Improving Researcher Homepage Classification with Unlabeled Data

SUJATHA DAS GOLLAPALLI, The Pennsylvania State University
CORNELIA CARAGEA, University of North Texas
PRASENJIT MITRA and C. LEE GILES, The Pennsylvania State University

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 changing content on the Web? We investigate this question in the context of identifying researcher homepages. We show experimentally that classifiers trained on existing datasets of academic homepages underperform on "non-homepages" present on current-day academic websites. As an alternative to obtaining labeled datasets to retrain classifiers for the new content, in this article we ask the following question: "How can we effectively use the unlabeled data readily available from academic websites to improve researcher homepage classification?"

We design novel URL-based features and use them in conjunction with content-based features for representing homepages. Within the co-training framework, these sets of features can be treated as complementary views enabling us to effectively use unlabeled data and obtain remarkable improvements in homepage identification on the current-day academic websites. We also propose a novel technique for "learning a conforming pair of classifiers" that *mimics* co-training. Our algorithm seeks to minimize a loss (objective) function quantifying the difference in predictions from the two views afforded by co-training. We argue that this loss formulation provides insights for understanding co-training and can be used even in the absence of a validation dataset.

Our next set of findings pertains to the evaluation of other state-of-the-art techniques for classifying homepages. First, we apply feature selection (FS) and feature hashing (FH) techniques independently and in conjunction with co-training to academic homepages. FS is a well-known technique for removing redundant and unnecessary features from the data representation, whereas FH is a technique that uses hash functions for efficient encoding of features. We show that FS can be effectively combined with co-training to obtain further improvements in identifying homepages. However, using hashed feature representations, a performance degradation is observed possibly due to feature collisions.

Finally, we evaluate other semisupervised algorithms for homepage classification. We show that although several algorithms are effective in using information from the unlabeled instances, co-training that explicitly harnesses the feature split in the underlying instances outperforms approaches that combine content and URL features into a single view.

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

General Terms: Algorithms

Additional Key Words and Phrases: Researcher homepage classification, co-training, conforming classifiers, unlabeled data

---

This research was supported in part by National Science Foundation (NSF) awards 1423337 and 1422951 to C. Caragea and C. L. Giles. Any opinions, findings, and conclusions expressed here are those of the authors and do not necessarily reflect the views of the NSF.
Authors' addresses: S. D. Gollapalli, 5 Dover Crescent, 03-14, Sinagpore, 130005; email: sujatha.das@gmail.com; C. Caragea, F228 Discovery Park, Computer Science and Engineering, University of North Texas, Denton, TX 76203, USA; email: ccaragea@unt.edu; P. Mitra, 313F IST Building, The Pennsylvania State University, University Park, PA 16802, USA; email: pmitra@ist.psu.edu; C. L. Giles, 311A, IST Building, The Pennsylvania State University, University Park, PA 16802, USA; email: giles@ist.psu.edu.

DOI: http://dx.doi.org/10.1145/2767135

ACM Reference Format:
Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2015. Improving researcher homepage classification with unlabeled data. ACM Trans. Web 9, 4, Article 17 (October 2015), 32 pages.
DOI: http://dx.doi.org/10.1145/2767135

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 [Gollapalli et al. 2011]. Researchers' homepages (also referred to as academic homepages or simply homepages in this article) have been successfully employed in tasks such as expertise search [Balog et al. 2007], extraction of academic networks, author profile extraction, and disambiguation [Tang et al. 2008], 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 to retrieve appropriate scientific research publications. For instance, in an independent experiment where paper titles from DBLP[4] were used to search the Web, we were able to obtain about 35% of the papers available online for free from the author homepages.

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 [Chakrabarti et al. 1999]. 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[5] that was collected 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, and events that do not occur in the WebKB dataset. We refer to university, department, and research center websites as academic websites in this article. Compared to a 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 [Ortega-Priego et al. 2006].

How can a homepage classifier keep up in the face of the 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? Semisupervised 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 the fact that labeling examples for any supervised learning problem requires intensive human labor [Nigam et al. 2000].

[1]http://citeseerx.ist.psu.edu.
[2]http://arnetminer.org/.
[3]http://scholar.google.com/.
[4]Details of this experiment are described in Section 9.
[5]http://www.cs.cmu.edu/~webkb/.

Against this background, one question that can be raised is this: can we design techniques to effectively adjust the previously trained classifier to the changing and diverse 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 article addresses specifically this question.

Contributions and organization. We address the homepage classification problem using the (readily available) *unlabeled* data from academic websites, originally introduced in Gollapalli et al. [2013].[6] In this extended work, we augment our contributions to academic homepage classification with our findings from a larger spectrum of experiments using other state-of-the-art classification approaches and newer datasets, specifically created for this task. Our contributions are summarized 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, resulting in an unacceptable yield from the perspective of a focused crawler.

—We design novel features based on URL surface patterns and terms to complement term and HTML features extracted from the content of homepages and show that these two sets of features can be treated as independent "views" for a researcher homepage. In a co-training setup, these views enable us to use the unlabeled data to successfully adapt classifiers to the changing academic environments.

We show co-training experiments on three different datasets. Our first dataset consists of homepages of computer science researchers from several U.S. universities previously used in Gollapalli et al. [2013]. We also include experiments on two newly constructed datasets: (1) a dataset compiled from ArnetMiner, which contains homepages of computer science researchers from research institutes and non-U.S. universities, and (2) a multidisciplinary homepage (MH) dataset, which contains homepages of researchers from disciplines other than computer science, such as physics, chemistry, and environmental sciences. The positive results obtained on all three datasets illustrate the generalizability of our proposed features and methods in identifying researcher homepages.

—Next, we design an iterative algorithm based on mini-batch gradient descent to minimize the disparity between the classifiers' predictions on the two views used in co-training. We show that with this formulation, we can effectively *mimic* the co-training process without requiring a validation dataset to track its progress.

—We study the feature selection (FS) and feature hashing (FH) techniques on the two views of homepages independently and in conjunction with co-training. FS removes redundant and irrelevant features from the data representation [Yang and Pedersen 1997; Forman 2003], whereas FH eliminates the need for a look-up dictionary by implicitly encoding it into a hash function [Weinberger et al. 2009]. We show that FS effectively improves homepage classification by removing the nondiscriminative features, whereas FH results in performance degradation.

—Finally, we answer the question "Is the two-view approach better for classifying homepages?" by performing a comprehensive set of experiments using other state-of-the-art semisupervised approaches that treat the URL and content features as a single view. We show that although most of these techniques improve the classification performance over the base classifiers using unlabeled instances, co-training that explicitly harnesses the split of features emerges as the winner.

Although in this article we focus on the design of accurate approaches for researcher homepage classification, our objective is to integrate this classification component in

[6]http://www.cse.unt.edu/∼ccaragea/papers/www13.pdf.

the context of focused crawling and 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 the article 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. FS and FH techniques are briefly described in Section 5, whereas the list of other semisupervised approaches that we studied for homepage classification is presented in Section 6. Experimental setup, datasets, evaluation measures, and results are discussed in Section 7. We include discussions and preliminary experiments related to a potential future direction in multiview learning in Section 8, followed by a summary and future extensions to our work in Section 10.

2. RELATED WORK

Researcher homepage classification is a well-studied webpage classification problem in the context of digital libraries such as CiteSeer [Li et al. 2006] and ArnetMiner [Tang et al. 2008]. Typically, content-based term features and HTML structure-based features are used for classifying webpages [Qi and Davison 2009]. 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. [2008].

The problem of gathering a high-quality researcher homepage collection was studied for Japanese websites by Wang and Oyama [2006] using on-page and anchor text features. Tang et al. [2008] studied homepage acquisition from search engine results using researcher names as queries. 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. The quality of this collection depends crucially on the accuracy of our content (term) and URL-based classifiers.

Term features are commonly used for addressing classification problems involving textual data [Manning et al. 2008]. We summarize some relevant previous work involving URL features to show their general effectiveness in addressing tasks pertaining to the Web.

Kan and Thi [2005] used URL features to predict the prestige of a webpage as modeled by PageRank, whereas Baykan et al. [2011] identified the topic of a webpage without examining the content on the page using tokens from the URL string. Bar-Yossef et al. [2009] and Koppula et al. [2010] addressed the webpage de-duplication problem using URLs extracted from query and Web server logs. Their goal was to identify duplicate URLs that result from aliasing and redirections on Web servers, which generally make crawling and indexing inefficient. Shih and Karger [2004] used URL features for improving applications such as ad blocking and recommendation. URL strings were also used to design efficient algorithms for large-scale clustering on websites that publish webpages by running scripts against databases [Blanco et al. 2011]. All of these works illustrate that several webpage classification, clustering, and extraction tasks can be handled effectively based on URL features alone, thereby avoiding the overhead of examining content of webpages.

