Researcher Homepage Classification using Unlabeled Data

Sujatha Das G.[1,3], Cornelia Caragea[1,4], Prasenjit Mitra[2,3], C. Lee Giles[2,3]
[1]Department of Computer Science and Engineering
[2]School of Information Sciences and Technology
[3]The Pennsylvania State University, University Park, PA-16803
[4]University of North Texas, Denton, TX-76203
gsdas@cse.psu.edu, ccaragea@unt.edu, pmitra@ist.psu.edu, giles@ist.psu.edu

ABSTRACT

A classifier that determines if a webpage is relevant to a specified set of topics comprises a key component for focused crawling. Can a classifier that is tuned to perform well on training datasets continue to filter out irrelevant pages in the face of changed content on the Web? We investigate this question in the context of researcher homepage crawling.

We show experimentally that classifiers trained on existing datasets for homepage identification underperform while classifying "irrelevant" pages on current-day academic websites. As an alternative to obtaining datasets to retrain the classifier for the new content, we propose to use effectively unlimited amounts of unlabeled data readily available from these websites in a co-training scenario. To this end, *we design novel URL-based features* and use them in conjunction with content-based features as complementary views of the data to obtain remarkable improvements in accurately identifying homepages from the current-day university websites.

In addition, *we propose a novel technique for "learning a conforming pair of classifiers"* using mini-batch gradient descent. Our algorithm seeks to minimize a loss (objective) function quantifying the difference in predictions from the two views afforded by co-training. We demonstrate that tuning the classifiers so that they make "similar" predictions on unlabeled data strongly corresponds to the effect achieved by co-training algorithms. We argue that this loss formulation provides insight into understanding the co-training process and can be used even in absence of a validation set.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Miscellaneous

General Terms

Algorithms

Keywords

co-training, consensus maximization, gradient descent

1. MOTIVATION

Professional homepages of researchers, which typically summarize research interests, publications and other metadata related to researchers, are shown to be rich sources of information for digital libraries [12]. Researchers' homepages (also referred to as academic homepages or simply homepages in this paper) have been successfully employed in tasks such as expertise search [2], extraction of academic networks, author profile extraction and disambiguation [40] as they provide *crucial* evidence for improving these tasks in digital libraries.

Furthermore, digital library systems such as CiteSeer[1], ArnetMiner[2], and Google Scholar[3] are primarily interested in obtaining and tracking researchers' homepages in order to retrieve appropriate scientific research publications. Given the infeasibility of collecting the entire content on the Web, a focused crawler aims to minimize the use of network bandwidth and hardware by selectively crawling only pages relevant to a (specified) set of topics [7]. A key component for such a crawler is a classification module that identifies whether a webpage being accessed during the crawl process is potentially useful to the collection. For digital libraries, the "yield" of such crawlers *highly* depends on the accuracy of researcher homepage classification.

Supervised methods for learning homepage classifiers rely on the availability of large amounts of labeled data. A widely used labeled dataset for webpage classification is the WebKB dataset[4] that is built in 1997. However, due to recent changes in the information content on academic websites, this dataset is becoming outdated. For example, there are now pages on academic websites that are related to various activities such as invited talks, news, events that do not occur in the WebKB dataset. We refer to university, department and research center websites as "academic websites" in this paper. Compared to few decades back, it is easier now to find faculty information, links to their homepages, information on research groups, course related notes and documents, and research papers from academic websites. Similarly, job postings, seminar announcements and notices are also being uploaded onto departmental websites in recent times [35].

How can a homepage classifier keep up in the face of rapidly changing types of pages on the Web? Specifically, given a classifier that identifies homepages with reasonable accuracy (as measured on the training datasets), how does it perform in the potentially different deployment environment? Semi-supervised methods that can exploit large amounts of unlabeled data together with limited amounts of labeled data for learning accurate classifiers have received significant attention in recent research in machine learning due to

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

[1] http://citeseerx.ist.psu.edu
[2] http://arnetminer.org/
[3] http://scholar.google.com/
[4] http://www.cs.cmu.edu/∼webkb/

the fact that labeling examples for any supervised learning problem requires intensive human labor [33].

Against this background, one question that can be raised is: *Can we design techniques to effectively adjust the previously-trained classifier to the changed content on the Web, while minimizing the human effort required for labeling new data, and under what conditions, can such an adjustment be possible?* The research that we describe in this paper addresses specifically this question.

Contributions and Organization. We present two approaches to researcher homepage classification using *unlabeled* data readily available from academic websites. More precisely, we first adapt the well-known co-training approach [4] to reflect the change in the data distribution over time. Second, we design an iterative algorithm based on the Mini-batch Gradient Descent technique for learning a conforming pair of predictors using two different views of the data. We restrict ourselves to homepages of researchers in Computer Science since training datasets are available for this discipline. To the best of our knowledge, the problem of researcher homepage classification using unlabeled data available during focused crawling was not addressed in previous research. The contributions of this work are as follows:

- We show that with the classifiers trained on existing datasets for researcher homepage classification we incorrectly identify pages of types not seen in the training datasets as homepages. This results in unacceptable yield from the perspective of a focused crawler.
- We design novel features based on URL surface patterns and terms to complement content-based term and HTML features while classifying homepages and show that these two sets of features can be treated as two "views" for a researcher webpage instance.
- We show that the URL and content-based views can be used successfully in a co-training setup to adapt classifiers to the changing academic environments using unlabeled data. This finding enables us to accurately crawl the new academic website content without having to label a new dataset for re-training the classifier.
- Inspired by the success of co-training on this problem, we investigate loss functions that capture the disparity between the classifiers' predictions on the two views afforded by co-training. We design an iterative algorithm based on the Mini-batch Gradient Descent technique for minimizing this loss and learning a conforming pair of predictors with the two views.
- Finally, we show that minimizing our proposed loss function on unlabeled data closely corresponds to the effect demonstrated by co-training techniques. We posit that this loss can, therefore, be used as a measure, for tracking the progress of co-training schemes even in the absence of a validation dataset.

Although in this paper we focus on the design of accurate approaches for researcher homepage classification, our objective is to integrate this classification component in the context of focused crawling, to improve retrieval and indexing of scientific publications in digital libraries such as CiteSeer and ArnetMiner. In these usage environments, since maintaining up-to-date collections of research literature is of primary importance, having an accurate list of homepage URLs for frequent, periodic tracking is both feasible and scalable, compared to examining the entire content at academic websites each time.

The rest of this paper is organized as follows: We briefly summarize closely related work in Section 2. Researcher homepage classification is discussed in Section 3. We elaborate on details of our co-training experiments and learning conforming predictor pairs in Section 4. Experimental setup, datasets and results are discussed in Section 5, followed by a summary and future extensions to our work.

