. The context-tree weighting method: Basic properties. *IEEE Transactions on Information Theory*, IT-41:653–664, May 1995.
19. X. Wu. Knowledge representation and inductive learning with XML. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 491–495, Beijing, China, 2004.