Heterogeneous Learner for Web Page Classification

Hwanjo Yu, Kevin Chen-Chuan Chang, Jiawei Han
University of Illinois at Urbana-Champaign
Department of Computer Science
University of Illinois, Urbana-Champaign, IL, USA
{hwanjoyu, kcchang, hanj}@uiuc.edu

Abstract

Classification of an interesting class of Web pages (e.g., personal homepages, resume pages) has been an interesting problem. Typical machine learning algorithms for this problem require two classes of data for training: positive and negative training examples. However, in application to Web page classification, gathering an unbiased sample of negative examples appears to be difficult. We propose a heterogeneous learning framework for classifying Web pages, which (1) eliminates the need for negative training data, and (2) increases classification accuracy by using two heterogeneous learners. Our framework uses two heterogeneous learners – a decision list and a linear separator which complement each other – to eliminate the need for negative training data in the training phase and to increase the accuracy in the testing phase. Our results show that our heterogeneous framework achieves high accuracy without requiring negative training data; it enhances the accuracy of linear separators by reducing the errors on "low-margin data". That is, it classifies more accurately while requiring less human efforts in training.

1. Introduction

Automatic categorization or classification of Web pages has been studied extensively, and most of those classification techniques are usually based on similarity between documents' contents or their hyperlink structures. However, the categories generated by those techniques do not always fit end-users' search purposes since they cannot consider each user's specific interest. Let's think about a query "Find XML experts" on a common search engine. We may want to hit the keyword "XML" or "experts" on any search engine, and try to refine the search results repeatedly until we collect a fair amount of XML expert pages. However, if we are able to specify a search class or domain into "resume" or "personal homepage," we could simply apply a search term "XML" within the classes of resume or personal homepage to collect XML expert pages.

Automatic classification of specific types of documents such as newspaper articles, patent documents, calls for papers, and personal homepage have been proposed for this problem[7, 15]. However, these solutions have some limitations: they are quite dependent on specific classes, and they require laborious work to create a new classifier of interest. In particular, they require collecting positive training data and unbiased negative training data that uniformly represents the negative class. Finally, they show relatively poor performance particularly on classifying *low-margin data*. *Low-margin data* is the data that is relatively close to the separator, thus is often misclassified by linear separators. For example, if a personal homepage has not much personal information, it may be close to the separator of personal homepage class, and thus becomes a *low-margin data*. This problem is a well-known drawback of linear separators such as Winnow, Perceptron, and Perceptron-like algorithms [14, 18, 4, 6, 19, 12].

We present here a new machine learning framework that exactly matches these problems of Web page classification. Our framework uses two heterogeneous learners – a decision list and a linear separator which complement each other – in both training and testing phases. There have been many attempts to use multiple homogeneous learners to increase classification accuracy. However, combination of homogeneous learners generally does not overcome the genuine weaknesss of each learner. The purpose of the decision list in training phase is to eliminate the need for negative training data in constructing a linear separator. The decision list in testing phase enhances the accuracy of the linear separator especially for low-margin data. As a result, our heterogeneous framework (1) makes easier to create a classifier for a new concept by reducing the work to collect training documents, and also (2) increases the final classification accuracy by complementing the weakness of linear separators for low-margin data.

The contributions of our framework are the following.

- Our heterogeneous framework enables *pre-filtering* stage in training phase to induce negative training data from universe and positive training data. Previous machine learning schemes need to classify large number of pages manually to prepare *unbiased* positive and negative training documents. The pre-filtering stage makes possible to construct a classifier without requiring negative training data, which speeds up the process of creating a classifier for a new class, but also opens a possible way to support type-specific queries on the Internet from sample pages.

- We propose a new *early-inclusion* stage for correctly classifying low-margin data. Linear separators such as Winnow, Perceptron, and SVMs have been studied extensively and have proved their outstanding performances when the environment has high dimensions, the number of active features is small, and the instance spaces are sparse. Consequently, the linear separators are the most widely used algorithms for Web page classification problems since Web page classification has the same properties as environment these linear separators work well. However, they have showed weakness in classification of low-margin data [14, 18, 4, 6, 19, 12]. Our *early-inclusion* stage complements this weakness and achieves higher accuracy for low-margin data without sacrificing any performance on other data.