2. RELATED WORK

Researcher homepage classification is a well-studied webpage classification problem in context of digital libraries such as CiteSeer [22] and ArnetMiner [40]. Typically, content-based term features and HTML structure-based features are used for classifying webpages [36]. We propose the use of URL features as additional evidence for homepage identification. A smaller set (compared to ours) of URL-based features (presence of part of the name, presence of the character '∼', etc.), was used in isolating homepages among the search engine results for researcher name queries by Tang, et al [40]. URL-based features are widely used in tasks related to the Web. For example, URL strings were used for extracting rules to solve the webpage de-duplication problem [21]. Shih and Karger [38] used URL features, the visual placements of links in the referring pages to a URL for improving applications such as ad-blocking and recommendation, whereas a preliminary study by Kan and Thi [20] illustrates the use of URLs in performing fast webpage classification.

The problem of collecting a high-quality researcher homepage collection was studied for Japanese websites by Wang, et al. using on-page and anchor text features [42]. Tang, et al. studied homepage acquisition from search engine results using researcher names as queries [40]. In contrast, we seek to apply focused crawling using a seed list of academic websites (where researcher homepages are typically hosted) to acquire such a collection. Focused crawling first proposed by Bra, et al. is a rich area of research on the Web [5, 19]. Chakrabarti, et al. [7] present a discussion on the main components involved in building a focused crawler. Although focused crawling is our motivating application, this paper deals with the classifier component of the crawler and not with the crawler itself.

We show that the focused crawling scenario presents novel challenges in using a pre-trained homepage classifier in identifying relevant pages. Specifically, the classifier needs to be attuned to the changing types of pages on the Web. Co-training is proposed as a solution for addressing this challenge for homepage classification. Blum and Mitchell [4] first proposed co-training, an approach for semi-supervised learning when the number of labeled examples available for training is limited, and applied it to webpage classification. This approach requires having two views of features for the instances and has been shown to work well when the two views satisfy certain assumptions on "sufficiency" and "independence" [30]. Recent research addresses techniques for decomposing the feature set into two views when such a split is not naturally available [9, 15].

Multiview learning (of which co-training is a special case, with two views) is typically addressed by maximizing "consensus" or agreement among the different views [39, 24, 10]. Most solutions to multiview learning tend to frame the problem in terms of a global optimization problem and simulta-

neously learn classifiers for all the underlying views. In some cases, the solutions depend on underlying classification algorithm used [17, 6]. Although our proposed algorithm based on mini-batch gradient descent seeks to maximize consensus as well, our approach is a generic technique assuming only that the underlying classifiers output initial "parameter vectors" that are altered using a simple, iterative algorithm.

3. FEATURES FOR HOMEPAGE CLASSIFICATION

Webpage or text classification is typically handled using "bag of words" approaches. Specifically, the frequently occurring and discerning terms are collected from training data to form a feature dictionary that is used to represent instances as normalized term frequency or TFIDF vectors [26]. Homepage classification was previously studied as a text classification problem using term features [32, 31]. Previous work on the same problem also used other content-based features related to the HTML structure of the page such as the number of images/tables on the page, and the terms commonly found in anchor text of homepages [12]. In this study, we extracted content-based and URL-based features from our training sets. These features and the size of feature sets are summarized in Table 1. The term dictionaries contain terms that occur in at least three documents (i.e., webpages) and at least five times in the training set.

In addition to term dictionaries, we hypothesize that the URL strings of homepages can provide additional evidence for identifying homepages. Hence, we design novel URL-based features based on surface patterns and presence in WordNet[5]. The URL-based features are explained in the next subsection.

Type of features	#features
Content-based	
Top unigrams	18674
Number of tables/links/images on the page	3
Unigrams from anchor text on the page	30
URL-based	
Top unigrams and bigrams from URL strings, surface pattern and wordnet features	1039

Table 1: Feature types and the size of feature sets used in homepage identification.

3.1 URL strings as additional evidence

The idea of using URL strings in academic homepage identification comes from an error analysis of a crawl obtained with the content-based classifier. Consider some example URLs we encountered in our crawl listed in Table 2.

With some knowledge in academic browsing, one can confidently guess that the webpages at the URLs (1), (2), and (3) are unlikely to be researcher homepages. Similarly, among the URLs (4), and (5), while the former seems to be a homepage, the latter seems to lead to a course page. The above conjectures are based on the presumption that the URL strings are not "arbitrary", but, instead conventions are observed that are indicative of the target content at the URL. For instance, in the previous examples, words such as "projects", "events", alphanumeric patterns of the terms in the URL indicate that the URLs, (1), (2), (3) and (5) are most possibly not researcher homepages.

Treating "/" as delimiters, we extract features from the URL string following the domain name of a webpage. The list of all unigrams and bigrams from URL strings that occur more than thrice in the training dataset, comprise the URL-term dictionary. For terms in the URL not present in this dictionary, we look for their presence in WordNet to check if they are common words or proper nouns. WordNet is a large, lexical database of nouns, verbs, adjectives and adverbs for English, organized as a concept graph [29, 16].

In addition, we capture the surface patterns of the URLs including the presence of hyphenated or underscored words, alphanumeric patterns, long words (i.e., words having greater than 30 characters), question marks and the presence of characters such as tilde. These features are designed to filter out the URLs that commonly represent course pages, announcements, calendars and other auto-generated content. For instance, a typical homepage URL string in Computer Science departments has the name of the researcher following the $\sim$ character after the domain name (e.g., `http://people.cs.umass.edu/~mccallum/`). This pattern is usually captured by our "TILDENONDICT" feature, where `mccallum` is a non-dictionary term. Partial sets of extracted features are shown along with the URLs listed in Table 2.

The above sets of features perform very well on the training datasets as shown Section 5. We, therefore, do not study other complicated, problem-specific feature design or feature selection. Instead our focus in this work is to study how these classifiers perform "in the wild". We also note here that, a classifier that can make accurate predictions using URL features can be quite beneficial from the perspective of *efficiency* for a focused crawler. A crawler can potentially bypass examining the content of a page if a confident decision can be made based on the URL string. However, we may not be able to always extract features from the URL strings. For instance, consider the following URLs from our crawls:

`http://john.blitzer.com/`
`http://clgiles.ist.psu.edu/`
`http://ben.adida.net/`

In these cases, it is not clear from the URL string that the target content refers to academic homepages. Even if complicated name-extraction based features were designed for the above cases, it is rare to find academic homepages with '.com' and '.net' domain suffixes. Based on the URL alone, we cannot be confident if the target content is an academic homepage or a company/personal homepage. For the second case, 'clgiles' could refer to a machine name. In addition to the above cases, given that feature dictionaries typically comprise features that meet a frequency requirement, we may not be able to extract features for all URLs. In our training datasets (Section 5), we were unable to extract URL features for about 27% of the instances. Therefore, content-based and URL features complement each other while identifying homepage instances and a focused crawler might be required to use either or both of these sets of features.

4. HOMEPAGE CLASSIFICATION USING UNLABELED DATA

We show in our experiments (Section 5) that, although content-based features perform extremely well on the training datasets, they are not very successful on the validation and test sets that were collected from the current-day academic websites. On the other hand, URL features show good performance on both training and validation datasets.