Focused crawling first proposed by Bra et al. is a rich area of research on the Web [Bra et al. 1994; Junghoo Cho et al. 1998]. Chakrabarti et al. [1999] present a discussion

Table I. Feature Types and the Size of Feature Sets Used in Homepage Identification

Type of Features	Features (#)
Content based	
Top unigrams	18,674
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	1,039

on the main components involved in building a focused crawler. Although focused crawling is our motivating application, this article 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 pretrained 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 [1998] first proposed co-training, an approach for semisupervised 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" [Nigam and Ghani 2000]. Recent research proposes techniques for decomposing the feature set into two views when such a split is not naturally available [Chen et al. 2011; Du et al. 2011].

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 [Sindhwani et al. 2005; Long et al. 2008; Christoudias et al. 2008]. Most solutions to multiview learning tend to frame the problem in terms of a global optimization problem and simultaneously learn classifiers for all of the underlying views. In some cases, the solutions depend on underlying classification algorithms used [Ghani 2002; Brefeld and Scheffer 2004]. 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 [Manning et al. 2008]. Homepage classification was previously studied as a text classification problem using term features [Nigam et al. 1998, 1999]. 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 [Gollapalli et al. 2011]. In this study, we extracted both content- and URL-based features from our training sets. These features and the size of feature sets are summarized in Table I. 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

Table II. Example URLs with Partial Sets of Extracted Features (Shown on the Next Line After Each URL)

	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

URL-based features based on surface patterns and their presence in WordNet.[7] The URL-based features are explained next.

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 that we encountered in our crawl, which are listed in Table II.

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 URLs (4) and (5), whereas the former seems to be a homepage, the latter seems to lead to a course page. The preceding 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," and 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 three times 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 [Miller 1995; Fellbaum 1998].

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 characters such as the tilde. These features are designed to filter out the URLs that commonly represent course pages, announcements, calendars, and other autogenerated content. For instance, a typical homepage URL string in computer science departments has the name of the researcher following the tilde 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 nondictionary term. Partial sets of extracted features are shown along with the URLs listed in Table II.

The preceding sets of features perform very well on the training datasets as shown in Section 7. In this work, we are particularly interested 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*

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

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 preceding 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 preceding 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 7), we were unable to extract URL features for about 27% of the instances. Therefore, content-based and URL features complement each other, whereas identifying homepage instances and a focused crawler might be required to use either or both of these sets of features.

4. USING CO-TRAINING FOR HOMEPAGE CLASSIFICATION

We show in our experiments (Section 7) 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. 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 these 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 in which the URL classifier makes mistakes, which the content-based classifier identifies correctly and vice versa.

Blum and Mitchell [1998] proposed co-training in the context of webpage classification. 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 were 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 [2000] 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*. In other words, 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

ALGORITHM 1: Procedure for Co-Training

```
Input: $L$, $U$, 's'
$L_1 \leftarrow L$, $L_2 \leftarrow L$
$\rho_1 \leftarrow \phi$, $\rho_2 \leftarrow \phi$
while $U \neq \phi$ do
    Compute $\rho_1$ using $\frac{|L_1^+|}{|L_1^-|}$, $\rho_2$ using $\frac{|L_2^+|}{|L_2^-|}$.
    Train classifier $C_1$ using ($L_1^1$, $\rho_1$).
    Train classifier $C_2$ using ($L_2^2$, $\rho_2$).
    $S \leftarrow \phi$
    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
Output: Classifiers $C_1$, $C_2$.
```

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 7 that illustrates the "new" types of webpages encountered in our crawl, potentially causing the pretrained content-based classifiers to underperform during crawling.

The number of negative instances encountered during our crawls is higher compared to the number of positive instances. Although 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 [Gollapalli et al. 2011]. 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 [Blum and Mitchell 1998]. We argue that avoiding this constraint is better in our scenario, as 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 [Bishop 2006; Liu and Zhou 2006]. We employ the idea of altering the misclassification 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 one-tenth the penalty incurred on making mistakes on a positive instance. For most implementations of classification algorithms, the mis-classification costs can be specified as a parameter during the training process [Hall et al. 2009].

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 views (content- 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 view 1 (or feature set 1) of the examples in L_1 is being used.

Based on previous studies in co-training [Blum and Mitchell 1998; Nigam and Ghani 2000], 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 semisupervised learning where confidently predicted unlabeled instances are added to the training set for retraining the classifier in subsequent rounds [Zhu 2005]. In contrast to self-training, which 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. In other words, 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 [Zhu 2005].

—*AddCrossRC*: This scheme is similar to AddCross with the additional ratio constraint (RC) on the number of positive and negative instances added in each round. This constraint was originally studied by Blum and Mitchell [1998] 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.

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. In other words, 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 [Balcan et al. 2005]. However, in practice, it is tricky to judge if co-training works for a problem and to verify if the assumptions are satisfied [Du et al. 2011]. 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 [Chen et al. 2011]. 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, such as support vector machines (SVMs) and maximum entropy (MaxEnt), output weight vectors capturing the importance of each feature as part of the training process [Bishop 2006].

One can expect co-training to benefit a classification problem if one classifier (e.g., C_1) can "guide" the other (C_2) on examples on which the latter makes mistakes. 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 [Sindhwani et al. 2005; Long et al. 2008; Christoudias et al. 2008]. The mismatch in predictions by C_1 and C_2 on unlabeled data can be quantified using a loss function. The mean squared error loss function commonly used in machine learning captures this loss as such:

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

The preceding formulation captures the average squared difference in predictions from the two views on unlabeled data. Here, $\mathbf{w_1}$ 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 (e.g., a function that outputs scaled distances from the separating hyperplane in case of SVMs). 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 to multiview learning methods, where learning the classifiers is folded into a global objective function in sophisticated ways [Sindhwani et al. 2005; Long et al. 2008; Christoudias et al. 2008], 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 (e.g., $\mathbf{w_1}, \mathbf{w_2} = \mathbf{0}$) 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 the two gradient descent steps is executed, whereas 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 [Nocedal and Wright 2006].

In each iteration, we employ mini-batch gradient descent to minimize the loss function, once with respect to $\mathbf{w_1}$ and next with respect to $\mathbf{w_2}$. 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 (online) gradient descent and batch gradient descent to obtain fast convergence during optimization by running gradient descent on small batches of randomly selected examples [Dekel et al. 2012].

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

ALGORITHM 2: Learning a Pair of Conforming Classifiers

```
Input: $\mathbf{w_1}$, $\mathbf{w_2}$, $U$, ‘s’, #oIters, #iIters, $\alpha$
$o = 0$
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: $\mathbf{w_1}$, $\mathbf{w_2}$
```

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. In other words, 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 the instance has a particular label with the logistic function [Bishop 2006]:

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

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, the conformity loss *directly* measures the effect of co-training as it is being applied. In contrast, Algorithm 1 is typically terminated either when no more examples are available or by tracking the performance on a validation dataset.

We provide a preliminary, experimental demonstration of the connection between co-training and our proposed algorithm in Section 7. 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. In Section 7, we show that our method can be used in lieu of a validation dataset for tracking the performance of co-training.

5. FEATURE SELECTION AND FEATURE HASHING

FS and FH methods involve different means of changing the feature representation used as input to machine learning algorithms. These methods have been widely used for text classification problems (e.g., see Yang and Pedersen [1997], Forman [2003], Weinberger et al. [2009], Shi et al. [2009], Forman and Kirshenbaum [2008], Langford et al. [2007], and Caragea et al. [2012]). In text classification, a document is generally represented using bag-of-words and n-gram approaches, which result in a dictionary of size d of all words or n-grams for a collection of documents. A text document is then represented as a vector $\mathbf{x}$ with as many entries as the number of words or n-grams in the dictionary. The entry k in $\mathbf{x}$ can record the frequency of word or n-gram k in the document, denoted by x_k, using appropriate normalization. Forman [2003] presented an extensive empirical analysis of FS for text classification and showed that FS applied to bag-of-words can make a learning task more accurate and efficient. Weinberger et al. [2009] and Shi et al. [2009] successfully used FH for a large-scale personalized email filtering problem and newswire article classification, respectively.

Given the success of FS and FH methods for large-scale text-related tasks, we study their applicability for homepage classification (independently and in conjunction with co-training). These methods are described next:

(1) FS methods target the removal of noninformative features using measures such as mutual information, correlation, or information gain (IG). We study FS using IG for homepage classification. Yang and Pedersen [1997] performed a comparative study of FS techniques on several corpora and reported IG to be among the most effective ones for text classification. IG measures the decrease in entropy given the information regarding the presence or absence of a feature. The "worth" of a feature with respect to a class is measured using $IG(C, f) = H(f) - H(C|f)$, where C, f, and H refer to the class label, feature, and entropy function, respectively [Bishop 2006].
(2) FH is a technique that eliminates the need for a look-up dictionary by implicitly encoding it into a hash function h.[8] For a text document, each token is directly mapped, using h, into a hash key, which represents the index of the token in the hashed feature vector. Note that multiple tokens can be mapped, through h, into the same hash key. Each index in the hashed vector stores the sum of "frequency counts" of all tokens that are hashed together into the same hash key. Weinberger et al. [2009] proved that for a feature vector $\mathbf{x}$ such that $\|\mathbf{x}\|_2 = 1$, the length of $\mathbf{x}$ is preserved with high probability for sufficiently large dimensions (or hash sizes) and sufficiently small magnitude of $\mathbf{x}$, such as $\|\mathbf{x}\|_\infty$ (lower and upper bounds are theoretically derived). As a consequence, for sufficiently large dimensions, not many collisions occur in the data due to hashing, thus resulting in performances similar to the bag-of-words approach. Hashing was effectively used in large-scale classification [Weinberger et al. 2009] and Web-related applications such as jointly modeling friendship and interest networks in social networks for *interest targeting* and *friendship prediction* [Yang et al. 2011].

