FloatCascade Learning for Fast Imbalanced Web Mining

Xiaoxun Zhang[1], Xueying Wang[2*], Honglei Guo[1], Zhili Guo[1], Xian Wu[1] and Zhong Su[1]

[1] IBM China Research Lab
Beijing, 100094, China
{zhangxx, guohl, guozhili, wuxian, suzhong}@cn.ibm.com

[2] Peking University
Beijing, 100871, China
{wangxy05}@sei.pku.edu.cn

ABSTRACT

This paper is concerned with the problem of Imbalanced Classification (IC) in web mining, which often arises on the web due to the "Matthew Effect". As web IC applications usually need to provide online service for user and deal with large volume of data, classification speed emerges as an important issue to be addressed. In face detection, Asymmetric Cascade is used to speed up imbalanced classification by building a cascade structure of simple classifiers, but it often causes a loss of classification accuracy due to the iterative feature addition in its learning procedure. In this paper, we adopt the idea of cascade classifier in imbalanced web mining for fast classification and propose a novel asymmetric cascade learning method called FloatCascade to improve the accuracy. To the end, FloatCascade selects fewer yet more effective features at each stage of the cascade classifier. In addition, a decision-tree scheme is adopted to enhance feature diversity and discrimination capability for FloatCascade learning. We evaluate FloatCascade through two typical IC applications in web mining: web page categorization and citation matching. Experimental results demonstrate the effectiveness and efficiency of FloatCascade comparing to the state-of-the-art IC methods like Asymmetric Cascade, Asymmetric AdaBoost and Weighted SVM.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *Data Mining*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms: Algorithms, Experimentation

Keywords: Fast imbalanced classification, Float Searching, Cascade learning, Web page categorization, Citation matching

1. INTRODUCTION

In this paper, we are concerned with the problem of Imbalanced Classification (IC) in web mining, which often arises on the web due to the "Matthew Effect": the rich get richer and the poor get poorer. As a result, the positive and negative examples exhibits distinct imbalance in web IC applications, i.e., the number of positive examples is far smaller than that of negative examples. One typical example is web directory (see Figure 1) where huge web pages are manually organized into hierarchical categories, such as ODP (Open Directory Project) [38], Yahoo! Directory [39] and Google Directory [40]. It has been observed that the category distribution is highly unbalanced on these popular web directories. For instance, on ODP directory, the maximum second-level category ("Society/religion and spirituality") contains about 80,000 pages, whereas some minimum second-level categories contain only one page ("Home/News and media", "News/chats and forums"). Another prominent example is citation matching (see Figure 2), which aims to identify whether two citations actually refer to the same publication [24, 34, 35]. It is a crucial step for online paper services such as Google Scholar [41] and CiteSeer [42]. Among a large amount of candidate pairs of citations, only a very few pairs are target co-references.

Figure 1. The example of web directory: ODP.

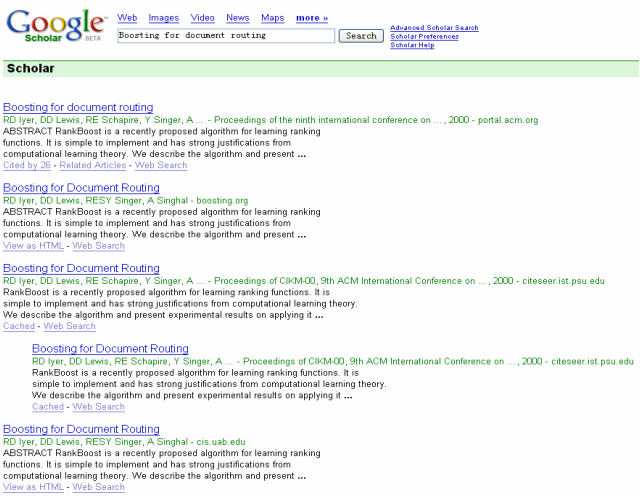

Figure 2. The example of citation matching: Google Scholar.

---

* This work is conducted while the author was on an internship at IBM China Research Lab.

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

Previous work has shown that imbalanced data would adversely affect classification accuracy. Classifiers without consideration of imbalance tend to be overwhelmed by major negative examples [1, 2]. Some IC approaches have been proposed in the fields of text mining and pattern classification, mainly including discarding negative examples [3, 4, 5], synthesizing positive examples [18] and assigning greater costs on positive examples than negative ones [6, 20, 21, 22, 23]. These methods primarily focused on improving the classification accuracy with particular concern on minor positive examples [16]. However, classification speed becomes an important issue to be addressed in many cases, especially for those web IC applications which need to deal with large volume of data and those online web services which require rapid response to the user request.