The rest of the paper is organized as follows. Section 2 describes the problem and our approach in detail. Section 3 presents the algorithm of each stage in the framework. In Section 4, we describe the experiment environment and results, and we evaluate the experiment results. Section 5 describes the related work. The conclusions and future works are discussed in Section 6.

2 Problem Description and Heterogeneous Framework

A typical algorithm for learning linear separators consists of two phases: training phase and testing phase (Figure 1). In training phase, the algorithm reads positive and negative training data to construct a linear separator (LS). The testing phase classifies testing documents by the LS constructed in the training phase. The typical learning framework suffers from the need of negative training data and the inherent inaccuracy for classifying low-margin data as we briefly mentioned in Section 1. To address these problems, we propose a heterogeneous framework built upon a linear separator (LS). Specifically, we introduce two stages: To eliminate the need for collecting class-specific negative data, our heterogeneous framework uses a *pre-filtering*

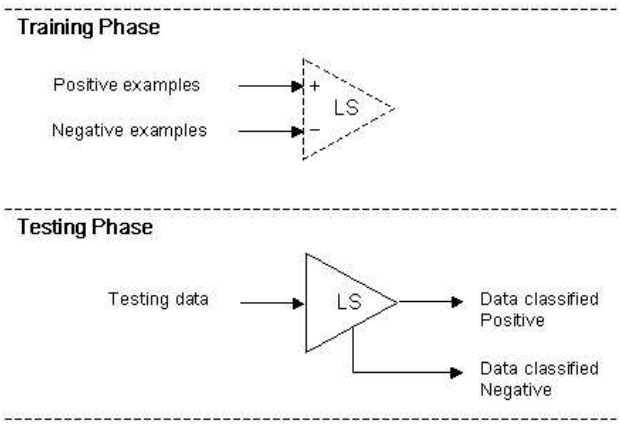

Figure 1. The typical learning framework

stage in the training phase. To improve classification accuracy on the low-margin data, the testing phase adopts an *early-inclusion* stage. The two stages use a decision list (DL) having different threshold parameter values. Figure 2 shows the linear separator (LS) and the decision list (DL) of the pre-filtering and early-inclusion stages in each phase of the framework. The DL of the pre-filtering and early-inclusion stages is constructed in the training phase I, and the LS is trained using pre-filtering DL in the training phase II. The early-inclusion DL and LS classify the testing data in the testing phase. In the following, we describe the problems and the approaches of each stage in details.

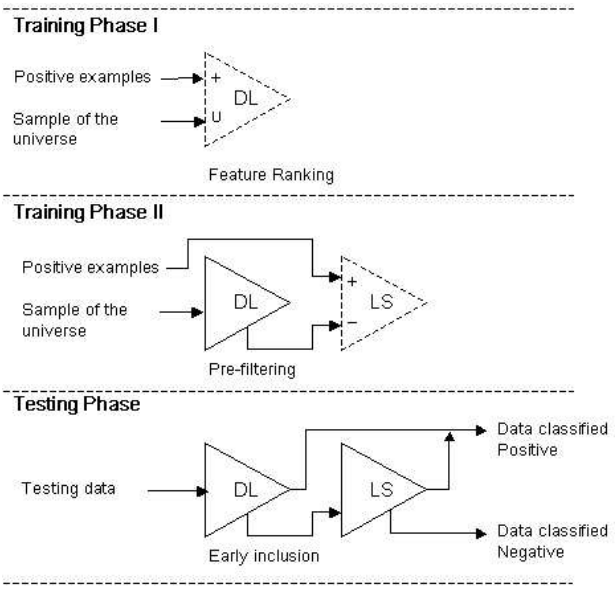

Figure 2. Our heterogeneous framework

2.1 Pre-filtering: Inducing Negative Training Data

An important issue to make a good linear separator is to prepare a good set of training documents that uniformly represent the concept without any bias. Since we are dealing with binary classification that separates a class of interest from the others, users would need to collect two sets of training data, the sample of class C and its complement $\neg C$. The portion of the class C in the universe ($P(C)$) is usually much smaller than the portion of its complement ($P(\neg C)$). For instance, "college admission pages" take very small percentages of the Internet. $P(C)$ may be less than 0.1%, and $P(\neg C)$ more than 99.9% in this case. Collecting negative training data $\neg C$ requires significant amount of manual classifications because the sample of negative data must represent the entire universe while excluding the instances of C.