Our goal in studying FS and FH for homepage classification is to answer the following questions:

[8]Note that h can be any hash function, such as `hashCode()` of the Java `String` class, or `murmurHash` function available online at http://sites.google.com/site/murmurhash/.

—How does the performance of FS and FH compare to that of bag-of-words for homepage classification?
—Is there value in combining FS and FH with co-training?

We compare the classification performance using all features, features selected using IG, and hash features obtained using `hashCode()` of the Java String class.

6. SEMISUPERVISED LEARNING

Semisupervised learning pertains to the use of unlabeled examples along with a few labeled examples for estimating parameters of machine learning algorithms [Bishop 2006]. The co-training [Blum and Mitchell 1998] algorithm is a generic procedure for including unlabeled instances into the learning process without reference to specific classification algorithms such as naive Bayes (NB) or SVMs. In contrast, in other semisupervised techniques, parameters are estimated using both labeled and unlabeled instances by including terms pertaining to both in the underlying objective functions (e.g., using regularization). Consequently, several existing classification algorithms have specific semisupervised variations. We compare co-training with the following semisupervised learning methods. Their choice is motivated with a view to choose state-of-the-art techniques in each category of models, namely probablistic (generative and discriminative) and nonprobabilistic [Bishop 2006]:

(1) *Naive Bayes multinomial with expectation maximization (NBM-EM)*: Nigam et al. [2000] suggest computing probabilistic labels on unlabeled examples and using them while estimating parameters for naive Bayes (NB) models. They use an iterative procedure based on the EM principle to repeatedly reassign probabilities over unlabeled examples until convergence is obtained. This procedure was shown to work well when the underlying data conforms to the generative assumptions of the model. Weighing factors for unlabeled instances and mixture models per class were also studied as extensions to the original formulation when such assumptions are violated [Nigam et al. 2000].
(2) *Generalized expectation (GE)*: Feature labeling was recently proposed by Druck et al. [2008] for discriminative classifiers. "Supervision" is provided in these models using (feature, label) affinities rather than fully annotated instances. As an example, consider the text classification problem where the documents are to be classified into classes: *homepage (+ve)* versus *nonhomepage (−ve)*. Even without looking at the entire document for assigning a label, from domain knowledge one can expect phrases such as *my research* or *my research interests* to be more associated with the positive class rather than the negative class. This intuition is captured by labeled features that express distributions such as { *'my research': +ve = 0.9, −ve = 0.1*}. For MaxEnt models, the objective function is suitably extended to incorporate these distributions in terms of expectation constraints. GE [Mann and McCallum 2010] and posterior regularization (PR) [Ganchev et al. 2010] can be used to enforce these expectation constraints while learning the models. Druck et al. [2008] proposed automated techniques for extracting labeled features given a labeled dataset.
(3) *Transductive support vector machines (TSVMs)*: TSVMs were proposed by Vapnik [1995] for improving the generalization accuracy of SVMs with the help of unlabeled data. This is achieved by adding an additional regularization term for unlabeled data to the objective function optimized by SVM. The test data (on which predictions are to be obtained) are used as unlabeled data to adjust the margin of classification in a transductive setting in TSVMs. TSVMs were shown to perform well on text classification problems because they inherently take co-occurrence

Table III. List of Seed URLs

http://www.cs.wisc.edu, http://www.cs.umich.edu, http://www.cs.umd.edu, http://www.cs.ucla.edu, http://www.cis.upenn.edu, http://www.cs.columbia.edu, http://www.cs.princeton.edu, http://www.eecs.berkeley.edu, http://www.cs.washington.edu, http://www.cs.brown.edu, http://www.cs.utexas.edu, http://www.cs.cornell.edu, http://www.eecs.mit.edu, http://cs.illinois.edu, http://www.cc.gatech.edu, http://www.cse.ucsd.edu

properties exhibited by words in text documents into account while adjusting the margin for separation [Joachims 1999].

7. EXPERIMENTS

We summarize our experiments next:

(1) We demonstrate the performance of both content- and URL-based features on the training and validation datasets.
(2) Next, we show that co-training can successfully address the problem of mismatch in the training and deployment environments for homepage classification.
(3) Then, we show that our proposed algorithm (Algorithm 2) achieves the same effect as co-training.
(4) We then evaluate FS and FH for homepage classification in conjunction with co-training and demonstrate that co-training out-performs all semisupervised approaches discussed in Section 6.
(5) Finally, we provide experiments on two newly created datasets to illustrate the generalizability of our proposed features and methods. Specifically, we include results of co-training on a dataset comprising of homepages from non-U.S. universities as well as research institutes and another dataset that has homepages of researchers from subject areas other than computer science.

7.1. Datasets

We describe the datasets available for studying academic homepage classification for the computer science discipline. The WebKB dataset was used previously by several researchers for studying webpage and text classification including semisupervised learning and co-training [Blum and Mitchell 1998; McCallum and Nigam 1999; Nigam et al. 1999, 2000]. The WebKB collection contains 8,282 academic webpages from computer science departments of four universities: Cornell, Washington, Texas, and Wisconsin, categorized into seven categories (student, faculty, staff, department, course, project, and other). The “other” class comprises pages that cannot be fit into the remaining six classes, such as 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 that we present shortly. Another set of author-provided homepages are available from the bibliographic resource for computer science and related areas, DBLP.[9] Although this collection of 6,000 homepages is more recent, we do not have negative instances as part of this dataset.

To mimic the deployment scenario, we crawled the university websites listed in Table III. These websites were selected arbitrarily from the list of top U.S. graduate schools in computer science (obtained from rankings in *U.S. News & World Report*[10]). We seeded our crawl with these URLs and used the open-source crawling software

[9]http://www.informatik.uni-trier.de/∼ley/db/.
[10]http://www.usnews.com/.

Table IV. Datasets Description

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

Note: a/b represents a instances out of which b are labeled positive.

Heritrix[11] (version 1.14.3) for obtaining all webpages of content-type "text/html," within a depth of five 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 universities listed in Table III and manually labeled them. From the remaining pages, another set of 500 pages were randomly chosen for validating or tuning the methods described in Sections 4, 5, and 6. 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. For labeled instances from DBLP (6,000 instances) and WebKB (8,282 instances), this could be done for 9,263 cases.We could extract both content and URL features for 145,245 out of 162,369 crawled pages. These pages are spread across unlabeled dataset (143,145 instances), validation dataset (500 instances), and test dataset (1,600 instances). A summary of the datasets[12] just described is shown in Table IV.

7.2. The ArnetMiner Dataset

The datasets described in the previous section are dominated by instances obtained from university websites located in the United States. Do the URL and content features continue to work well with homepages obtained from research institutes rather than university homepages? How well do our proposed methods generalize on (English) academic homepages obtained from other (non-U.S.) countries? Datasets that contain these types of homepages are not publicly available. Hence, we created a representative dataset compiled from ArnetMiner that we refer to as the *AM* dataset.

Tang et al. [2008] studied metadata extraction from researcher homepages as part of their ArnetMiner system. Their publicly available dataset contains 898 homepages annotated for researcher metadata. This dataset neither has examples of nonhomepages nor does it have the URL information corresponding to the webpages. To address these shortcomings, we created the AM dataset as described next:

—From the profiling dataset, a subset of researchers was selected randomly. The homepages corresponding to the researchers in this subset were located online by searching for the researcher name using the popular search engine Google. Next, a few links on the domain that hosts the homepage were manually examined to collect negative examples for our dataset. The AM dataset compiled in this fashion contains URLs and webpages pertaining to 113 homepages and 126 nonhomepages. The top-level and the country-code top-level domain names extracted from the URLs of these webpages are listed in Table V. As can be seen from these domain names, several pages in the AM dataset are from countries other than the United States. The dataset also has pages from nonuniversity websites such as ".com, .gov, .org."

[11]https://webarchive.jira.com/browse/HER.
[12]All raw datasets, dictionaries, and feature files are available upon request.