In real-time face detection, the classifier is required to locate the very few faces quickly and accurately among millions of image regions [9]. Asymmetric Cascade (AsyCascade) can greatly speed up face detection by building a cascade structure of simple classifiers [10]. It uses a small number of features in the early stages to exclude the large majority of non-face regions. Complex computation is only reserved for the small number of face-like regions. As a result, the classification speed is significantly raised. Unfortunately, AsyCascade usually achieves fast classification at the expense of classification accuracy. This is mainly due to the iterative feature addition in its learning process. Classifiers with more features run the risk of poorer generalization and more computation time.

In this paper, we adopt the idea of cascade classifier in imbalanced web mining for fast classification, and propose a new asymmetric cascade learning method called FloatCascade to improve the accuracy. Compared with AsyCascade, FloatCascade can select fewer but more effective features at each stage of the cascade classifier. It uses a float searching scheme [30] to remove and/or replace features that cause higher false positive rates. The quantity and quality of available features become the key factors to the success of FloatCascade. A decision-tree scheme is hereby adopted to enhance feature diversity and discrimination capability. We evaluate FloatCascade on two typical web IC applications: web page categorization and citation matching. Experimental results demonstrate that: 1) the classification time of FloatCascade is further reduced compared with AsyCascade because fewer features are required for classification; 2) FloatCascade is consistently superior to AsyCascade and even better than non-cascade asymmetric methods like Asymmetric AdaBoost [9] and Weighted SVM [6, 23] because more effective features are found for classification.

The remainder of the paper is organized as follows. Related work is reviewed in Section 2. Section 3 presents our FloatCascade learning. Experiments, evaluation and analysis are conducted in Section 4. Finally, the conclusion is given in Section 5.

2. RELATED WORK

2.1 Imbalanced Classification Problem

The IC problem has attracted considerable attention in the fields of text mining and pattern classification, e.g., text categorization [6, 17, 23, 25], reference matching [24, 31], spam detection [27], duplicate detection [32], medical diagnosis [27] and oil spills detection [33]. Existing IC approaches can be broadly divided into two categories: re-sampling and re-weighting methods. Re-sampling methods artificially balance the two classes by over-sampling positive examples or down-sampling negatives ones [1, 2, 3, 4, 5]. Re-weighting methods are also known as cost-sensitive methods [6, 20, 21, 22, 23], where positive examples are assigned greater costs than negative ones. However, until now, the IC problem has not received much attention in the community of web mining, though it occurs very often on the web.

Since manual categorization is prohibitively time-consuming and labor-expensive for web-scale applications, there has been much work on automatic web page categorization. Generally, there are two major kinds of automatic categorization approaches [11, 12, 13]: content-based and context-based methods. Content-based methods build the classifier using words or phrases in web pages. Naive Bayes (NB) [11, 15] and Support Vector Machine (SVM) [11, 14] have been approved to be effective methods along this line. Context-based methods additionally exploit hyperlink and hypertext among web pages [12, 13]. However, most of these approaches ignore imbalanced distribution of web categories, which results in adverse classification accuracy [1]. Some works on text categorization have noticed this problem and attempted to improve the accuracy by exploring imbalanced text distribution, e.g., SVM on re-sampled data [17, 25] and Weighted SVM with asymmetric cost [6, 23]. Though these works might achieve better classification accuracy, they fundamentally neglect the important issue of classification efficiency, which makes them inapplicable to the web-scale applications. By contrast, FloatCascade considers both classification effectiveness and classification efficiency by taking advantage of the inherent imbalance of web categories.

Citation matching aims to identify whether two citations actually refer to the same publication [31]. It is not a trivial problem since various data inconsistencies may occur between citations, e.g., name abbreviation, incorrect spellings, different formatting and citation mistakes. Citation matching is a crucial problem for paper search engines, where fast classification is highly desirable for online response to user queries. An efficient two-stage method called Canopy was proposed for citation matching in [24]. It first places the citations which are potential co-references into the same cluster using a rough metric, and then conduct complex computation in each cluster using a rigorous metric. In some sense, Canopy can be regarded as a simplified two-stage AsyCascade classifier, but AsyCascade differs from it in two essential aspects: 1) the two metrics used in Canopy are manually determined while all the features used in AsyCascade are automatically selected; 2) Canopy reduces the classification time by excluding the citation pairs between different clusters while AsyCascade achieves fast classification by quickly discarding the majority of negative examples in early stages.