One might argue that using a sample of universal set itself as a substitute for negative training data is valid since $P(C)$ is usually much smaller than $P(\neg C)$. However, a small number of false positives in the training data can impair the classification performance of linear separators, as our experiments show in Section 4.

Our heterogeneous framework does not require collecting $\neg C$ to construct a classifier. Instead, it uses a uniform sample of the universe, which can be collected automatically from a repository such as DMOZ[1]. Once the uniform sample is collected, it is reused for any classes, and thus making the training of a new classifier much easier. In our experiments, we used a random sampler to collect a uniform sample of the universe from DMOZ automatically. The pre-filtering stage in the training phase constructs an unbiased sample of negative training data from the sample of the universe and the positive training data. The unbiased negative training data is used then with the positive training data in order to learn a linear classifier. Our results show that the pre-filtering stage helps to achieve high accuracy without requiring negative training data.

2.2 Early-inclusion: Identifying Low-margin Data

The performance of linear separator such as Winnow, Perceptron, and Perceptron-like algorithms is dependent on the distance between the separator and example, usually referred to as "margin" [6]. In other words, most classification errors of those linear separators come from low-margin data, the data near the separator. The distance D between an example and the separator is formulated as

$$D = \frac{w \cdot h(d)}{||w|| ||h(d)||}, \quad (1)$$

where w is a weight vector of the linear function (i.e. $< w_1, w_2, w_3, ..., w_n >$), and $h(d)$ is a feature vector of an example d (i.e. $< h_1(d), h_2(d), h_3(d), ..., h_n(d) >$).

In order to identify the relationship between the distance D and the classification performance on real Web data, we randomly chose three common classes – "computer shopping," "sports shopping," and "stocks and bonds" – from DMOZ, and classified the pages of the classes using one of those linear separators (i.e. the Winnow algorithm). Figure 3 shows the distribution of examples at each distance from the separator in the feature space. For this figure, we used randomly selected 200 pages of each class to train each class, and used another 200 pages for testing. The figure shows that each class has fairly many pages near the separator and that most of false negatives come from examples near the separator. This observation shows that the inaccuracy of classifying low-margin data is a major problem for Web page classification.

The distance of an example from the separator is determined by the weight (w) of every active feature ($h_i(d)$) of the example as shown in the formula of D. The linear separators such as Winnow, Perceptron, or Perceptron-like algorithms determine the weight of each feature according to its likelihood to appear in the training data. Therefore, an example would be located near the separator if the example has a small number of *discriminative* positive features that may not occur frequently in the positive class, and if the example also has many non-positive features. For instance, if a personal homepage has the title "personal homepage", it is a personal homepage even if it does not have any personal information in it. That is, "personal homepage" in title is a discriminative feature for the personal homepage class although it does not occur frequently in the class.

Our early-inclusion stage helps to identify those misclassified low-margin data. As a result, this stage reduces the total classification error rate. The early-inclusion stage identifies the small number of discriminative positive features from positive training data and the sample of the universe, and uses these features to label obvious positive pages before sending the documents to the next stage of linear separator function as Figure 2 shows.

Our results show that the accuracy of classification increases for all test classes we considered (i.e. personal homepages, resume pages, and college admission pages) by adding the early-inclusion stage, which implies the early-inclusion stage actually identifies the discriminative features that may not get weights high enough to be classified correctly by the linear separators.