Table V. Number of Webpages per Hostname in the URLs for the AM Dataset

edu	24	com	24	it	19	ca	17	uk	16
org	12	de	11	jp	9	fr	8	pl	7
kr	7	il	7	gov	6	es	6	cz	6
au	6	sg	5	fi	5	be	5	tw	4
se	4	dk	4	br	4	no	3	nl	3
cn	3	sk	2	net	2	ie	2	hk	2
at	2	pt	1	in	1	eu	1		

7.3. Homepage Dataset for Researchers from Diverse Disciplines

We now address this question: Are homepages of researchers from other disciplines and subject areas similar to homepages of researchers from computer science and related areas? Intuitively, we expect some similarities to persist among researcher homepages across disciplines. For instance, cue phrases such as "publications," "faculty," "grants," and "research interests" may occur on homepages from other disciplines as well, such as mathematics and chemistry. How well do the URL and content-based features proposed by us generalize across disciplines? In particular, since we have labeled datasets of homepages of researchers in computer science and related areas, can we use them to identify homepages of researchers in other disciplines? To answer these questions, we collected the MH dataset, a dataset that includes homepages and nonhomepages from diverse disciplines.

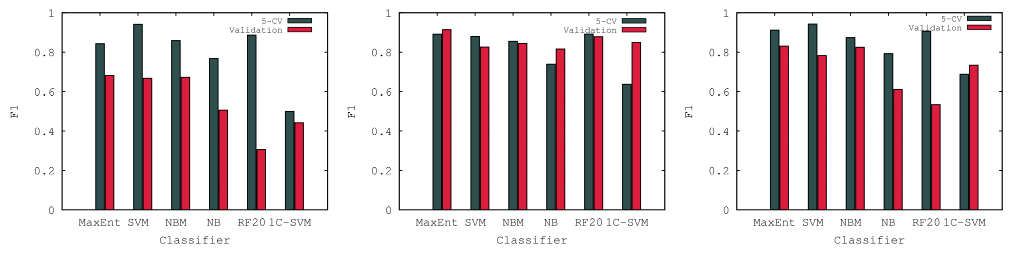
The E-print network[13] provides a listing of webpages related to researchers from a variety of disciplines, including environmental sciences, mathematics, and chemistry. These webpages include researcher homepages, group or lab homepages, and publication listings. To mimic our crawl scenario accurately, we collected the MH dataset as follows. After obtaining the webpages from E-prints, we randomly selected subsets of author names from five disciplines: biotechnology, chemistry, environmental sciences, geosciences, and mathematics. For each discipline, we manually located about 30 researcher homepages on the respective institute or university website using Google. From the same website, for every researcher homepage located, we collected about 3 nonhomepages to comprise the negative instances. In total, we were able to collect 149 researcher homepages and 448 nonhomepages to comprise the MH dataset. We discuss the performance of our classifiers on this dataset in Section 7.11.

7.4. Classification Experiments

Precision, recall, and F1 are standard measures to evaluate the performance of classification algorithms [Manning et al. 2008]. Weighted measures were proposed to evaluate classifiers on datasets that are highly imbalanced with respect to the number of examples from different classes [Witten et al. 2011]. Let pp and np denote the proportions of positive and negative examples in a dataset ($pp + np = 1$). If Prec_p denotes the precision of the positive class and Prec_n that of the negative class, the weighted precision is given by $pp \times \text{Prec}_p + np \times \text{Prec}_n$. The other measures, recall and F1, are similarly scaled using pp and np values to obtain their weighted counterparts. We use weighted measures to evaluate classification performance in our experiments.

We study the performance of our content-based and URL features using the classification algorithms: NB, NBM, random forests (RF), SVMs using a linear kernel (SVM), and MaxEnt. NB and NBM are generative models, whereas RF is an ensemble method using decision trees. Discriminative algorithms such as SVMs [Vapnik 1995; Cristianini and Shawe-Taylor 2000] and MaxEnt classifiers [Nigam et al. 1999] are

[13]http://www.osti.gov/eprints/pathways/.



(a) Content (b) URL (c) Content+URL

Fig. 1. Performance of various feature sets on homepage classification. Fivefold cross-validation performance is compared with that obtained on the validation dataset obtained from crawls.

being used extensively for text classification problems in 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., Bishop [2006]). The NB algorithm is well studied in the context of co-training and its theoretical analysis, although co-training is a classifier-independent technique. For text classification, the NBM classifier is different from NB in terms of its modeling of term counts using multinomial distributions [McCallum and Nigam 1999].

7.4.1. Classification Performance and Error Analysis. Figure 1(a) and (b) show the weighted $F1$ measure on our training WebKB+DBLP (fivefold cross-validation) and validation crawl datasets using the classification algorithms with content-based and URL features, respectively. We used classifier implementations provided by Weka [Hall et al. 2009], libSVM [Chang and Lin 2011], and Mallet [McCallum 2002]. Where applicable, we tuned the parameters on the training datasets for the best performance (e.g., the C parameter for SVM and number of trees in RF). Similarly, to handle potential imbalance in instances belonging to different classes during co-training, we use appropriate misclassification costs (the CostMatrix option in Weka and the "w" setting in libSVM).

As Figure 1(a) and (b) 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 and URL features. Based on these results, for all subsequent experiments, we chose the better performing of all classification algorithms—NBM, SVMs, and MaxEnt—for further study. The performance of URL-based classifiers is typically higher compared to that of content-based classifiers, 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, incorrectly labeling 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 seven 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, and slides.
(2) Webpages that describe code, widgets, scripts, and 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, and forms.
(5) Webpages pertaining to alumni-related information, job, and contest calls.

Table VI. Features Ranked Using Information Gain on Training and Crawl (test+validation) Datasets

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

Note: The overlapping features from both sets are shown in bold. The relatively high overlap in URL features explains why they perform well in both training and deployment environments.

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 IG [Bishop 2006] on the training and crawl (test+validation) datasets to understand the feature-class correlation between the two datasets. The top-10 features ranked by this measure, shown in Table VI, also point to the difference between the two environments. However, our aim is not to model new types of webpages; rather, we wish to learn a discriminator that isolates academic homepages from nonhomepages. In our experiments, we noticed that webpages belonging to the preceding new types (1) and (2) 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.

We now evaluate the following approaches for improving the performance of content-based classifiers for the new environment. First, 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 [Schölkopf et al. 2000; Yu et al. 2004]. One-class methods are typically used for outlier 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 set is better, but not comparable to the best-performing classifiers on the validation set (see Figure 1(a)–(c), 1C-SVM).

Next, we learned classifiers on the combined set of features for each instance (content+URL). The performance of these classifiers is shown in Figure 1(c). As the figure indicates, when compared to Figure 1(a) and (b), using the combined set of features on the validation set is better than using content features alone, but still worse than using URL features alone.

7.4.2. Feature Extraction and Classification Times. As discussed in Section 2, several previous works noted the efficiency advantages in using a short URL-based string for performing Web-based tasks. This benefit is particularly significant in the context of focused crawling. If a webpage can be classified accurately based on the URL string alone, the overhead of downloading and parsing the webpage to make a content-based decision can be avoided. We provide the classifier training, prediction, and feature extraction times for our feature sets in Table VII. We show runtime results for NBM, SVMs, and MaxEnt (chosen due to reasons explained in the previous section). We performed all experiments five times with no other process running on our experimental machine[14] and summarize the average times in Table VII.

[14]All experiments for this work were run on a 16-core, 800MHz, 32GB RAM, AMD Opteron, Linux server.

Table VII. Average Feature Extraction and Prediction Times per Instance and Time for Training the Classifiers

Classifier	Features	Model Training Time (ms)	Average Prediction Time (ms)
MaxEnt	Content	2,927.8	2.2524
SVM	Content	103,850.0	9.7064
NBM	Content	513.4	0.0604
MaxEnt	URL	1,233.6	1.0620
SVM	URL	3,236.8	0.18320
NBM	URL	110.4	0.0156
Average time to extract URL features per instance			0.1009
Average time to extract content features per instance			4.8353
Average time to parse webpage content per instance			2.1433

Note: All times are in milliseconds (ms).

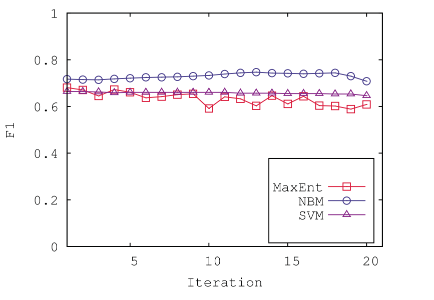
The training times of the classifiers depend on the number of features, the number of training instances, the actual algorithms used for parameter estimation, and their implementations. For example, the NBM classifiers only use the counts of features seen during training for computing model parameters, whereas the MaxEnt and SVM trainers estimate model parameters by formulating and solving an optimization problem over the training instances. Despite these factors, it is clear from Table VII that the URL-based classifiers are orders of magnitude faster during training and prediction phases compared to the content-based classifiers.

We also specify the times for extracting content and URL features per training instance (webpage) in Table VII. In an actual crawl, we also need to account for the (content) download times. Since our experiments were performed in the offline setup, we were unable to record these times in the table. However, webpage download times depend on other settings in the crawler, such as timeout/retry intervals and external factors such as Web traffic and robots.txt settings on the Web servers of crawl sites. As the numbers in Table VII indicate, from the perspective of efficiency it is clearly advantageous when an accurate decision can be made based on URL strings alone. The download, HTML parsing, and content feature extraction overhead can potentially be avoided when accurate URL-based classification is possible.