[5] http://wordnet.princeton.edu/

-	URL
1	www.cs.columbia.edu/robotics/projects/visual_control/allen-realtime.html SEQBEGIN_robotics, robotics, projects, hyphenatedword, hyphenatedword
2	www.cs.ucla.edu/events/events-archive/2011/limits-of-communication events, hyphenatedword, NUMBER, hyphenatedword
3	http://www.cc.gatech.edu/hg/image/63622?f=ccfeature QMARK, hg, image, NONDICTWORD, NONDICTWORD_SEQEND
4	http://www.cs.umd.edu/∼djacobs/index.html TILDENONDICT, index
5	www.cs.umd.edu/∼djacobs/CMSC828/CMSC828.htm TILDENONDICT, ALPHANUM, ALPHANUM

Table 2: Example URLs with partial sets of extracted features (shown on the next line after each URL)

However, as pointed out in the previous section, we may not be able to extract URL features for all instances and it is, therefore, imperative to have an accurate content-based classifier as well.

We now address the questions: *Can we adapt the content-based classifier to perform well in the deployment environment with the help of the URL-based classifier? Can the two classifiers "teach" each other so as to perform better in the new environment, using the co-training approach?* Since the URL and content features provide evidence for classifying a webpage instance independently, intuitively, it appears possible that there are instances that the URL classifier makes mistakes on, which the content-based classifier identifies correctly and vice versa.

Blum and Mitchell proposed co-training in context of webpage classification [4]. In their datasets, webpages are representable in terms of two distinct views: using terms on webpages and terms in the anchor text of hyperlinks pointing to these pages. When few labeled examples are available for training, they showed that co-training could be used to obtain predictions on the unlabeled data to enlarge the training set. Blum and Mitchell's experiments and the subsequent experiments by Nigam and Ghani [30] showed that when a natural split of features is available, co-training that explicitly leverages this split has the potential to outperform classifiers that do not.

We study the applicability and extension of co-training for our problem. Although the essential motivation is to make use of the naturally available feature split and enable classifiers to learn from each other, we highlight the following aspects of our setup: Previous studies and benefits from co-training were illustrated on datasets where the unlabeled data is arguably from a *similar distribution*. That is, the positive and negative instances in the labeled datasets are representative of those in the unlabeled data. This is in contrast to our case, where our positive class is fairly well-defined (homepages), whereas the negative class is described in terms of "not positive". More precisely, although our training dataset has examples for the negative class, webpages encountered during the crawls can belong to types not encountered in the labeled data. We present an error analysis in Section 5, that illustrates the "new" types of webpages encountered in our crawl, potentially causing the pre-trained content-based classifiers to underperform during crawling.

The number of negative instances encountered during our crawls is higher in comparison with the number of positive instances. While this aspect was noticed during our experiments, a previous estimation experiment using mark-recapture methods had indicated that academic homepages comprise a minute fraction of the Web [12]. We can expect this imbalance to become more prominent as more examples are sampled over the co-training rounds. In the algorithm studied by Blum and Mitchell, the ratio between the number of positive and negative instances added from the unlabeled data is maintained to be the same as that in the training dataset during each iteration of co-training [4]. We argue that avoiding this constraint is better in our scenario since we want the datasets to be more representative of the changing distribution.

Most classification algorithms are sensitive to the number of positive and negative instances available in the training data and are known to learn biased classifiers in case of severe imbalance [3, 23]. We employ the idea of altering the mis-classification costs for the underlying classifiers during each round of co-training to handle this problem. For example, if the training dataset has 10 positive and 100 negative instances, we can set the penalty incurred on making mistakes on a negative instance to be $\frac{1}{10}$th of the penalty incurred on making mistakes on a positive instance. For most implementations of classification algorithms, the misclassification costs can be specified as a parameter during the training process [18].

Our co-training setup is detailed in Algorithm 1. L and U represent the labeled and unlabeled datasets, respectively, available at each iteration. They comprise instances with both the views (content-based and URL-based feature sets). For a round of co-training, we train classifiers, C_1 and C_2, on the two available views, using misclassification costs, ρ_1 and ρ_2, respectively. Next, "s" number of examples are sampled without replacement into S from the unlabeled data and C_1 and C_2 are used to obtain predictions for these instances. The $GetConfidentEgs$ method is a generic placeholder that stands for a function that determines what instances from S are chosen for addition in subsequent rounds of co-training. We use the notation L_1^+ to represent the positive instances in the set L_1 whereas L_1^1 indicates that the view 1 (or feature set 1) of the examples in L_1 is being used.

Based on previous studies in co-training [4, 30], we studied the following strategies for this function:

- **AddBoth**: In this scheme, we add all examples from S that are labeled by C_1 or C_2 confidently to the training set for the next round. This approach is similar to self-training used in semi-supervised learning where confidently predicted unlabeled instances are added to the training set for retraining the classifier in subsequent rounds [45]. However, in contrast with self-training that uses a single view, in **AddBoth**, confident predictions are obtained from two sources (view 1 and 2) for addition into subsequent rounds.

- **AddCross**: In this scheme, examples from S, confidently labeled by C_1 are added to view 2 for the next

round and vice versa. That is, we use the examples confidently labeled by one classifier while training the other classifier in the next round. Cross-addition also seems resilient to handling the possibility of cascaded errors over the iterations. If a classifier makes a confident but incorrect prediction, we would like to avoid feeding this example in the next round to the same classifier, a common problem in self-training [45].

- **AddCrossRC**: This scheme is similar to **AddCross** with the additional constraint on the number of positive and negative instances added in each round. This constraint was originally studied by Blum and Mitchell and ensures that the ratio of the number of positive and negative instances added in each round is the same as that in the initial labeled dataset [4].

Algorithm 1 Procedure for Co-training

***I*nput**: L, U, 's'
$L_1 \leftarrow L$, $L_2 \leftarrow L$
$\rho_1 \leftarrow \phi$, $\rho_2 \leftarrow \phi$
while $U \neq \phi$ **do**
  Compute ρ_1 using $\frac{|L_1^+|}{|L_1^-|}$, ρ_2 using $\frac{|L_2^+|}{|L_2^-|}$.
  Train classifier C_1 using (L_1^1, ρ_1).
  Train classifier C_2 using (L_2^2, ρ_2).
  $S \leftarrow \phi$
  Sample 's' examples from U and move them to S.
  $U \leftarrow U \setminus S$
  $S_1, S_2 \leftarrow GetConfidentEgs(S, C_1, C_2)$
  $L_1 \leftarrow L_1 \cup S_1$, $L_2 \leftarrow L_2 \cup S_2$
end while
***O*utput**: Classifiers C_1, C_2.

The co-training algorithm is general and can be applied with any choice of classifiers on the two views. Blum and Mitchell provided a PAC-style analysis of co-training with probabilistic classifiers and showed that co-training works when the assumptions on sufficiency and independence are met. That is, each view should be sufficient to predict the class label, and the two views are independent given the class label. Recent studies have proposed relaxed criteria under which co-training techniques still work [1]. However, in practice, it is tricky to judge if co-training works for a problem and to verify if the assumptions are satisfied [15]. These questions are more relevant in context of recent research in obtaining two views from a single view when two views are not naturally available for applying co-training [9]. With this context, we now discuss our formulation of the effect obtained with co-training, in terms of a loss function. This formulation allows us to track whether the co-training process is beneficial for a given problem, even without the use of a validation dataset.