3 Algorithms

This section explains the algorithms of each stage in the framework. First, we discuss the linear separator algorithms

[1] http://dmoz.org

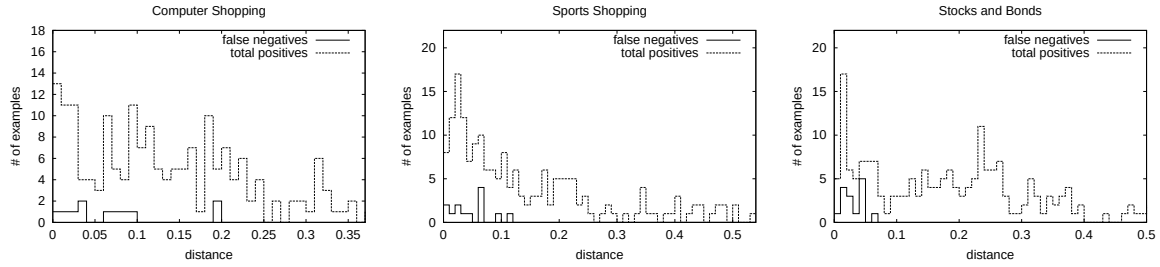

Figure 3. The distribution of misclassified examples according to their distance from the separator (The distribution of false positives is also similar, but we only show false negatives because the *early-inclusion* deals with only false negatives. See Section 4 for details)

that can be plugged into this framework. After that, we describe the feature ranking algorithm that we use for the pre-filtering and early-inclusion stages, and explain how each stage uses this feature ranking algorithm.

3.1 Linear separators

Different kinds of linear separators can be used in our framework. In comparison, Winnow performs better than Perceptron when the number of relevant features is small. Perceptron performs better than Winnow when the instance spaces are sparse [11]. We use Winnow [14] and SVM (Support Vector Machine) [4] in our experiments because both algorithms have shown good performance for Web page classification and are among the most widely used for this purpose.

3.2 Feature ranking

Both the pre-filtering and early-inclusion stages use a feature ranking algorithm that ranks only positive features according to their frequencies of occurrence in the positive example and the universe. We call *positive features* the features that are active in positive examples. Among such features we distinguish *strong positive features* that are very likely to appear in a positive example. We call *strong positive* the examples that have one or more active strong positive features. In the following we will develop a quantitative model describing these concepts.

We would like to measure the strength of a positive feature. For this purpose, we use weights that capture the relative frequencies of the corresponding feature in the sample of a particular class s: $f_s(x) = n_s(x)/N_s$, where $n_s(x)$ is the number of pages in the sample having feature x and N_s is the total number of pages in the sample. We calculate the weight of feature x as follows:

$$w(x) = \frac{f_+(x)}{f_U(x)} * (f_+(x) - f_U(x)), \qquad (2)$$

where $f_+(x)$: the relative frequency of feature x in the sample of positive class, and $f_U(x)$: the relative frequency of feature x in the sample of the universe.

Note we do not use entropy loss to rank features because it is hard to estimate the probabilities of positive and negative examples, or P_+ and P_- respectively. In this case, a small error in estimating the probability of each class can make a large difference in ranking result when P_+ is extremely small compared to P_- (as we discussed in the previous section).

Our weighting formula considers the rate ($\frac{f_+}{f_U}$) and difference ($f_+ - f_U$) of each feature's frequency in the positive class and the universe. This formula has shown advantages among several other weighting formulas that were explored in our experiments (including TFIDF weights).

We will use this weighting function in constructing the pre-filtering algorithm which uses strong features to approximate negative training data.

3.3 Pre-filtering

The pre-filtering stage uses strong positive features to filter out strong positive data from the sample of the universe, and thus induces a good approximation of the negative training data. Figure 4 shows the pre-filtering algorithm.

We first extract positive features from T_+ and T_U, and rank them by the weighting formula in Equation (2). (Steps 1 through 3 of the algorithm in Figure 4). We take out a chunk of top ranked features by applying the following threshold function θ onto the list (set) of positive features (Steps 4 and 5 of the algorithm in Figure 4).

$$\theta = max_{x \in t}(\gamma(x)), \qquad (3)$$

where $t = \{\ x \mid x\ is\ the\ strongest\ positive\ feature\ in\ some\ document\ d,\ \forall d \in T_+\}$.