2.2 Asymmetric Cascade

Simple classifier at each stage of AsyCascade is trained using Asymmetric AdaBoost (AsyBoost) whose learning objective is to reserve as many positive examples. AsyBoost is an extension of AdaBoost [9] which combines multiple weak classifiers to form a strong ensemble classifier [8]. Mostly, a weak classifier is just a single feature. AdaBoost provides an effective feature selection mechanism by iteratively re-weighting the training examples. It selects the feature with the lowest weighted error at each round and adds it to the ensemble classifier. As re-weighting proceeds, the weights of correctly classified examples are decreased while the weights of misclassified ones are increased. This forces the subsequent weak classifiers to gradually focus on hard examples. AsyBoost [9] further assigns greater costs to false negatives than false positives by up-weighting the positive examples. This forces

the subsequent weak classifiers to asymmetrically focus on positive examples. As a result, AsyBoost can effectively reduce the misclassification of positive examples.

The main learning objective of AsyCascade is to achieve radically reduced classification time as well as increased detection rate. Toward this end, AsyCascade leverages the inherent imbalanced data distribution from two aspects: 1) **Re-weighting**. Each simple classifier is trained using AsyBoost for reserving more positive examples. The process of re-weighting all examples is repeated once adding a new feature to the ensemble classifier. 2) **Re-sampling**. Each stage is trained on all positive examples and on a subset of negative examples. The process of re-sampling negative examples is repeated once adding a new stage to the cascade classifier. In moving from the previous stage to the next one, correctly classified negative examples are replaced with the unused ones while misclassified negative examples are retained. This poses a more challenging classification task for the next stage, and thus a more complex classifier is usually learned. As seen, AsyCascade combines the advantages of the re-sampling and re-weighting techniques to achieve fast classification. It has been improved by some subsequent works in two ways. The first way attempts to improve AsyBoost using a better re-weighting scheme [26, 27] while the second one tries to build a better cascade classifier [28, 29]. Unfortunately, all of these works achieved fast classification at the cost of decreased accuracy. As compared, FloatCascade can achieve better classification accuracy as well as higher classification speed simultaneously.

3. FLOATCASCADE LEARNING

In this paragraph, we first highlight the learning objective of FloatCascade in Section 3.1. Then, we present FloatCascade learning from its training and testing in Section 3.2. After that, in Section 3.3, we build the decision-tree feature for FloatCascade learning. Finally, in Section 3.4, we discuss some related issues.

3.1 Learning Objective

Web data often manifests two distinct characteristics with respect to data distribution: 1) there are a large amount of examples; 2) there are a small number of positive examples in comparison with a large quantity of negative ones. Such web data poses some great challenges for learning a classifier, including

- ***Balanced classification accuracy***: AsyCascade hopes to detect as many positive examples in its learning. Formally, the false negative rate (fn) at its each stage should be lower than a predefined threshold ϑ, i.e.,

$$fn \le \vartheta$$

  However, a classifier with enough low fn is still useless in real-world applications if it has too high false positive rate (fp). In an extreme case, all the examples can be simply classified as positive examples. In this way, all the positive examples are reserved, but all the negative examples are misclassified. In order to control the misclassification of negative examples, FloatCascade wants to locate the minimum fp on condition that fn is not greater than ϑ. Specifically, the learning objective at each stage of FloatCascade learning can be formulated as

$$\min(fp) \quad s.t. \quad fn \le \vartheta$$

  As a consequence, FloatCascade can achieve not only very low false negative rate to avoid missing minor positive examples but also very low false positive rate to avoid introducing too many negative examples. The overall classification accuracy is well balanced and favorably raised.

- ***Fast classification speed***: fast classification is required to deal with the huge number of examples. FloatCascade achieves this goal by exploiting the natural unbalanced distribution of the positive and negative examples. It hopes to find more effective features to quickly exclude more obvious negative examples with simple computation. In the mean time, it hopes to use fewer features at each stage of the cascade classifier, which further reduces the computation time in classification.