4.1 Learning Conforming Predictors on Unlabeled Data

We assume that classifiers, C_1 and C_2 trained on the two views are parameterized in terms of their weight vectors, $\mathbf{w_1}$ and $\mathbf{w_2}$. Most classification algorithms e.g., Support Vector Machines (SVM) and Maximum Entropy (MaxEnt), output weight vectors capturing the importance of each feature as part of the training process [3].

One can expect co-training to benefit a classification problem if one classifier (say, C_1) can "guide" the other (C_2) on examples that the latter makes mistakes on. This guidance is provided by adding examples confidently labeled by C_1 to the subsequent round of training C_2. This observation hints at the possibility of directly manipulating C_2, based on C_1's prediction for an example that C_2 is not confident about. This effect can be achieved by optimizing a function that directly captures the mismatch in the predictions of the two classifiers.

Elaborating further, given that the concept classes, "positive" and "negative" are still the same on unlabeled data, if C_1 and C_2 are accurate, they would make similar predictions on the unlabeled data. This intuition is the basis for "consensus maximization" widely adopted in multiview learning, of which co-training is a special case with two views [39, 24, 10]. The mismatch in predictions by C_1 and C_2 on unlabeled data can be quantified using a loss function. The squared error loss function commonly used in machine learning captures this loss as:

$$\mathcal{L}_U(\mathbf{w_1}, \mathbf{w_2}) = \frac{1}{|U|} \sum_{u \in U} \left(f_1(\mathbf{w_1}, u) - f_2(\mathbf{w_2}, u)\right)^2$$

The above formulation captures the average squared-difference in predictions from the two views on unlabeled data. $\mathbf{w_1}$ and $\mathbf{w_2}$ correspond to the parameter vectors corresponding to C_1 and C_2, respectively, and u refers to an example from U, having two views, u_1 and u_2. For a given example, $u = (u_1, u_2)$, the functions, f_1 and f_2 act on u_1 and u_2 respectively, and make the predictions from C_1 and C_2 comparable. These functions could be generic (e.g. a function that outputs the probability that the instance is positive) or classifier-dependent (for e.g. a function that outputs scaled distances from the separating hyperplane in case of Support Vector Machines). Minimizing $\mathcal{L}$ corresponds to adjusting the weight vectors, $\mathbf{w_1}$ and $\mathbf{w_2}$, so that they make similar predictions on U.

In contrast with multiview learning methods, where learning the classifiers is folded into a global objective function in sophisticated ways [39, 24, 10], we adopt a simpler approach that works off the initial parameter vectors and iteratively modifies them in a "co-training like" manner. Note that this initialization plays a crucial role in avoiding trivial solutions (such as $\mathbf{w_1}, \mathbf{w_2} = \mathbf{0}$) that are potentially possible since the loss is optimized only on unlabeled instances. Our proposed technique for obtaining the "pair of conforming classifiers" is described in Algorithm 2.

In Algorithm 2, we start with the original parameter vectors $\mathbf{w_1}$ and $\mathbf{w_2}$ from classifiers C_1 and C_2, respectively, and iteratively adjust these vectors so that the values of $f_1(\mathbf{w_1}, u_1)$ and $f_2(\mathbf{w_2}, u_2)$ look similar for all $u \in U$. The input parameter, $\#oIters$, refers to the number of times the inner loop comprising of the two gradient descent steps is executed, where as, the $\#iIters$, and α are parameters for the gradient descent algorithm. Overall, the values of $\#oIters$, $\#iIters$, and α control the rate of convergence of the algorithm and can be set experimentally. These parameters can be set based on the base classifiers used, noting when the decrease in the objective function value is below a threshold. Adaptive tuning of these parameters by tracking the change in the value of the objective function in every iteration is a subject for future study [34].

In each iteration, we employ mini-batch gradient descent to minimize the loss function, once w.r.t. $\mathbf{w_1}$ and next w.r.t. $\mathbf{w_2}$. The mini-batch gradient descent algorithm is a hybrid approach often used for large-scale machine learning problems. This approach combines the best of stochastic (on-

Algorithm 2 Learning a pair of Conforming Classifiers

Input: $\mathbf{w_1}$, $\mathbf{w_2}$, U, 's', #oIters, #iIters, α
$o = 0$
while $o \leq \#oIters$ **do**
  %Perform Mini-batch Gradient Descent to obtain a new $\mathbf{w_2}$
  $i = 0$
  while $i \leq \#iIters$ **do**
    $S = \phi$. Sample 's' examples from U into U_t
    for $u \in U_t$ **do**
      if $f_1(u_1, \mathbf{w_1})$ is confident **then**
        Add u to S
      end if
    end for
    $\mathbf{w_2} \leftarrow \mathbf{w_2} - \alpha \frac{\partial \mathcal{L}_S}{\partial \mathbf{w_2}}$.
    $i \leftarrow i + 1$
  end while
  %Perform Mini-batch Gradient Descent to obtain a new $\mathbf{w_1}$
  $i = 0$
  while $i \leq \#iIters$ **do**
    $S = \phi$. Sample 's' examples from U into U_t.
    for $u \in U_t$ **do**
      if $f_2(u_2, \mathbf{w_2})$ is confident **then**
        Add u to S
      end if
    end for
    $\mathbf{w_1} \leftarrow \mathbf{w_1} - \alpha \frac{\partial \mathcal{L}_S}{\partial \mathbf{w_1}}$.
    $i \leftarrow i + 1$
  end while
  $o \leftarrow o + 1$
end while
Output: $\mathbf{w_1}$, $\mathbf{w_2}$

line) gradient descent and batch gradient descent to obtain fast convergence during optimization by running gradient descent on small batches of randomly selected examples [13].

In our algorithm, in each iteration, a small batch of instances are randomly sampled from the unlabeled data, U and the loss function defined using instances for which $\mathbf{w_1}$ makes confident predictions from this sampled set. This loss is minimized using gradient descent to adjust $\mathbf{w_2}$. A similar process is then applied for adjusting $\mathbf{w_1}$ using confident predictions from $\mathbf{w_2}$. In effect, as the algorithm proceeds, we are adjusting the parameters of each classifier so that it makes predictions that are aligned with those of the other classifier's confident predictions. Upon convergence, both $\mathbf{w_1}$ and $\mathbf{w_2}$ are adjusted so that they make conforming predictions on the unlabeled data.

In our experiments, we used the differentiable, logistic sigmoid function for f_1 and f_2. Typically, classifiers use the parameter vector, $\mathbf{w}$, for computing decision values for each instance. That is, given an instance $\mathbf{x}$, the dot product value, $\langle \mathbf{w}, \mathbf{x} \rangle$, is used for determining the label assignment for the instance. This value can be 'squashed' to a number between 0 and 1 indicating that the probability that instance has a particular label with the logistic function [3]:

$$P(t) = \frac{1}{1 + e^{-t}} \quad \text{with} \quad \frac{dP(t)}{dt} = P(t) \cdot (1 - P(t))$$

Given, the simple form for the derivative, we can directly use the values of f_1 and f_2 (that we compute anyway), for computing the gradients in Algorithm 2. Although the effect obtained by Algorithm 2 is similar to that of co-training, there are certain benefits to using this algorithm instead of co-training. In practice, the co-training process is terminated either when no more examples are available or when the performance converges on the validation dataset. A loss function capturing the non-conformity among the predictions from the two classifiers *directly* provides a way to measure the effect of co-training as it is being applied.