The threshold θ is the worst ranking among the strongest positive features in each document. For example, say that we have only three positive documents d_1, d_2, and d_3, and

Input: T_+ a sample of the positive class, T_U a sample of the universe
Output: T_- a sample of the negative class needed for training the linear separator stage
Notation: A document d denotes a set of its active features. ($x \in d$ means a feature x is present in document d. $x \in T_+$ implies $x \in d, \forall d \in T_+$)
Algorithm:

1. Extract all positive features R from T_+ ($R = \cup_{x \in T_+} x$)
2. Rank the features in R based on the weighting function $w(x)$ ($w(x) = \frac{f_+(x)}{f_U(x)} * (f_+(x) - f_U(x))$)
3. Let $\gamma(x)$ returns the ranking of feature x (a smaller number indicates higher ranking.)
4. Compute a threshold θ with the following formula
   $\theta = max_{x \in t}(\gamma(x))$,
   $t = \{\, x \mid x\ is\ the\ strongest\ positive\ feature\ in\ some\ document\ d,\ \forall d \in T_+ \}$.
5. Extract strong positive features S by applying θ to R ($S = \{\, x \in R \mid \gamma(x) < \theta \,\}$)
6. Extract negative documents T_- ($T_- = \{\, d \in T_U \mid d \cap S = \emptyset \,\}$)

Figure 4. The pre-filtering algorithm

the global rankings $\gamma(x)$ of the strongest positive feature in each document are 11, 14, and 7 respectively. (That is, feature x with $\gamma(x) = 11$ is the strongest positive feature in document d_1.) Then we set the threshold θ to 14, which is the lowest ranking of them. This heuristic for the threshold θ fairly reasonably approximates negative training examples which does not significantly impact classification performance.

The negative training data T_- is constructed by excluding the documents having any of those strong features S from the sample of the universe T_U. We assume that the document d is positive if the intersection $d \cap S$ not empty. We exclude such documents from T_U to obtain an induced sample of the negative class.

3.4 Early-inclusion

In early-inclusion stage, we scan test documents to see if they have any discriminative features. If they have, we label them as positives, and exclude them from the next stage. We will call a feature x as discriminative if $\gamma(x) < \theta'$ (i.e. the feature x is ranked higher than θ') for some threshold γ'. We apply a linear function computed in the training phase only to the rest of the test documents which are not labeled in the early-inclusion stage. We induce the discriminative positive feature set by applying another smaller threshold θ' to the same feature ranking function γ. Note that θ' is smaller than θ because the discriminative positive feature set needs stronger requirement of positivity.

Thus, the set of discriminative positive features S' is

$$S' = \{x \in R \mid \gamma(x) < \theta' \} \quad (4)$$

Documents from the testing set T having at least one of active features from S' are labeled positive such that

$$T'_+ = \{\, d \in T \mid d \cap S' \neq \emptyset \,\} \quad (5)$$

The documents that are not labeled in the early-inclusion stage are routed into the linear separator.

The threshold θ' can be adjusted through a validation process, such that it is set as high as possible under the condition that the early-inclusion (T'_+) does not introduce false positive. Our experiment (Table 1 in Section 4) shows that the early-inclusion stage does not increases the number of false positives while it does decrease the number of false negatives.

This two-stage classification can be viewed as a serial connection of a simple decision list and a linear separator because the early-inclusion is essentially a decision list using discriminative features. Our experiments described in Section 4 show that this combination of two classifiers complements the weakness of each other.

4 Experiments

4.1 Experimental Methodology

To test our framework, we chose three classes: personal homepages, college admission pages, and resume pages. For each class, we collected 233, 97, and 100 positive training documents respectively. We randomly selected 2514 pages from DMOZ to construct a uniform sample of the universe. For each class, we tested around 600 pages that are not duplicated with the training data. In reality, personal homepages are extremely diverse, so we confined the personal homepage class to pages having personal information. Pages having only images or extremely small amount of text were removed from the experiments.

We extracted features from different parts of a page—URL, title, headings, link, anchor-text, normal text, and meta tags. Each feature is a predicate indicating whether each term or special character appears in each part, e.g., '~' in URL, or a word 'homepage' in title. We did not use stemming or a stoplist because it could hurt performance in

Web page classification. For example, a common stopword, "I" or "my", is a good indicator of a student homepage.

As we mentioned in previous section, we performed the experiments using two different linear separators, Winnow [14] and SVM (Support Vector Machine) with linear kernel function [4]. We use SNOW (Sparse Network Of Winnow) [2] for Winnow implementation, and SVM LIGHT [9] for SVM implementation.

4.2 Results

We first show the performance comparison on the three classes between the typical framework and our heterogeneous framework. After that, we discuss the concrete impacts of early-inclusion stage on performance.