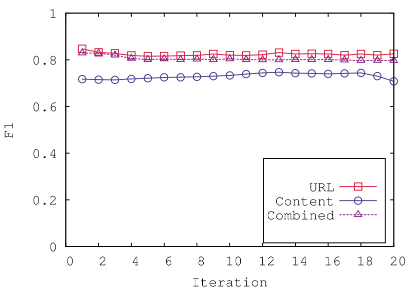
7.5. Self-Training Experiments

Self-training, an iterative approach commonly used in semisupervised learning, works as follows. Unlabeled examples, predicted confidently by the classifier in one iteration, are added back to enlarge the training dataset that is used to retrain the classifier for the subsequent iteration [Yarowsky 1995]. The intuition behind self-training has been compared to pseudorelevance feedback employed in information retrieval [Nigam and Ghani 2000; Manning et al. 2008]. Although this approach was successfully applied to some problems before, it is known to suffer from cascaded errors resulting in the final classifier being less effective than the initial one for some applications [Zhu 2005].

We evaluate the performance of self-training on the validation dataset using our sets of features. The results of self-training on content features using different training algorithms is illustrated in Figure 2(a). As can be seen in this figure, compared to the performance at iteration 1 (start of the self-training process), the validation performances as the iterations proceed are not significantly better. Indeed, depending on the classification algorithm, decreased accuracy can be observed *on the validation set* (MaxEnt). In our experiments, we found self-training to not benefit the URL classifier or the classifier trained using the combined set of content and URL features. This behavior is illustrated using the NBM classifier in Figure 2(b).

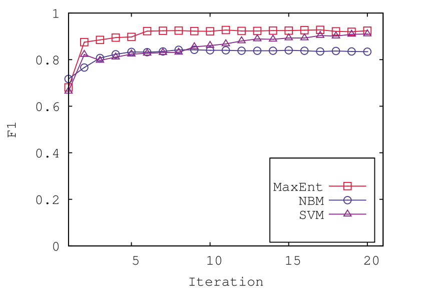


(a) Different classification algorithms and content features.

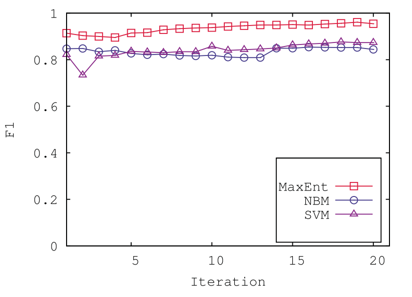


(b) Different feature sets and NBM.

Fig. 2. Performance on the validation dataset with self-training.



(a) Co-training (content).



(b) Co-training (URL).

Fig. 3. Performance on the validation dataset with co-training on URL and content features with the AddCross scheme.

7.6. 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 5,000 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. We set the number of co-training iterations to 20. The performance on the validation dataset was found to converge for both URL and content classifiers with these settings. Note that the labeled training set is WebKB+DBLP, whereas the unlabeled training set is the data from our crawl.

Figure 3 shows the performance of co-training with content-based and URL classifiers *on the validation set* using AddCross. Co-training successfully manages to pull up the performance of the content-based classifiers using the unlabeled data. The URL classifiers 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 3(b)).

We evaluated the different schemes—AddBoth, AddCross, and AddCrossRC—for adding unlabeled examples during co-training. Figure 4 illustrates these runs using

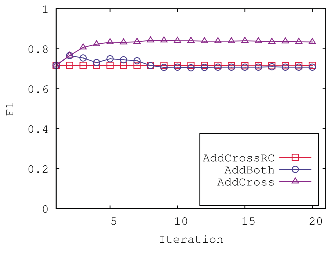


Fig. 4. Co-training schemes (NBM).

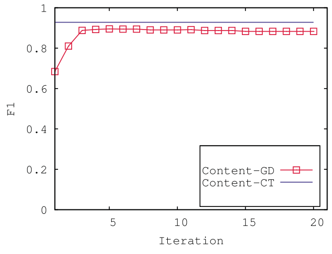


Fig. 5. GD with MaxEnt (content).

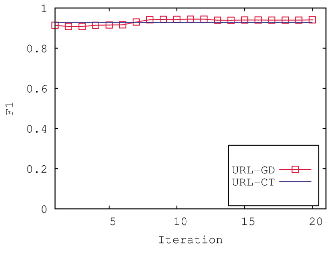


Fig. 6. GD with MaxEnt (URL).

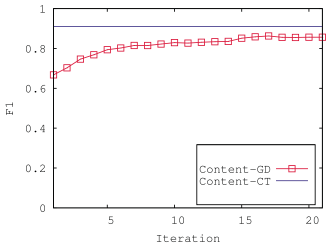


Fig. 7. GD: Content (SVM).

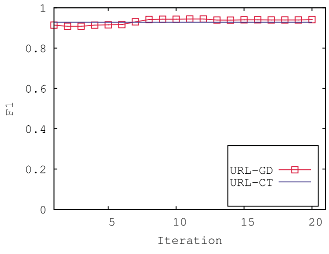


Fig. 8. GD: URL (SVM).

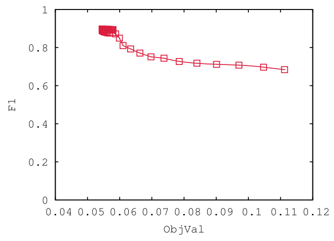


Fig. 9. ObjVal vs. F1.

the NBM classifier on URL and content features. As can be noticed in these plots, different addition schemes result in different co-training performance using the NBM classifier. However, with SVM and MaxEnt algorithms as base classifiers, we noticed no significant differences in the performance using the different schemes. The AddCrossRC scheme also performs similarly, except it takes longer to converge as only a constant number of examples $(p + n)$ are added in each iteration, where $p : n$ is the ratio of positive to negative instances, in the original labeled set. In general, AddCross performs either better than or on par with the other schemes. Since in Algorithm 2 we use a similar scheme as in co-training for adding instances from the unlabeled to the labeled set, we chose AddCross for comparisons with gradient descent shown in the next section.

Not accounting for class imbalance via the misclassification costs results in performance degradation over the co-training iterations. This happens due to the large number of negative instances in the unlabeled data that are moved to the training set during co-training iterations. On training with these skewed datasets, the performance on the positive 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 RC being maintained over the iterations, as we started with an almost balanced training dataset (see Table IV for data distribution).

7.7. Gradient Descent Experiments

Figures 5 and 6 show a run of Algorithm 2 with initial weight vectors obtained with MaxEnt classifiers on content and URL features, respectively. The 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 7 and 8.

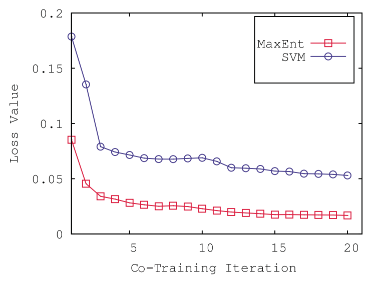


Fig. 10. Reducing value of the squared-error loss as co-training progresses.

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

Fig. 11. Features that are most affected after running Algorithm 2 on MaxEnt weight vectors.

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 section) alongside the curves in Figures 5 through 8 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 preset 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.

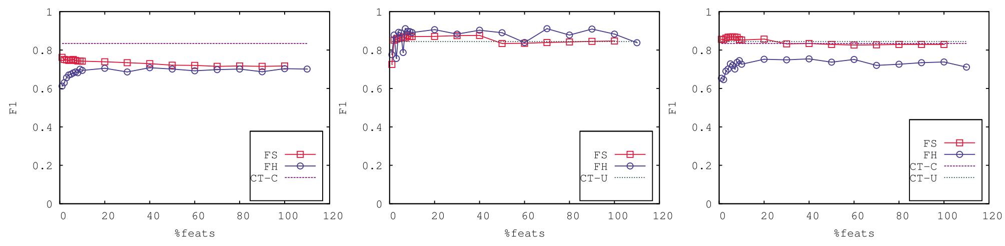
We observed the objective values to be converging in about 20 iterations when the algorithm is initialized with the weight vectors from MaxEnt, whereas it takes about 50 iterations with those from SVM. These values correspond to the $\#oIters$ in Algorithm 2. The $\#iIters$ and α values for mini-batch 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 at least 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 [Nocedal and Wright 2006]. Experimenting with these parameters is left for future work. With the settings just described, the runtimes 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 5 minutes and 4 hours on our experimental machine (described in Section 7.4.2).

We plot the $F1$ on the validation dataset against the computed objective function value in Figure 9. The plot depicts the close correspondence between reducing the discrepancy between predictions based on the two views and the improved 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 10. 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 11.