3.2 Float Searching for Cascade Learning

3.2.1 Motivation

The training procedure of AsyCascade is automatic and adaptive, which is conducted by the predefined accuracy requirement of the ensemble classifier at each stage [10]. It requires each ensemble classifier to achieve very high detection rate (which is near 100%) and only modest false positive rate (which is not over 50%). In this paper, detection rate refers to false negative rate and recall. The detection rate is usually satisfied by lowering the threshold of the ensemble classifier. However, a lower threshold would yield a classifier with more false positives. It has to add more features to reduce the false positive rate. As a result, it will introduce some unnecessary and even deleterious features. The building of AsyCascade classifier is just such a stage-wise process adding features in greedy manner. Unfortunately, two problems may arise in the classifier dependent on more features: 1) the classifier becomes less efficient since more computation time is needed to classify an example; 2) the classifier becomes less effective since over-fitting problem is more likely to occur in a complex classifier.

To attack these problems, we propose a new asymmetric cascade learning method called FloatCascade. It successfully uses less but more effective features at each stage by adopting a float searching scheme [33]. Float searching is an efficient feature selection method with a backtrack mechanism. Backtracks are performed to remove some previous features that cause accuracy drops when a new feature is added. FloatCascade adopts the detection rate and false positive rate to conduct such backtrack process. In specific, it removes the features causing higher false positive rates given that the required detection rate is attained. Next, we introduce FloatCascade learning in details from its training and testing procedures respectively.

3.2.2 Training Procedure of FloatCascade

Give a training set $Z = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ where $n = a + b$ of which a examples have $y_i = +1$ and b examples have $y_i = -1$. In web IC problems, $a << b$.

The training process of FloatCascade is described in Figure 3. In the figure, F_{target} denotes the target false positive rate for the overall cascade classifier; ϑ is the acceptable minimum false negative rate and f is the acceptable maximum false positive rate for each stage; P is the set of training positive examples, V is the set of training positive examples and N is a subset of training negative examples by re-sampling all the training negative examples with the

maximum number $b_{\max}$; $M_{\max}$ is the maximum number of features at each stage. Asymmetrically, false negatives cost k times more than false positives. H_M denotes the ensemble classifier of M *th* stage in the cascade classifier, which linearly combines multiple weak classifiers $\{h_i\}$.

Cascade Learning for adding a new stage
While $F_i > F_{target}$
1 $i \leftarrow i+1; M = 0; F_i = F_{i-1}$
2 Initialization for stage learning
  2.1 $w_i^{(0)} = \frac{1}{2a}$ for positive examples in P and
    $w_i^{(0)} = \frac{1}{2b_{\max}}$ for negative examples in N
  2.2 $f_i^{\min} = f$, where $i = 1,\ldots,M_{\max}$
3 **Stage Learning** for selecting a new feature
  While $F_i > f$
    3.1 $M \leftarrow M+1$
    3.2 Adding features by Asymmetric AdaBoost
      3.2.1 Select h_M (a decision-tree feature) based on P and N; $H_M = H_{M-1} \cup \{h_M\}$
      3.2.2 Decrease threshold for a false negative rate of at most ϑ on V
      3.2.3 Evaluate the false positive rate f_{H_M} of H_M on V
      3.2.4 If $f_{H_M} < f_M^{\min}$, then $f_M^{\min} = f_{H_M}$
    3.3 Deleting features by Float Searching
      3.3.1 $h' = \arg\min_{h \in H_M} f(H_M - h)$
      3.3.2 If $f_{H_M - h'} < f_{M-1}^{\min}$ then
        3.3.2.1 $H_{M-1} = H_M - h'$; $f_{M-1}^{\min} = f_{H_M - h'}$; $M = M-1$
        3.3.2.2 go to 3.3.1
    3.4 $F_i = f_M^{\min}$
    3.5 **Re-weighting** all the examples
      3.5.1 Update all the weights of the examples
        $w_i^{(M)} \leftarrow \exp(-y_i H_M(x_i))\exp(y_i \log\sqrt{k})$
      3.5.2 Normalize to $\sum_i w_i^{(M)} = 1$
4 **Re-sampling** on the negative examples
  4.1 $N \leftarrow \phi$
  4.2 if $F_i > F_{target}$, then evaluate the current cascaded classifier on the set of negative training examples put any false positive into the set N

Figure 3. The training process of FloatCascade.