4.2.1 Performance comparison

Table 1 shows the classification error rates of Winnow and SVM respectively on the three classes "resume", "college admission", and "personal homepage." Typical learning framework that uses positive training data and samples of universe as negative training data shows high rates of false negatives. This problem is the consequence of placing too deep in the positive class. Using pre-filtering stage, the false negatives decrease substantially without much sacrificing the performance on negative testing documents. The early-inclusion stage decreases even more the classification errors on the positive examples while keeping the classification errors on the negative examples from increasing (Table 1).

4.2.2 Impact of the early-inclusion stage on performance and data margin

The early-inclusion stage uses a decision list. Figure 5 shows the performance gain by serializing the early-inclusion stage and the linear classification stage for personal homepage classification. The first graph in Figure 5 shows the number of false negatives at each distance when using only a linear separator (i.e. Winnow using the SNOW [2]). In this graph, most errors of the linear separator come from low-margin data. The second graph in the figure shows the classification errors when using only a decision list of the early-inclusion stage. The classification errors of the second graph are fairly spreaded over every distance compared to the linear separator graph. The last graph of the figure 5 shows the reduced classification errors using the two stages connected serially. In this case, the decision list of early-inclusion stage complements the linear separator's shortcoming by reducing the errors on low-margin data, which results in higher overall accuracy. Figures 6 and 7 show the same results when classifying resume pages and college admission pages respectively.

4.3 Discussion

When we classify Web pages manually, we may consider many factors such as title, text, url, and so on as the typical machine learning algorithms do. However, a small number of discriminative positive features could determine the class of a page regardless of the other factors. (e.g. if we see "personal homepage" on title, it is likely to be a personal homepage no matter what content it has.) A learning algorithm such as linear separators (i.e. Winnow, Perceptron, or Perceptron-like algorithms) may not weigh the discriminative features high enough. This problem can occur if the features occur infrequently. In this case, the linear separator can misclassify a page if the page has much non-relevant information. Usually, the distance of those pages are close to the separator as we discussed in Section 2.

Our framework identifies discriminative features using the feature ranking algorithm we explained in Section 3. The early-inclusion stage uses discriminative features to include low-margin positive data in the positive class, preventing misclassification of this data by a linear separator. Thus, the final classification accuracy increases.

5 Related Work

The research community has been extensively working on text classification problems, and there have been many attempts to extend this work to Web page classification [5, 17, 1, 3]. Most of those classification techniques are usually based on similarity of documents or their hyperlink structures. They usually require pre-defined categories that cover the entire target domain and that each class is assumed to be mutually exclusive.

Matsuda and Fukushima identified Web page type classification problem, and tried to solve it by human description of structural characteristics of a type [16]. Yi and Sundaresan used Naive bayes method with structured vector model [21]. It performs well on structured or tightly formatted semi-structured documents such as US Patent data (XML sharing same DTD) or Resumes (tightly formatted documents), but those structured vector model is not likely to perform well for classification of normal HTML documents having various formats. Glover et al. classified "personal homepage" and "call-for-paper." [7] They classified manually a large amount of documents to collect negative training data and used SVM with fixed set of features. Similar methods were used for classification of newspaper articles [8] and patent documents [15]. Most of these works are quite dependent on specific classes, and they require much efforts to create a new classifier for a new class of interest.

There are some recent papers on multistage classification [10, 20, 13] that use the arrangement of homogeneous classifying elements to achieve better quality of classification.

	Winnow						SVM					
	T		T+P		T+P+E		T		T+P		T+P+E	
	f-	f+	f-	f+	f-	f+	f-	f+	f-	f+	f-	f+
Resume	6.38	2.06	4.26	0.19	1.06	0.19	3.19	0.56	3.19	0.56	1.06	0.56
Admission	8.25	0.00	2.06	0.89	1.03	0.89	3.09	0.00	3.09	0.89	2.06	0.89
Homepage	27.67	0.22	6.29	3.56	3.50	3.56	13.29	0.45	6.99	2.67	5.60	2.67

T: typical learning framework using positive training data and the universe as a substitute for the negative training data. T+P: typical learning framework + pre-filtering stage. T+P+E: typical learning framework + pre-filtering stage + early-inclusion stage. f-: % of false negatives. f+: % of false positives.