Table VIII compares the performance obtained with co-training and the gradient descent algorithms. The "After-CT" entries show the performance of 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 (classifiers) obtained after running

Table VIII. Performance on the Test Dataset Using Content Features after Co-Training (CT) and Gradient Descent (GD)

Method	Precision	Recall	F1
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



(a) Content (b) URL (c) Content+URL

Fig. 12. FS and FH experiments with NBM.

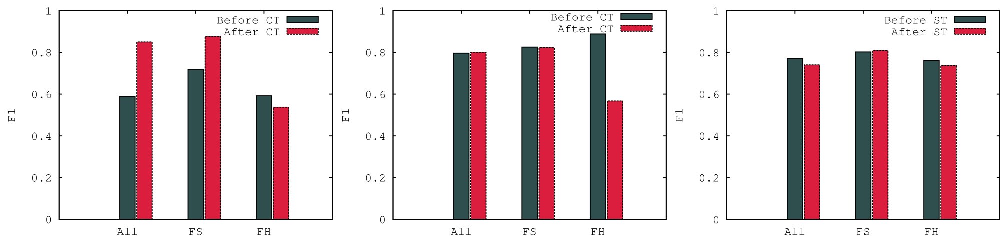
Algorithm 2, starting with the original weight vectors. As can be seen in this table, the performance of our proposed algorithm closely matches the performance obtained with co-training.

7.8. Feature Selection and Feature Hashing Experiments

In this section, we present results of our FS and FH experiments on the various sets of features using the NBM classifier. Figure 12 shows the performance on the validation dataset with different numbers of chosen content, URL, and content+URL features. The points indicate the $F1$ obtained on the validation dataset for a given percentage of the total features in the corresponding view. For example, 1% of 1,039 URL features corresponds to the top-10 URL features selected using IG on the FS curve and refers to the number of hash buckets used in FH.

From Figure 12, we notice that when compared to the performance including all features (the point corresponding to 100% on the FS curve), several other configurations that yield better validation performance compared to this point are possible. Indeed, we can see several points higher on y-axis compared to the point corresponding to FS 100%. These observations are consistent with previous research. FS that removes non-informative features obtains benefits over including all features, whereas FH performs similarly or worse compared with classifiers that use all features. Note that the point corresponding to 100% on the FH curve differs from the setting when all features are included due to potential collisions caused by the hash function. However, when using hashed features, due to the encoding via the hash function, there is no need to store dictionaries while generating the feature representations, thus making them efficient for large-scale classification problems [Weinberger et al. 2009; Caragea et al. 2012].

As illustrated in Figure 12(c), even with FS or FH, the performance obtained using the combined set of features (content+URL) is lower than that obtained by the individual classifiers after co-training. From Figure 12(b), we can see that using FS and FH, the URL classifier has configurations for which the performance is even better than that obtained with co-training and all features. This observation motivates our next



(a) Content (b) URL (c) Content+URL

Fig. 13. Performance on the test dataset with FS and FH.

Table IX. Performance on the Test Dataset after Co-Training/Self-Training

Features	Precision	Recall	F1
URL (All)	0.944	0.731	0.800
—with FS	0.936	0.761	0.822
—with FH	0.933	0.455	0.567
Content (All)	0.934	0.803	0.850
—with FS	0.918	0.847	0.876
—with FH	0.921	0.423	0.537
Content+URL (All)	0.941	0.651	0.740
—with FS	0.941	0.742	0.808
—with FH	0.939	0.646	0.736

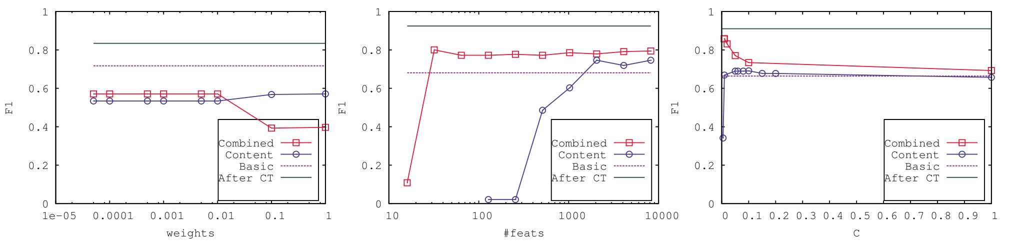
set of experiments, where we check the performance of co-training after applying FS and FH on the URL and content views independently.

The performances before and after applying co-training and self-training with all features (All) and with features selected using IG (FS), and with hashed feature representations (FH), are shown on content, URL, and content+URL features in Figure 13(a), (b), and (c), respectively, on the *test* dataset. The number of features (or hash buckets) that give the best performance on the validation set (previous experiment) are used in this experiment. We can see that FS combined with co-training can offer further improvements in identifying homepages.

Table IX shows the performance on the test dataset after co-training (or self-training) using all features (All), top selected features by IG (FS), and hash features (FH). We see from the table that when compared to using all features, first performing FS and then applying co-training or self-training results in about 2% to 9% F1 improvements on the test dataset. However, the performance using hashed representations degrades when jointly used with co-training. A potential reason for this behavior could be that collisions happen during the co-training iterations in such a way that features observed in unlabeled instances (but not in labeled instances) get mapped via the hash function to the same aggregated feature (“bucket”), resulting in accumulation of noise over the iterations.

7.9. Comparison with Other Semisupervised Learning Approaches

The co-training algorithm enhances the content classifier performance using predictions from the URL classifier on unlabeled data. For a fair comparison, we combine the URL and content features to form a single view before applying the semisupervised learning algorithms discussed in Section 6. Our experiments indicate that this setting enables semisupervised algorithms to harness co-occurrence properties of features to yield improvements over the base classifiers. In contrast, minimal or no improvements are noticed when semisupervised learning is used with content features only.



(a) NBM with EM (b) Generalized Expectation (c) Transductive SVM

Fig. 14. Comparison of semisupervised methods.

We describe our experiments with the semisupervised techniques listed in Section 6: NBM-EM, GE, and TSVM. We show the performance on the validation dataset for different parameters of the algorithms in Figure 14. Since our objective is to improve the content-based classifier for cases where URL features are absent (Section 3), we explicitly tune the parameters using the content features of the validation datasets for the final evaluation (on the test dataset).

In Figure 14, the “Basic” plots refer to the base performance when the classifier is trained using content features of labeled instances and evaluated on the content view of the validation instances. The “After CT” plots refer to performance of the content classifier after co-training. The “Combined” plots show the performance NBM-EM, GE, and TSVM using the combined (URL and content) set of features, whereas the “Content” plots show the performance using content features only. These plots show performance of the semisupervised algorithms for different choices of parameters as described later.

Figure 14(a) shows the performance of the NBM-EM model with different settings of weights for unlabeled instances. The performance degrades over the EM iterations for a wide range of weight choices. It was observed in previous studies that this behavior is common when the underlying data does not conform to the assumed generative distribution, and mixture distributions were proposed as a solution to fix this problem [Nigam et al. 2000]. Although this seems like a possible explanation for low performance of NBM-EM for our problem, in the homepage crawling setting, given the different and diverse types of “negative instances” encountered during the crawl, it is not possible to estimate the mixture parameters a priori (Section 7.4).

Next, we experiment with the MaxEnt classifier. For using the feature labeling framework, we have a choice of options: using labeled training instances versus validation instances in the transductive setting, PR and GE for enforcing the feature constraints and labeled features extracted using a previously proposed heuristic, or IG as suggested in Druck et al. [2008]. We experimented with these options in Mallet [McCallum 2002] and show the best setting (GE with labeled instances and labeled features extracted based on IG) in Figure 14(b). The plots show performance of the GE trainer using different numbers of labeled features.

For TSVMs, the data on which predictions are to be made is included during training as unlabeled data (the transductive setup). We used the TSVM implementation provided in the SVMLight package.[15] Figure 14(c) illustrates the sensitivity of TSVMs to the value of the margin parameter, “C.”

Both GE and TSVMs yield improvements over the base classifiers (Basic plots) when the combined view of features is used (Figure 14(b) and (c)). The Basic plots refer to the performance when the classifiers are trained on labeled instances only (without

[15]http://svmlight.joachims.org/.

Table X. Performance on the Test Dataset with Different Semisupervised Approaches

Method	Precision	Recall	F1
Naive Bayes Multinomial			
Basic	0.932	0.689	0.770
Co-training	**0.934**	**0.803**	**0.850**
Maximum Entropy			
Basic	0.8806	0.4167	0.5380
Co-training	**0.9234**	**0.9295**	**0.9262**
GET (Combined)	0.8817	0.7265	0.7925
Support Vector Machines			
Basic	0.9413	0.4947	0.6048
Co-training	**0.9147**	**0.8167**	**0.8559**
TSVM (Combined)	0.8691	0.6409	0.7334

Table XI. Co-Training and Self-Training on the AM dataset (NBM)

Feature Set	Precision	Recall	F1
URL	0.613	0.586	0.573
—after CT	0.703	0.703	0.703
Content	0.755	0.745	0.744
—after CT	0.76	0.753	0.753
Content+URL	0.646	0.636	0.634
—after ST	0.649	0.649	0.649

Note: The sample sizes are set to 10, and the algorithms are run in transductive mode.

any unlabeled data). The algorithms when run on combined view are able to harness co-occurrence with URL features to show improvements beyond the Basic performance using unlabeled data. In contrast, using content features alone, the algorithms do not show significant improvements.