As described, FloatCascade includes two-level learning: **cascade learning** and **stage learning**. Cascade learning adds a new stage to the cascade classifier if F_{target} is not achieved. It mainly includes stage learning in Step 3 and re-sampling in Step 4. The essential difference between FloatCascade and AsyCascade just lies in stage learning. Stage learning in AsyCascade continuously adds new features to the ensemble classifier if f is not satisfied. Differently, FloatCascade adopts a backtrack method to remove some pervious features for the minimum false positive rate when ϑ is satisfied. Feature addition in Step 3.1 is accomplished by AsyBoost assigning greater costs to positive examples, and feature deletion in Step 3.2 is accomplished by float searching removing features that cause higher false positive rates. It should be noted that, for the purpose of enhancing the generalization capability of the learned classifier, the accuracy of each ensemble classifier is evaluated on the validation set V rather than the training set itself in Step 3.2.2.

3.2.3 Testing Procedure of FloatCascade

The testing process of FloatCascade is depicted in Figure 4. When an example is input, a negative decision made at any stage leads to the immediate rejection of the example, while a positive decision from the previous stage triggers the evaluation of the next stage. So, the input example is classified as positive only if it passes the tests of all the stages. Simple classifiers are used to exclude the majority of negatives examples before more complex classifiers are called upon. Even though there are possibly many stages in the final FloatCascade classifier, most are not evaluated for a typical negative example since it has been excluded by the early stages. As a consequence, the classification speed is raised greatly. Though the testing process of FloatCascade is seemingly similar to AsyCascade, FloatCascade has two important improvements in classification performance: 1) the classification time is further reduced because fewer features are required for classification; 2) the classification accuracy is further raised because more effective features are found for classification.

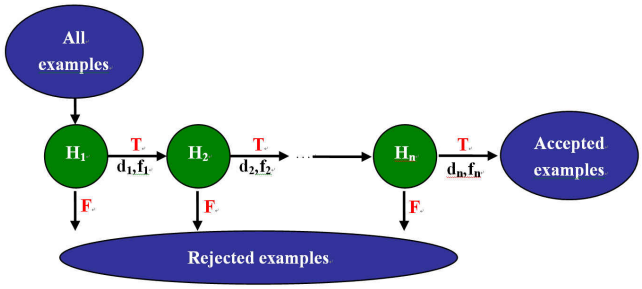

Figure 4. The testing process of FloatCascade.

From Figure 4, we can find out why FloatCascade can achieve better classification accuracy than AsyCascade. Assuming that all ensemble classifiers $\{H_i\}$ are constructed independently, the detection rate and the false positive rate of the overall cascade classifier are respectively given by $\prod_{i=1}^{n} d_i$ and $\prod_{i=1}^{n} f_i$. Since each ensemble classifier of FloatCascade is trained to achieve lower false positive rate f_i than AsyCascade under the same threshold ϑ of the detection rate d_i, FloatCascade is able to beat AsyCascade in overall classification accuracy hopefully.

3.3 Decision-tree Feature

FloatCascade attempts to use fewer but more effective features to build the ensemble classifier. At this time, the quantity and quality of available features become the key factors to its success.

- ***Feature quantity***. The success of AsyCascade applied to face detection benefits from a very large and varied set of features available. For example, over six million Harr-like features were extracted in [9, 10]. Unfortunately, in many web mining tasks, it is hard to extract so many features. In the case of citation matching, a citation only consists of several short text fields, so it lacks of enough contexts for feature extraction. For web page categorization, each term is commonly used as a feature. Even so, the total number of terms is generally far smaller than six million. The issue of feature quantity is more important to FloatCascade. The limited number of features reduces the possibility of finding fewer but more effective features.
- ***Feature quality***. Even though there have been a lot of features available, it is still difficult to find effective features if each of

them has weak discrimination capability. The useable features should exhibit sufficient diversity, which requires that each feature should capture the different characteristic of the data. Classification speed and classification accuracy of the first few stages is critically important to the overall performance of the cascade classifier, because it needs to find a very few features in these early stages for rejecting many negative examples and meanwhile accepting almost all positive ones. The issue of feature quality is especially significant to FloatCascade. Only when there are enough effective features, the number of required features at each stage could be reduced and the accuracy of the ensemble classifier could be raised.