Table 1. Error rate (%) comparison of each framework using Winnow and SVM

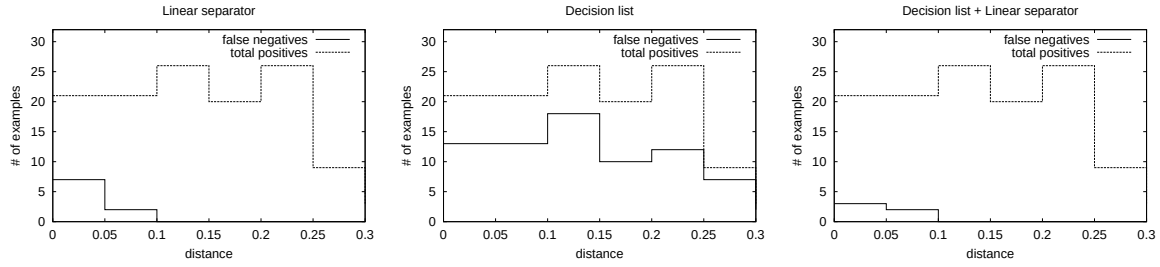

Figure 5. Advantage of the two heterogeneous stages for personal homepage classification

However, like typical learning algorithms, they also require the positive and negative training sets.

H. Yu et al. [22] also propose a scheme to remove the requirement of negative training pages in Web page classification. The algorithm uses a rule-based algorithm in the first stage to map an initial boundary from positive and unlabeled data. After that, it iterates SVMs to induce an accurate class boundary from the initial boundary. The marginal property of SVM guarantees the convergence of the boundary.

Our framework also uses heterogeneous stages. However, we identify another intrinsic problem of Web page classification – high error on low-margin data. By using the two heterogeneous learners in both training and testing phases, we obviate the need for negative training data, while at the same time reduce the errors on low-margin data.

6 Conclusions

In this paper, we propose a heterogeneous learning framework for classifying Web pages, which (1) eliminates the need for negative training data, and (2) increases classification accuracy by using two heterogeneous learners. Our framework uses two heterogeneous learners – a decision list and a linear separator which complement each other – to eliminate the need for negative training data in the training phase and to increase the accuracy in the testing phase. Our results show that our heterogeneous framework achieves high accuracy without requiring negative training data; it enhances the accuracy of linear separators by reducing the errors on "low-margin data". That is, it classifies more accurately while requiring less human efforts in training.

References

[1] D. Boley, M. Gini, R. Gross, E.-H. S. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems*, 1999.

[2] A. J. Carlson, C. M. Cumby, J. L. Rosen, and D. Roth. *SNoW User Guide*. Cognitive Computation Group, Computer Science Department, University of Illinois at Urbana-Champaign, August 1999.

[3] H. Chen, C. Schuffels, and R. Orwig. Internet categorization and search: A self-organizing approach. *Journal of Visual Communication and Image Representation*, 7(1):88–102, 1996.

[4] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, (20):273–297, 1995.

[5] M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1/2):97–119, 2001.

[6] Y. Freund and R. E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 5(14):771–780, 1999.

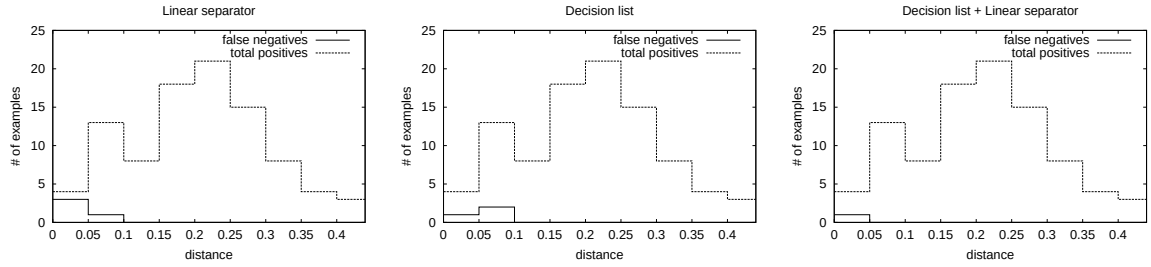

Figure 6. Advantage of the two heterogeneous stages for resume page classification