As can be seen in Figure 14(b) and (c), co-training outperforms both GE and TSVMs by a large margin on the validation dataset. Based on these experiments, we choose the best-performing semisupervised approaches and parameter settings for the final evaluation on the test dataset in Table X.

7.10. Performance on the AM Dataset

We show the results of co-training and self-training using the AM dataset to illustrate the applicability of our proposed techniques to webpages from other academic domains. As described in Section 7.2, the AM dataset is a small collection of webpages from research institutes and non-U.S. universities. Table XI shows the performance, on the AM dataset before and after co-training (AddCrossRC, Section 4), of the URL and content classifiers trained on the DBLP+WebKB dataset (see Table IV for the dataset description) using NBM. Since unlabeled data from appropriate domains is unavailable for the AM dataset, we run the co-training algorithm in transductive mode. In this configuration, the test (AM) data on which predictions are to be made is treated as unlabeled data [Blum and Mitchell 1998; Nigam and Ghani 2000].

Notice that in contrast to the experiments on datasets related to U.S. universities, the content-based classifier does better than the URL classifier on the AM dataset in Table XI. The reason for this behavior is indicated in Figure 15, where we show the top-10 URL features on the training and AM datasets based on IG. It appears that the URL features corresponding to the non-U.S. webpages are different from those in

DBLP+WebKB	AM
TILDENODICT	**TILDENODICT**
TILDENODICT_SEQEND	person
ALPHANUMBER	research
NONDICTWORD	NUMBER
courses	content
ALPHANUMBER_SEQEND	who
users_NONDICTWORD	personal
users	TILDESURFPAT
NONDICTWORD_SEQEND	**ALPHANUMBER**
homes	seminars

Fig. 15. Overlap in the top-10 features (IG) of training and AM datasets.

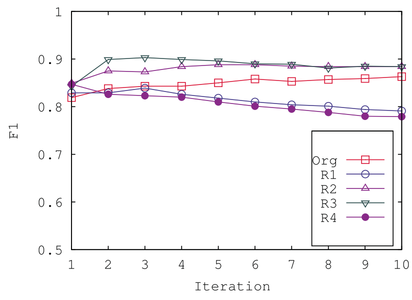


Fig. 16. Comparison performance with random splits and original split.

Table XII. Performance of URL and Content Classifiers on the Institute Webpages from the ArnetMiner Dataset and the MH Dataset

Dataset	Pos/Neg Instances	Features	Precision	Recall	F1
MH (Other Disciplines)	149/448	URL	0.832	0.743	0.761
		Content	0.797	0.649	0.673
AM (Research Institutes)	18/26	URL	0.582	0.578	0.580
		Content	0.739	0.721	0.724

Note: The URL and content classifiers were trained using the NBM algorithm and co-training on the training and unlabeled datasets from Table IV.

the training dataset, unlike our previous test dataset based on U.S. university crawls (Table VI). This difference once again highlights why it is beneficial to have accurate and separate classifiers on the two independent views. Depending on the environment, one set of features may be more valuable for discriminating the correct set of pages. The encouraging improvements in the classification measures on the AM dataset after co-training show that our proposed techniques have the potential to work well on webpages from other academic domains.

7.11. Performance on Webpages from Research Institutes and Diverse Disciplines

Table XII summarizes the performance of our URL and content classifiers on the MH dataset described in Section 7.3. The table also includes the performance of our classifiers on webpages corresponding to researchers from research institutes available from the AM dataset. In other words, the entries in the AM dataset with URLs from ".com/.gov/.net/.org" websites (Table V) are evaluated separately. Since our training datasets are based on datasets from U.S. university websites of computer science departments, our goal in this experiment is to evaluate the generalizability of our proposed features and classifiers on datasets from diverse disciplines (different from computer science) as well as from research institutes.

As can be seen from the table, the performance of our features and classifiers, trained on computer science–related webpages, generalize well on both datasets: the MH dataset and the AM Research Institutes homepage dataset.

8. FUTURE DIRECTION: LEARNING FEATURE SPLITS

In classification problems related to webpages and images, multiple views such as URLs and target page content, and tagged text and pixel features, are directly available. However, consider a classification problem when such a feature split exists but is not known. Instead, all features are available as a combined set. Based on the experimental

Table XIII. Web Search Results Based on DBLP Paper Titles from 2014

Total Number of Searches	100
Number of papers not found on the Web	22
Number of papers available for a fee	54
Numbe of papers available for free (nonhomepage)	25
Number of papers available from author homepages	12
Overlap between "fees" and "free"	4
Overlap between "free" and "from homepages"	3

results presented so far, it can be concluded that when a split of features is available, co-training that explicitly makes use of this split tends to outperform other semisupervised approaches that use a single view. So, given a combined set of features, how can we find a feature split if it exists? Additionally, what comprises a good split?

To motivate the preceding future direction, we combined all content and URL features available with the homepage instances and performed multiple runs of co-training using random split of features into two balanced views. The results of co-training with these splits, including the original split of features into URL and content views, are illustrated using the "product classifier" performance curves in Figure 16. In co-training, the classifiers from the two views independently estimate predictive distributions for a test instance. Blum and Mitchell [1998] proposed using the normalized product of class probabilities from the independent views to obtain a single predictive distribution for an instance (which we refer to as the product classifier).

As illustrated in Figure 16, a "wrong" split of features could result in performance degradation compared to that obtained using the URL and content split. However, what is more interesting is that other splits of features are possible that can yield even better classification performance than that obtained with the known split. We propose finding such splits as a future direction to pursue in co-training research.

9. A DISCUSSION ON RESEARCH PAPERS AVAILABILITY FROM AUTHOR HOMEPAGES

Back in 2001, Lawrence performed a correlation analysis between the citations accumulated by papers and their online availability [Lawrence 2001]. The conclusion of this analysis was as follows: "Articles freely available online are more highly cited. For greater impact and faster scientific progress, authors and publishers should aim to make research easy to access." In this era, when researchers have access to the online space via their professional homepages, we examine if researchers indeed make their research publications available online to the extent that is possible (e.g., subject to copyright rules). Hence, to understand research paper availability from authors' homepages, we performed two experiments, as detailed next.

In the first experiment, we did a simple search on Google using paper titles available from DBLP to determine the online availability of the corresponding papers. The XML records from DBLP were parsed[16] to obtain a list of 198,849 paper titles published in the year 2014. From these titles, we randomly selected 100 titles and performed the following experiment on the Web using the popular search engine Google. We used the paper title in quotes as the query string (for *exact match*) and restricted the search results to be PDF files (using the option *filetype:pdf*).[17] We summarize the results of this experiment in Table XIII.

[16]http://dblp.uni-trier.de/xml/.

[17]The Web searches were performed by a graduate student from the University of North Texas who has significant experience with academic webpages. The search was done during the third week of January 2015.

From the 100 Web searches, we were able to locate 54 papers on publisher websites such as ACM and Springer for a fee; 25 papers were available for free on websites such as ResearchGate, conference websites, and so on; and 12 papers were available from the homepages of at least one of the authors of the papers searched for. About 9/12= 75% of the titles/papers found on author homepages were not found on other free websites. Note that the numbers in Table XIII do not add up to 100 due to overlaps. Overall, out of the 100 titles, we found the corresponding papers available online either for a fee or for free for 78 titles (and did not find the papers on the Web for 22 titles). Based on these numbers, about 12/78= 15% of the papers available online can be recovered from author homepages. From the perspective of an automated, free-access digital library such as CiteSeer, the papers available from homepages constitute about $12/(25 + 12 - 3)$= 35% of the overall "obtainable" content.

In the second experiment, we obtained the datasets of crawled PDF documents used in a previous study by Caragea et al. [2014]. These datasets consist of two sets of 1,000 manually labeled PDF documents sampled in the years 2011 and 2014 from the CiteSeer crawls. Each PDF document is annotated using one of the labels from the set = {*paper, slides, book, thesis, CV, others*}. The label "paper" refers to a research paper, whereas the label "others" is used for documents that cannot be annotated using one of the other labels in the set. In addition to the actual PDF file, the URL at which the document was obtained is recorded in these datasets.

We randomly selected 200 documents from each dataset (i.e., the 2011 and 2014 datasets) such that 100 documents were selected from those labeled as "paper" and the other 100 were selected from those labeled as "others" (in each dataset). We thus obtained a total of 400 documents and manually examined their URLs to record if the documents were obtained from an author homepage. We found that 58/200= 29% of the documents marked as "paper" were obtained from author homepages, whereas only one out of the 200 documents marked as "other" was obtained from a homepage. Based on these numbers, we can conclude that author homepages are indeed a good source for crawling research publications and avoiding "junk" (from the perspective of scientific digital libraries).