We provide a preliminary, experimental demonstration of the connection between co-training and our proposed algorithm in Section 5. A more detailed analysis, study of other choices for the loss function $\mathcal{L}$ and the functions, f_1 and f_2, are a subject of future work. Nevertheless, quantifying the discrepancy in predictions from the two views and an algorithm to directly address this aspect is an exciting step in understanding when co-training works. We show in Section 5 that our method can be used in lieu of a validation dataset for tracking the performance of co-training.

5. EXPERIMENTS

We discuss 3 types of experiments: First, we study the performance of content-based and URL-based features on the training and validation datasets. Second, we show that co-training can successfully address the problem of mismatch in the training and deployment environments for homepage classification. Finally, we show that our proposed algorithm (Algorithm 2), achieves the same effect as co-training.

5.1 Datasets

We describe the datasets available for studying academic homepage classification for the Computer Science discipline. The WebKB dataset was previously used by several researchers for studying webpage and text classification including semi-supervised learning and co-training [4, 27, 31, 33]. The WebKB collection contains academic webpages from Computer Science departments of four universities: Cornell, Washington, Texas and Wisconsin, categorized into 7 categories (student, faculty, staff, department, course, project, and other). The "other" class comprises pages that cannot be fit into the remaining six classes; for instance, a publications page that links to a page belonging to a faculty page. This collection was obtained in 1997 and is not quite representative in terms of the types of webpages available on the academic websites of the current day as shown in the error analysis we present shortly. Another set of author-provided homepages are available from the bibliographic resource for Computer Science and related areas, DBLP[6]. Although this collection of 6000 homepages is more recent, we do not have negative instances as part of this dataset.

For the deployment scenario, we crawled the university websites listed in Table 3. These websites were selected arbitrarily from the list of top US graduate schools in Computer Science (obtained from rankings in US News[7]). We seeded our crawl with these URLs and used the open-source crawling software, Heritrix[8] (version 1.14.3), for obtaining all web pages of content-type 'text/html', within a depth of 5 starting at the parent URL. In total, we were able to obtain 162,369 webpages using this process. This crawl was performed in April, 2012 and hence, represents a relatively recent snapshot of content at these URLs. Note that, this setup was used for the purpose of experiments. Our final goal is to embed accurate classifiers into the crawler, so as to avoid obtaining webpages that are not homepages.

To validate the performance of our classifiers, we randomly selected sets of 100 webpages from each of the 16

[6] http://www.informatik.uni-trier.de/~ley/db/
[7] http://www.usnews.com/
[8] https://webarchive.jira.com/browse/HER

(1) http://www.cs.wisc.edu, (2) http://www.cs.umich.edu, (3) http://www.cs.umd.edu, (4) http://www.cs.ucla.edu (5) http://www.cis.upenn.edu, (6) http://www.cs.columbia.edu, (7) http://www.cs.princeton.edu, (8) http://www.eecs.berkeley.edu (9) http://www.cs.washington.edu, (10) http://www.cs.brown.edu, (11) http://www.cs.utexas.edu, (12) http://www.cs.cornell.edu, (13) http://www.eecs.mit.edu, (14) http://cs.illinois.edu, (15) http://www.cc.gatech.edu, (16) http://www.cse.ucsd.edu

Table 3: List of Seed URLs

Training(WebKB+DBLP)	Unlabeled(Crawl)	Test(Crawl)	Validation(Crawl)
9263/4719	143145	1600/89	500/42

Table 4: Datasets Description: Entry a/b represents a instances out of which b are labeled positive

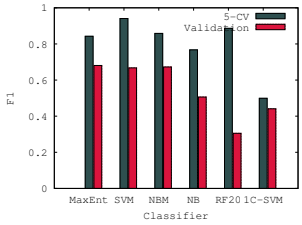

Figure 1: Content Features

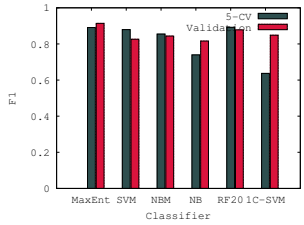

Figure 2: URL Features

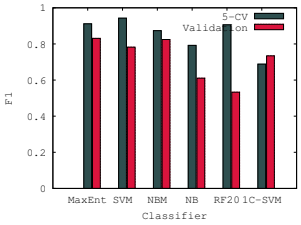

Figure 3: Content+URL Features

universities listed in Table 3 and manually labeled them. From the remaining pages, another set of 500 pages were randomly chosen for validating or tuning the methods proposed in Section 4. For our experiments, we only consider instances for which both the views are available, that is, pages from which we are able to extract both URL and content features. A summary of the datasets [9] just described is shown in Table 4.

5.2 Classification Experiments

We study the performance of our content-based and URL features using the classification algorithms: Naive Bayes (NB), Naive Bayes Multinomial (NBM), Random Forests (RF), Support Vector Machines using a linear kernel (SVM) and Maximum Entropy (MaxEnt). Naive Bayes and Naive Bayes Multinomial are generative models whereas Random Forests is an ensemble method using Decision Trees. Discriminative algorithms such as Support Vector Machines [41, 11] and Maximum Entropy classifiers [31] are being used extensively for text classification problems in the recent times. These methods output parameter vectors as part of the training process that can be manipulated directly in our Algorithm 2. Further details on different classification algorithms and their parameters can be found in a standard machine learning textbook (e.g. [3]). The Naive Bayes algorithm is well-studied in context of co-training and its theoretical analysis, although co-training is a classifier-independent technique. For text classification, the Naive Bayes Multinomial classifier is different from Naive Bayes in terms of its modeling of term counts with multinomials [27].

Figure 1 shows the weighted $F1$ measure on our training (five-fold cross-validation) and validation datasets using the classification algorithms above with content-based features. Figure 2 shows similar results with URL-based features. We used classifier implementations provided by Weka [18], libSVM [8] and Mallet [28]. Where applicable, we tuned the parameters on the training datasets for the best performance (e.g., the C parameter for SVM, the number of trees in RF). Similarly, to handle potential imbalance in instances belonging to different classes during co-training, we use appropriate misclassification costs (e.g., using the CostMatrix option in Weka and the "w" setting in libSVM).