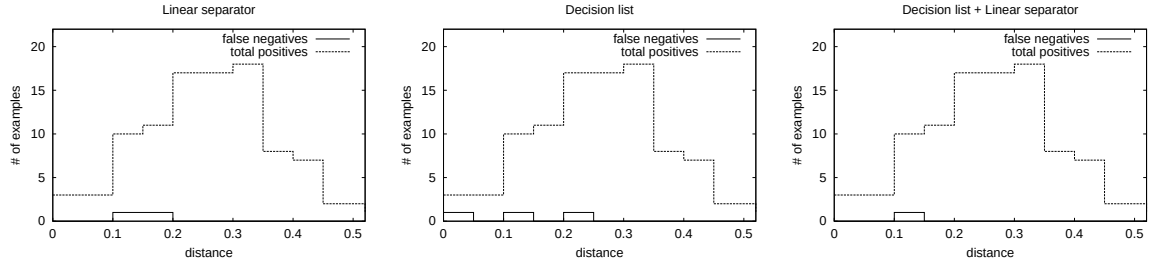

Figure 7. Advantage of the two heterogeneous stage for college admission page classification

[7] E. J. Glover, G. W. F. S. Lawrence, W. P. Birmingham, A. Kruger, C. L. Giles, and D. M. Pennock. Improving category specifi c web search by learning query modifi cations. In *Symposium on Applications and the Internet, SAINT 2001*, San Diego, California, January 8-12, January 8-12 2001.

[8] J. Hayes and W. S. P. A system for content-based indexing of a database of news stories. In *Proceedings of Second Annual Conference on Innovatative Applications of Artifi cial Intelligence*, pages 1–5, 1990.

[9] T. Joachims. Making large-scale svm learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[10] C. Kaynak and E. Alpaydin. Multistage cascading of multiple classifi ers: One man's noise is another man's data. In *ICML*, 2000.

[11] J. Kivinen, M. K. Warmuth, and P. Auer. The Perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bound when few input variables are relevant. *Artifi cial Intelligence*, 1-2:325–343, 1997.

[12] A. R. Klivans and R. A. Servedio. Learning dnf in time $2^{O(n^{1/3})}$. citeseer.nj.nec.com/329971.html.

[13] A. Kosorukoff. Genetic synthesis of cascade structures for particle classifi cation. In D. Whitley, editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 170–174, Las Vegas, Nevada, USA, 8 2000.

[14] N. Littlestone. Learning quickly when irrelevant attributes abound. a new linear-threshold algorithm. *Machine Learning*, (2):285–318, 1988.

[15] H. Mase, H. Tsuji, H. Kinukawa, Y. Hosoya, K. Koutani, and K. Kiyota. Experimental simulation for automatic patent categorization. In *Proceedings of Advances in Production Management Systems*, pages 377–382, 1996.

[16] K. Matsuda and T. Fukushima. Task-oriented world wide web retrieval by document type classifi cation. In *CIKM '99*, Kansas City, Mo, USA.

[17] H.-J. Oh, S. H. Myaeng, and M.-H. Lee. A practical hypertext categorization method using links and incrementally available class information. In *SIGIR 2000*, Athens, Greece, 2000.

[18] F. Rosenblatt. The perceptron: A probabilistic for information storage and organization in the brain. *Psychological Review*, (65):386–407, 1958. Reprinted in Neurocomputing (MIT Press, 1988).

[19] R. A. Servedio. On pac learning using winnow, perceptron, and a perceptron-like algorithm. citeseer.nj.nec.com/329971.html.

[20] J. Smith and S. Chang. Multi-stage classifi cation of images from features and related text. In *EDLOS Workshop*, San Miniato, Italy, 1997.

[21] J. Yi and N. Sundaresan. A classifi er for semi-structured documents. In *KDD 2000*, Boston, MA USA, 2000.

[22] H. Yu, J. Han, and K. C.-C. Chang. Pebl: Positive-example based learning for web page classifi cation using svm. In *KDD*, Edmonton, Alberta, Canada, 2002.