10. CONCLUSIONS

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 types of pages 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 indicate that even without a validation set, one can track the effect of the co-training process via our loss function. We also showed extensive comparisons using FS, FH, and other semisupervised learning approaches with co-training. From our experiments, on the homepage classification task, it appears that co-training that explicitly harnesses the split of features into two independent views outperforms other semisupervised approaches that treat all features as a single view.

In the 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.

ACKNOWLEDGMENTS

We are grateful to Kishore Neppalli and Corina Florescu for their help with various dataset construction and labeling tasks. We thank our anonymous reviewers for their detailed and constructive feedback that helped in improving the findings and presentation in this article.

REFERENCES

Maria F. Balcan, Avrim Blum, and Ke Yang. 2005. Co-training and expansion: Towards bridging theory and practice. In *Proceedings of Neural Information Processing Systems (NIPS'05)*.

Krisztian Balog, Toine Bogers, Leif Azzopardi, M. de Rijke, and Antal van den Bosch. 2007. Broad expertise retrieval in sparse data environments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'07)*. 551–558.

Ziv Bar-Yossef, Idit Keidar, and Uri Schonfeld. 2009. Do not crawl in the DUST: Different URLs with similar text. *ACM Transactions on the Web* 3, 1, 3:1–3:31.

Eda Baykan, Monika Henzinger, Ludmila Marian, and Ingmar Weber. 2011. A comprehensive study of features and algorithms for URL-based topic classification. *ACM Transactions on the Web* 5, 3, 15:1–15:29.

Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, New York, NY.

Lorenzo Blanco, Nilesh Dalvi, and Ashwin Machanavajjhala. 2011. Highly efficient algorithms for structural clustering of large websites. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*. 437–446.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT'98)*. 92–100.

Paul De Bra, Geert Jan Houben, Yoram Kornatzky, and Reinier Post. 1994. Information retrieval in distributed hypertexts. In *Proceedings of the 11th International Conference on Machine Learning (ICML'94)*.

Ulf Brefeld and Tobias Scheffer. 2004. Co-EM support vector learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*.

Cornelia Caragea, Adrian Silvescu, and Prasenjit Mitra. 2012. Combining hashing and abstraction in sparse high dimensional feature spaces. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI'12)*.

Cornelia Caragea, Jian Wu, Kyle Williams, Sujatha Das Gollapalli, Madian Khabsa, Pradeep Teregowda, and C. Lee Giles. 2014. Automatic identification of research articles from crawled documents. In *Proceedings of the Web-Scale Classification Workshop: Classifying Big Data from the Web colocated with WSDM*.

Soumen Chakrabarti, Martin van den Berg, and Byron Dom. 1999. Focused crawling: A new approach to topic-specific Web resource discovery. In *Proceedings of the 8th International Conference on World Wide Web (WWW'99)*. 1623–1640.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology* 2, 3, Article No. 27.

Minmin Chen, Kilian Weinberger, and Yixin Chen. 2011. Automatic feature decomposition for single view co-training. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*.

C. Mario Christoudias, Raquel Urtasun, and Trevor. Darrell. 2008. Multi-view learning in the presence of view disagreement. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI'08)*.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. 2012. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research* 13, 1, 165–202.

Gregory Druck, Gideon Mann, and Andrew McCallum. 2008. Learning from labeled features using generalized expectation criteria. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*. 595–602.

Jun Du, Charles X. Ling, and Zhi-Hua Zhou. 2011. When does cotraining work in real data? *IEEE Transactions on Knowledge and Data Engineering* 23, 5, 788–799.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research* 3, 1289–1305.

George Forman and Evan Kirshenbaum. 2008. Extremely fast text feature extraction for classification and indexing. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. 1221–1230.

Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research* 11, 2001–2049.

Rayid Ghani. 2002. Combining labeled and unlabeled data for multiclass text categorization. In *Proceedings of the 19th International Conference on Machine Learning (ICML'02)*. 187–194.

Sujatha Das Gollapalli, Cornelia Caragea, Prasenjit Mitra, and C. Lee Giles. 2013. Researcher homepage classification using unlabeled data. In *Proceedings of the 22nd International Conference on World Wide Web (WWW'13)*. 471–482.

Sujatha Das Gollapalli, C. Lee Giles, Prasenjit Mitra, and Cornelia Caragea. 2011. On identifying academic homepages for digital libraries. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries (JCDL'11)*. 123–132.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter* 11, 1, 10–18.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Proceedings of the 16th International Conference on Machine Learning (ICML'99)*. 200–209.

Junghoo Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. 1998. Efficient crawling through URL ordering. In *Proceedings of the 7th International Conference on World Wide Web (WWW-7)*. 161–172.

Min-Yen Kan and Hoang Oanh Nguyen Thi. 2005. Fast webpage classification using URL features. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM'05)*. 325–326.

Hema Swetha Koppula, Krishna P. Leela, Amit Agarwal, Krishna Prasad Chitrapura, Sachin Garg, and Amit Sasturkar. 2010. Learning URL patterns for webpage de-duplication. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM'10)*. 381–390.

John Langford, Lihong Li, and Alex Strehl. 2007. *Vowpal Wabbit Online Learning Project*. Technical Report.

Steve Lawrence. 2001. Free online availability substantially increases a paper's impact. *Nature* 411, 6837, 521.

Huajing Li, Isaac G. Councill, Levent Bolelli, Ding Zhou, Yang Song, Wang-Chien Lee, Anand Sivasubramaniam, and C. Lee Giles. 2006. CiteSeerX: A scalable autonomous scientific digital library. In *Proceedings of the 1st International Conference on Scalable Information Systems (InfoScale'06)*. Article No. 18.

Xu-Ying Liu and Zhi-Hua Zhou. 2006. The influence of class imbalance on cost-sensitive learning: An empirical study. In *Proceedings of the 6th International Conference on Data Mining (ICDM'06)*.

Bo Long, Philip S. Yu, and Zhongfei (Mark) Zhang. 2008. A general model for multiple view unsupervised learning. In *SDM*.

Gideon S. Mann and Andrew McCallum. 2010. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of Machine Learning Research* 11, 955–984.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, MA.

Andrew McCallum and Kamal Nigam. 1999. A comparison of event models for naive Bayes text classification. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. Retrieved September 5, 2015, from http://mallet.cs.umass.edu.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM* 38, 11, 39–41.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM'00)*. 86–93.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *Proceedings of the IJCAI Workshop on Machine Learning for Information Filtering*.

Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. 1998. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the 15th National/10th Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence (AAAI'98/IAAI'98)*. 729–799.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. 2000. Text classification from labeled and unlabeled documents using EM. *Machine Learning* 39, 2–3, 103–134.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization*. Springer.

Jose Luis Ortega-Priego, Isidro F. Aguillo, and Jos Antonio Prieto-Valverde. 2006. Longitudinal study of contents and elements in the scientific Web environment. *Journal of Information Science* 32, 4, 344–351.

Xiaoguang Qi and Brian D. Davison. 2009. Web page classification: Features and algorithms. *ACM Computing Surveys* 41, 2, Article No. 12.

Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. May 2000. New support vector algorithms. *Neural Computation* 12, 5, 1207–1245.

Qinfeng Shi, James Petterson, Gideon Dror, John Langford, Alex Smola, Alex Strehl, and Vishy Vishwanathan. 2009. Hash kernels. In *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*.

Lawrence K. Shih and David R. Karger. 2004. Using URLs and table layout for Web classification tasks. In *Proceedings of the 13th International Conference on World Wide Web (WWW'04)*. 193–202.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of the ICML Workshop on Learning with Multiple Views*.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*. 990–998.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY.

Yuxin Wang and Keizo Oyama. 2006. Web page classification exploiting contents of surrounding pages for building a high-quality homepage collection. In *Proceedings of the 9th International Conference on Asian Digital Libraries: Achievements, Challenges, and Opportunities (ICADL'06)*. 515–518.

Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. 2009. Feature hashing for large scale multitask learning. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML'09)*. ACM, New York, NY, 1113–1120.

Ian H. Witten, Eibe Frank, and Mark A. Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Morgan Kaufmann, San Francisco, CA.

Shuang-Hong Yang, Bo Long, Alexander J. Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: Joint friendship and interest propagation in social networks. In *Proceedings of the 20th International Conference on World Wide Web (WWW'11)*. 537–546.

Yiming Yang and Jan O. Pedersen. 1997. A comparative study on feature selection in text categorization. In *Proceedings of the 14th International Conference on Machine Learning (ICML'97)*. 412–420. http://dl.acm.org/citation.cfm?id=645526.657137.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics (ACL'95)*. 189–196.

Hwanjo Yu, Jiawei Han, and K. C.-C. Chang. 2004. PEBL: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering* 16, 1, 70–81.

Xiaojin Zhu. 2005. *Semi-Supervised Learning Literature Survey*. Technical Report. Computer Sciences, University of Wisconsin-Madison. http://www.cs.wisc.edu/∼.

Received January 2014; revised February 2015; accepted April 2015