As the figures illustrate, discriminative algorithms such as SVM and MaxEnt generally outperform the generative models such as NB and NBM on the homepage identification task using both content (Figure 1) and URL (Figure 2) features. The performance of URL-based classifiers (Figure 2) is typically higher compared to that of content-based classifiers (Figure 1), especially on the validation set that was collected from the crawled data. We performed an error analysis on the validation set and noticed that the content-based classifiers suffer from a high false-positive rate, and incorrectly label negative instances as positive. The WebKB dataset includes negative instances coming from types such as course, department, and project-related pages. However, about 212 out of the 500 validation instances could not be categorized into any of the 7 types present in WebKB. Instead, we can capture these pages under the following new types:

1. Webpages related to colloquium, seminars, lectures, publications, papers, talks, slides.
2. Webpages that describe code, widgets, scripts, datasets.
3. Webpages related to department activities such as picnics, pages with embedded photos, and personal pages.
4. Webpages pertaining to information on news, events, highlights, faq, forms.
5. Webpages pertaining to alumni-related information, job and contest calls.

Given that our validation set only comprises 500 instances, it is reasonable to suspect that other types of webpages exist in our crawled collection. We used Information Gain [3] on the training and crawl-based datasets to understand the feature-class correlation between the two datasets. The top-10 features ranked by this measure, shown in Table 5, also point to the difference between the two environments. However, our aim is not to model new types of webpages, but rather, we wish to learn a discriminator that isolates academic homepages from non-homepages. In our experiments, we noticed that webpages belonging to the new types 1 and 2 above were often misclassified as academic homepages. However, surface patterns and cue words such as "seminars" in the URLs are effective for classifying these instances correctly.

[9] all datasets, dictionaries and feature files are available upon request

URL		Content	
training	crawl	training	crawl
TILDENODICT	ALPHANUMBER	gmt	university
TILDENODICT_SEQEND	TILDENODICT	server	computer
ALPHANUMBER	ALPHANUMBER_ALPHANUMBER	type	science
NONDICTWORD	HYPHENATEDWORD	html	department
courses	ALPHANUMBER_SEQEND	content	numImages
ALPHANUMBER_SEQEND	TILDENODICT_SEQEND	text	numLinks
users_NONDICTWORD	QMARK	date	cs
users	NUMBER	professor	box
NONDICTWORD_SEQEND	courses	university	ri
homes	NUMBER_SEQEND	research	providence

Table 5: Features ranked based on Information Gain on training and crawl datasets

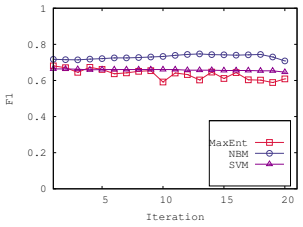

Figure 4: Self-training: Content

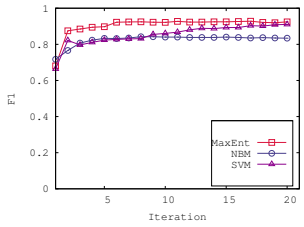

Figure 5: Co-training: Content

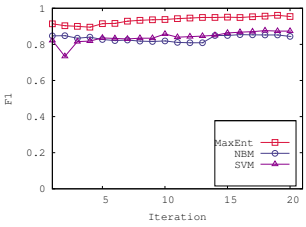

Figure 6: Co-training: URL

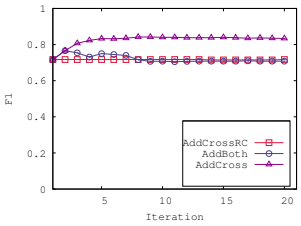

Figure 7: Co-training Schemes (NBM)

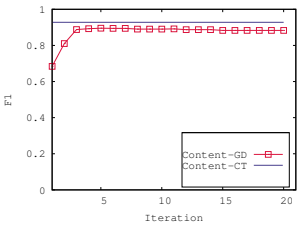

Figure 8: GD: Content (MaxEnt)

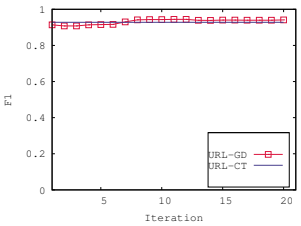

Figure 9: GD: URL (MaxEnt)

Further, we studied the following approaches for improving performance of content-based classifiers for the new environment: First, we learned classifiers on the combined set of features for each instance (content+URL). The performance of these classifiers is shown in Figure 3. As the plots indicate, using the combined set of features on the validation set is better than using content features alone (Figure 1), but still worse than using URL features alone (Figure 2).

Second, we trained a one-class SVM (linear kernel) for identifying homepages. One-class classifiers learn discriminators for a class by using just the positive instances without explicitly modeling the other classes [37, 44]. One-class methods are typically used for detecting outliers and novelty detection. However, one-class SVMs do not work as well as binary classifiers for our problem on the training datasets. Their performance on the validation sets is better, but not comparable to the best performing classifiers on the validation datasets (see Figures 1, 2, and 3, 1C-SVM).

Finally, we employed self-training to train the content-based classifiers using the unlabeled data. Self-training is an iterative approach commonly used in semi-supervised learning. Unlabeled examples, predicted confidently by the classifier are added back to enlarge the training dataset used in retraining the classifier in the subsequent iteration [43]. The intuition behind self-training has been compared to pseudo-relevance feedback employed in information retrieval [30, 26]. Although, this approach was successfully applied to some problems before [43], for some applications, is known to suffer from cascaded errors resulting in the final classifier being less effective than the initial one [45]. Our content-based classifiers do not show any improvements with self-training and, depending on the classification algorithm, even show decreased accuracy *on the validation set*. This behavior is illustrated in Figure 4.

We chose the better performing of all classification algorithms: NBM, SVMs and MaxEnt as base classifiers for the remaining experiments.

5.3 Co-training Experiments

We studied co-training (Algorithm 1) using the different schemes, **AddBoth**, **AddCross**, and **AddCrossRC** for selecting unlabeled instances for the next round. We use the same type of algorithm for training classifiers on both views (e.g., SVM for URL features as well as for content-based features). We sampled 5000 instances from the unlabeled dataset in each iteration and consider an instance for addition only if a prediction was made for this instance with a confidence probability ≥ 0.9. Note that the labeled training set is WebKB+DBLP, whereas the unlabeled training set is the data from our crawl. Figure 5 shows the performance of content-based classifiers (within co-training) *on the validation set*. Figure 6 shows similar results using URL-based classifiers. As shown in the figures, co-training successfully manages to pull up the performance of the content-based classifiers using the unlabeled data. The URL classi-

fiers being more stable add relevant instances to the labeled datasets over successive rounds, enabling the content-based classifier to retrain itself over the iterations and learn to discriminate better among the current-day pages on the web. Although the initial rounds do not result in improvements in the performance for the URL classifier, once the content-based classifiers are up in accuracy, they are able to provide useful examples to the URL classifiers, in turn, resulting in improvements even for the URL classifier (Figure 6).

Figure 7 shows the effect of different schemes for instance selection during co-training iterations for the NBM algorithm. Difference in performance on the validation dataset was noticed for the co-training schemes with NBM as base classifiers. For SVM and MaxEnt pairs, there was no observable difference in the performance attained by the different schemes, AddBoth and AddCross. The **AddCross** scheme does either better or on par with the other schemes. In addition, since we use a similar scheme for adding instances in Algorithm 2, we chose **AddCross** for comparisons with the Gradient Descent based approach in the next section.