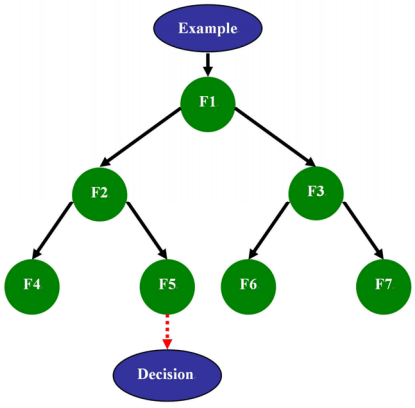

Figure 5. A three-level decision-tree feature.

To enhance feature diversity and discrimination capability, a three-level decision scheme is adopted to build a tree feature for FloatCascade learning. Such a kind of decision-tree feature has also been successfully used in [27, 28] for text categorization. A decision-tree feature consists of seven nodes, which is illustrated in the Figure 5. Each node contains a single feature, e.g., a term feature in web page categorization and a metric feature in citation matching. Since each node can select a different feature and each feature can set a different threshold, a considerable number of decision trees could be generated and the available decision-tree features are vastly enriched. In addition, since a variety of features are now combined as an overall decision-tree feature, the feature discriminative power is significantly enhanced. In fact, a single feature can be viewed as a one-level decision-tree feature just corresponding to the root note, which is extensively used in the community of face detection [9, 10].

3.4 Discussion

3.4.1 Comparison and Analysis

Float searching scheme has also been used in FloatBoost [19] to remove features resulting in higher error rates. Nevertheless, FloatCascade and FloatBoost are different in principle. Essentially, the former improves AsyCascade while the latter improves AsyBoost. FloatBoost follows AsyBoost in the way of minimizing a quantity related to error rate [19], which is at best an indirect way of meeting the learning objective of cascade learning as pointed out in [31, 32]. In the context of cascade learning, the learning objective of the ensemble classifier is to achieve high false negative rate and moderate false positive rate instead of a minimum error rate [9]. FloatCascade is specially designed to improve the cascade classifier using float searching. It makes the largest improvement in the false positive rate of each ensemble classifier when the detection rate is satisfied, which is directly consistent with the learning goal of asymmetric cascade classifier.

It is well known that most of algorithms have to trade efficiency for effectiveness, or vice versa. But, our FloatCascade can achieve high classification speed as well as good classification accuracy. Based on the following experiments, the classification accuracy of FloatCascade is even comparable to non-cascade methods. In AsyCascade, it focuses too much on the positive examples at the cost of too many misclassified negative examples, which results in an unfavorable decrease in overall classification accuracy. By contrast, FloatCascade attains a satisfactory balance between false negative rate and false positive rate and ensures that the overall classification accuracy is not decreased and even slightly raised.

Notably, asymmetric cascade learning is essentially independent of AsyBoost. In other words, AsyCascade and FloatCascade can be implemented without AsyBoost. In fact, any feature addition method (even random addition) can be used to provide candidate features for ensemble classifiers. After that, FloatCascade adopts float searching to remove some deleterious features for enhanced ensemble classifiers. ***Its key insight is that feature refinement (i.e., feature deletion in FloatCascade) can remedy the accuracy loss caused by continuous feature addition in the way of directly maximizing the learning objective of the cascade learning***.

The implementation of AdaBoost, AsyBoost, AsyCascade and FloatCascade are relatively easier than other popular classification models, such as SVM. AdaBoost has only one parameter, namely the iteration number. In the experiment, we set it be 100 and report the best accuracy. AsyBoost has an additional parameter k for asymmetric cost. We automatically set it to be the ratio of negative examples over positive examples in each category. FloatCascade and AsyCascade have the same parameter settings. As explained in Section 3.2.3, some important parameters such as F_{target}, ϑ and f can be determined according to the learning objective of cascade classifier. In our experiment, we set $F_{target} = 1e-6$, $\vartheta = 0.99$ and $f = 0.5$. Once these parameters are set, the training of FloatCascade and AsyCascade is fully data-driven requiring no any manual intervention and inspection.