Due to space constraints, we do not show the comparison between co-training without the mis-classification costs accounted for. Not surprisingly, not accounting for class imbalance results in mistakes on the positive instances due to the large number of negative instances in the training data and the performance on this class reduces drastically over the co-training iterations for both the **AddBoth** and **AddCross** schemes. The performance on **AddCrossRC** is not affected due to the ratio constraint being maintained over the iterations since we started with an almost balanced training dataset (see Table 4). However, the performance with the **AddCrossRC** is not as good as the **AddCross** scheme with class imbalance accounted for.

5.4 Gradient Descent Experiments

Figures 8 and 9 show a run of Algorithm 2 with initial weight vectors obtained with MaxEnt classifiers. These initial vectors are obtained by running the MaxEnt trainer over the labeled (training) dataset. The figures show the classification performance on the validation set after the termination of each iteration (from 1 to 20) of Algorithm 2, which minimizes the loss function in each round. Similar plots with the initial weight vectors obtained from SVMs are shown in Figures 10 and 11.

The figures also show the comparison of the proposed gradient descent algorithm with co-training. We plot the maximum $F1$ score that was obtained with co-training experiments (previous sub-section) alongside the curves in Figures 8-11, to illustrate that our proposed algorithm in effect attains similar performance improvements that are possible with co-training. Although these plots show the $F1$ variation over the validation set, note that, the validation set is not used for tracking the optimization process. Instead, the algorithm can terminate either after a pre-set number of iterations or by explicitly tracking the objective value for convergence. Each iteration of the algorithm involves running mini-batch gradient descent twice, once for each classifier.

We observed the objective values to be converging in about 20 iterations when the algorithm is initialized with the weight vectors from MaxEnt whereas it takes about 50 iterations with those from SVM. These values correspond to the $\#oIters$ in Algorithm 2. The $\#iIters$ and α values for Mini-batch Gradient Descent were set to 50 and 0.1, respectively, in all experiments. About 1% of unlabeled data was randomly sampled in each round and examples that were predicted with 90% confidence were used for computing the loss function that is minimized with Gradient Descent. The #iIters and α values affect the rate of convergence for Gradient Descent. We chose these values based on experimentation, instead of adaptively using techniques such as line search [34]. Experimenting with these parameters is left for future work. With the settings just described, the run times for convergence were similar to that of co-training. In general, depending on the classification algorithms used, the co-training experiments took times ranging between 5min-4hrs on a 16-core, 800MHz, 32GB RAM, AMD Opteron, Linux server.

We plot the $F1$ on the validation dataset against the computed objective function value in Figure 12. The plot depicts the close correspondence between reducing the discrepancy between predictions based on the two views and the improving performance on the validation dataset. We also illustrate the connection between the effect of co-training and our proposed loss function by plotting the value of our loss function on unlabeled data available in that iteration as co-training progresses in Figure 13. This plot highlights the fact that when co-training works, it seems to be due to the reducing discrepancy between the predictions from the two views used in co-training. The list of top-10 features that undergo the most change in the MaxEnt weight vectors after Algorithm 2 converges are shown in Figure 15.

5.5 Evaluation on the Test Set

Method	Precision	Recall	F1
NBM-Before	0.8910	0.4700	0.5890
NBM-After-CT	0.9350	0.7980	**0.8470**
SVM-Before	0.9413	0.4947	0.6048
SVM-After-CT	0.9147	0.8167	0.8559
SVM-After-GD	0.9175	0.8826	**0.8977**
MaxEnt-Before	0.8806	0.4167	0.5380
MaxEnt-After-CT	0.9234	0.9295	**0.9262**
MaxEnt-After-GD	0.9272	0.9401	0.9158
GET-1	0.8668	0.5598	0.6719
GET-2	0.8761	0.5614	0.6724
MaxEnt-Combined-Before	0.9469	0.7219	0.7938
MaxEnt-After-ST	0.9418	0.7098	0.7849
GET-1	0.9273	0.4765	0.5895
GET-2	0.9365	0.7591	**0.8201**

Table 6: Before/After performance on the test set with Content Features

So far, we have been showing performance results using the validation set (Table 4) for the sake of illustration. We summarize our evaluation with our final classifiers on the test dataset in Figure 14 and Table 6. The table shows the weighted precision, recall and $F1$ measures [26] whereas the figure illustrates the improvements in $F1$ via bar charts. We compare the original content-based classifiers trained on the training datasets (the 'Before' entries) with our proposed methods. The 'After-CT' entries use the classifiers obtained after co-training was employed with unlabeled data and terminated after convergence was attained on the validation datasets (typically in about 20 iterations). The 'After-GD' entries use the weight vectors obtained after running Algorithm 2 starting with the original weight vectors.

We also compared MaxEnt with "expectation regularization", a technique proposed by Mann, et al [25]. In this semi-supervised learning method, additional regularization terms are added to the objective function while learning classifiers of the exponential family. These additional terms encourage

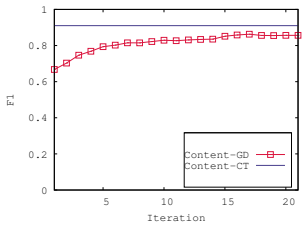

Figure 10: GD: Content (SVM)

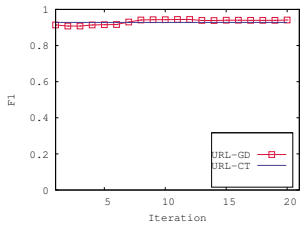

Figure 11: GD: URL (SVM)

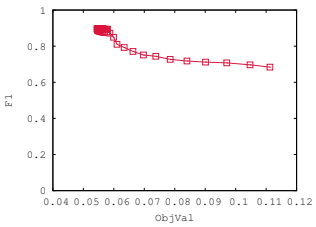

Figure 12: ObjVal vs. F1

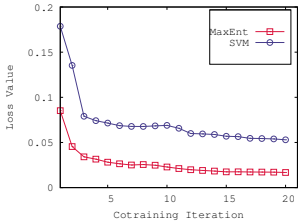

Figure 13: Reducing value of the squared-error loss as co-training progresses

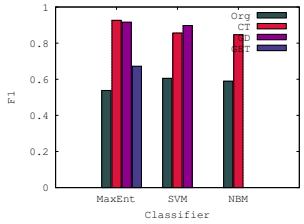

Figure 14: Performance of our methods on the test datasets

URL	Content
NONDICTWORD	numLinks
TILDENODICT	numImages
NONDICTWORD_SEQEND	research
LONGWORD	here
LONGWORD_SEQEND	courses
QMARK	slide
TILDENODICT_SEQEND	academics
HYPHENATEDWORD	education
grads	publications
ALPHANUMBER	document

Figure 15: Features that are most affected after running Algorithm 2 on MaxEnt weight vectors

the predictions on unlabeled data to meet certain expectations that capture label priors or feature-correlations. The expectations are expressed as soft constraints while learning the classifiers [14]. For example, for our problem, we could specify constraints, which indicate that the feature 'homepage' in the HTML title strongly suggests that the class-label is positive.

We obtained the code for automatically generating constraints from the labeled data using information gain and an implementation of the expectation regularization framework with MaxEnt classifier (GETrainer) from the authors[10]. Using the parameter settings suggested by the authors, the performance of the GETrainer on the test dataset is listed in the "GET-1" and "GET-2" rows in Table 14. Labeled data can be used to extract constraints but class labels are not required for learning the GETrainer. The "GET-1" row refers to the performance using unlabeled data from the crawl environment while training, where as the "GET-2" row shows the performance when the labeled dataset was used during training. Based on the performance of the GETrainer on the validation dataset with different number of constraints, we chose the number of constraints to be 8000 for the "GET-1" setting and 6000 for "GET-2".

Figure 14 and Table 6 indicate that the "expectation regularization" technique is succesful in capturing feature constraints and improves the performance of our content-based classifier. We also evaluated the GETrainer using feature vectors containing both content-based and URL features. For the combined set of features, the GE method achieves an improved performance although it is not very high compared to that obtained with the original classifier ('Before' entry). However, larger benefits are obtained by harnessing the split of features in terms of the two views.

6. SUMMARY AND FUTURE WORK

We studied the problem of adapting a classifier trained on a labeled dataset of webpages to a related environment containing newer types of webpages in the context of focused crawling for researcher homepages. We showed that co-training techniques, which use two different views of the data, can *effectively* incorporate unlabeled data to improve the classification performance in the deployment (crawling) scenario. Although our evaluation is specifically for homepage classification, we posit that our findings hold for other problems or domains. It is reasonable to expect a mismatch in training/test environments in any focused crawling situation, given the changing rate of content on the Web. Intuitively, we can expect our techniques to work well when the following criterion is met: a view v_1 must be able to predict at least one unlabeled example confidently that view v_2 cannot and vice versa. When this condition is satisfied, the views can "help each other" over the iterations.

We also proposed a novel formulation of co-training in terms of a loss function. This loss can be directly minimized via a mini-batch gradient descent algorithm. Our results indicated that even without a validation set, one can track the effect of the co-training process via our loss function.

In future, it would be interesting to explore other aspects of our algorithm for learning "conforming pairs of classifiers" as well as other forms of the loss function and function choices for comparing classifier predictions. It would also be interesting to explore classifiers' performance when small fractions of the newer types of webpages, manually labeled, will be added to the training sets. For focused crawling, our motivating scenario, we will study the benefits of folding in our proposed URL and content-based classifiers into the crawl process both in terms of yield and efficiency.

7. ACKNOWLEDGMENTS

We thank the members of IBM India Research Labs and Daniel Kifer (Penn State) for helpful discussions related to optimization algorithms. This material is based upon work supported partially by the United States National Science Foundation under Grant No. 0845487.

[10]We thank G. S. Mann, G. Druck and A. McCallum for sharing their implementation with us.

8. REFERENCES

[1] M. F. Balcan, A. Blum, and K. Yang. Co-Training and Expansion: Towards Bridging Theory and Practice. In *NIPS*. 2005.

[2] K. Balog, T. Bogers, L. Azzopardi, M. de Rijke, and A. van den Bosch. Broad expertise retrieval in sparse data environments. In *SIGIR*, 2007.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[4] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[5] P. D. Bra, G. jan Houben, Y. Kornatzky, and R. Post. Information retrieval in distributed hypertexts. In *In RIAO*, 1994.

[6] U. Brefeld and T. Scheffer. Co-em support vector learning. In *ICML*, 2004.

[7] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *WWW*, 1999.

[8] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM TIST*, 2, 2011.

[9] M. Chen, K. Weinberger, and Y. Chen. Automatic feature decomposition for single view co-training. In *ICML*, 2011.

[10] C. M. Christoudias, R. Urtasun, and T. Darrell. Multi-View learning in the presence of view disagreement. In *UAI*, 2008.

[11] N. Cristianini and J. Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, 2000.

[12] S. Das, C. L. Giles, P. Mitra, and C. Caragea. On identifying academic homepages for digital libraries. In *JCDL*, 2011.

[13] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *JMLR*, 13, 2012.

[14] G. Druck, G. Mann, and A. McCallum. Learning from labeled features using generalized expectation criteria. In *SIGIR*, 2008.

[15] J. Du, C. Ling, and Z.-H. Zhou. When does cotraining work in real data? *Knowledge and Data Engineering, IEEE Transactions on*, may 2011.

[16] C. Fellbaum. *WordNet: An electronic lexical database*. MIT Press, 1998.

[17] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *ICML*, 2002.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11, 2009.

[19] J. Junghoo Cho, H. Garcia-Molina, and L. P. Page. Efficient crawling through url ordering. In *WWW*, 1998.

[20] M.-Y. Kan and H. O. N. Thi. Fast webpage classification using url features. In *CIKM*, 2005.

[21] H. S. Koppula, K. P. Leela, A. Agarwal, K. P. Chitrapura, S. Garg, and A. Sasturkar. Learning url patterns for webpage de-duplication. In *WSDM*, 2010.

[22] H. Li, I. G. Councill, L. Bolelli, D. Zhou, Y. Song, W.-C. Lee, A. Sivasubramaniam, and C. L. Giles. Citeseerx: a scalable autonomous scientific digital library. In *InfoScale*, 2006.

[23] X.-Y. Liu and Z.-H. Zhou. The influence of class imbalance on cost-sensitive learning: An empirical study. In *ICDM*, 2006.

[24] B. Long, P. S. Yu, and Z. M. Zhang. A general model for multiple view unsupervised learning. In *SDM*, 2008.

[25] G. S. Mann and A. McCallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *ICML*, 2007.

[26] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[27] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI*, 1999.

[28] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

[29] G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38, 1995.

[30] K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *CIKM*, 2000.

[31] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI Workshop on Machine Learning for Information Filtering*, 1999.

[32] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *AAAI*, 1998.

[33] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2-3), 2000.

[34] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[35] J.-L. Ortega-Priego, I. F. Aguillo, and J. A. Prieto-Valverde. Longitudinal study of contents and elements in the scientific web environment. *Journal of Information Science*, 32(4), 2006.

[36] X. Qi and B. D. Davison. Web page classification: Features and algorithms. *ACM Comput. Surv.*, 41(2), Feb. 2009.

[37] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, May 2000.

[38] L. K. Shih and D. R. Karger. Using urls and table layout for web classification tasks. In *WWW*, 2004.

[39] V. Sindhwani, P. Niyogi, and M. Belkin. A Co-Regularization approach to semi-supervised learning with multiple views. In *ICML*, 2005.

[40] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: extraction and mining of academic social networks. In *KDD*, 2008.

[41] V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[42] Y. Wang and K. Oyama. Web page classification exploiting contents of surrounding pages for building a high-quality homepage collection. In *ICADL*, 2006.

[43] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL*, 1995.

[44] H. Yu, J. Han, and K.-C. Chang. Pebl: Web page classification without negative examples. *IEEE TKDE*, jan. 2004.

[45] X. Zhu. Semi-Supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison, 2005.