3.4.2 The Generality of FloatCascade Learning for Imbalanced Web Miming

Arising from the extensive impact of "Matthew Effect" on the social web, the problem of imbalanced classification occurs very often in web mining. Besides web page categorization and citation matching, there are also many other web IC applications, such as

- Web search. Though there are a great number of web pages, only a very small number of ones are really relevant to a search query, even if search engines have filtered so many irrelevant ones in their searching results.
- Spam detection. Generally, there are only a very small amount of spam documents compared with a very large amount of legitimate ones, such as spam emails and spam web pages.
- Keyword extraction. Usually, only several terms occurring in web pages can be served as their keywords, even if the web page contains many textual terms in its body.

Fortunately, all of these web mining tasks could be solved in the framework of learning to classification. The general solution is to

define some task-specific features, extract features from the web example, and accomplish the classification in the feature space. Some prior work has actually complied with this paradigm to attack these problems, such as fRank for static ranking [42], SVM for spam detection [43] and learning to extract keywords from web pages [44]. Our FloatCascade also follows this learning framework, but more importantly, it additionally exploits the imbalance nature in these web IC problems to achieve desirable improvement in both classification effectiveness and classification efficiency. In fact, many web mining problems can be attacked in the framework of imbalanced classification by proper and subtle problem reformulation. Considering the popularity of web IC problems and the generality of our FloatCascade, we expect that FloatCascade is very promising for many web mining applications.

We highlight the importance of feature to learning framework of imbalanced web classification in two aspects: 1) it should extract specific features in specific mining tasks. For example, in the task of web page categorization, it can use common word feature; however, in the scenario of citation matching, it ought to extract similarity metric features from citation pair. 2) it should enhance the discrimination and diversity of available features as possible. The decision-tree scheme might be a good option. We will verify the feature importance for FloatCascade learning in the following experiments.

4. EXPERIMENTS

We evaluate FloatCascade on the tasks of web page categorization and citation matching, two typical web IC applications. We first introduce experimental methodology in Section 4.1. Then, we present the experiments on web page categorization in Section 4.2 and citation matching in Section 4.3 respectively.

4.1 Experimental Methodology

We evaluate all the classification algorithms in terms of efficiency as well as effectiveness. The testing time (millisecond) is recorded for each algorithm, keeping the computer in the same situation as possible, e.g., CPU usage and memory usage. We use such recorded time to distinguish the two algorithms in classification efficiency if there is an evident gap between them. The computer used for our experiments is equipped with Intel Core2 processor 2.66GHz and 2.0 GB memory.

Table 1. The distribution of the examples after classification.

	Positive	Negative
TRUE	*NTP*	*NTN*
FALSE	*NFP*	*NFN*

After classification, testing examples fall into four cases shown in Table 1, where *NTP* and *NTN* respectively denotes the number of the positive and negative examples correctly predicted, while *NFN* and *NFP* respectively denotes the number of the misclassified positive and negative examples. We evaluate the accuracy of imbalanced classification in two aspects. One is about retrieving positive examples and the other is about classifying all the examples. The standard measures are adopted for both aspects. Specifically, retrieval accuracy is evaluated with precision (*P*), recall (*R*) and F1-measure (*F1*), and classification accuracy is evaluated with false negative rate (*FN*), false positive rate (*FP*) and error rate (*ER*). These measures are calculated as follows:

$$P=NTP/(NTP+NFP),\ R=NTP/(NTP+NFN),\ F1=2PR/(P+R),$$

$$FN=NFN/(NTP+NFN),\ FP=NFP/(NTN+NFP),$$

$$ER=(NFP+NFN)/(NTP+NTN+NFP+NFN).$$

F1 and *ER* are the two overall measures. The higher *F1* is, the better the retrieval accuracy is. The lower *ER* is, the better the classification accuracy is. Though the sum of *R* and *FN* is 1 by their definitions, we still list all these measures in the experiments for the sake of completeness and convenient comparison.

In the experiment of web page categorization, two popular methods including NB and SVM are implemented. NB assumes a generative model and uses the joint probability of words in document to estimate the possibility of a document belonging to a category. SVM seeks a hyper-plane in a high dimensional kernel space to discriminate the positive and negative examples with maximum margin. Besides, a popular IC method, Weighted SVM (WSVM) [23], is implemented where positive examples out-weight negative examples in their training cost. We implement SVM and WSVM with linear kernel using SVMlight [40]. In the experiment of citation matching, SVM and WSVM are carried out for comparison. We use our own implementation of AdaBoost (Ada), AsyBoost (AA), AsyCascade (AC) and FloatCascade (FC).

4.2 Experiment on Web Page Categorization

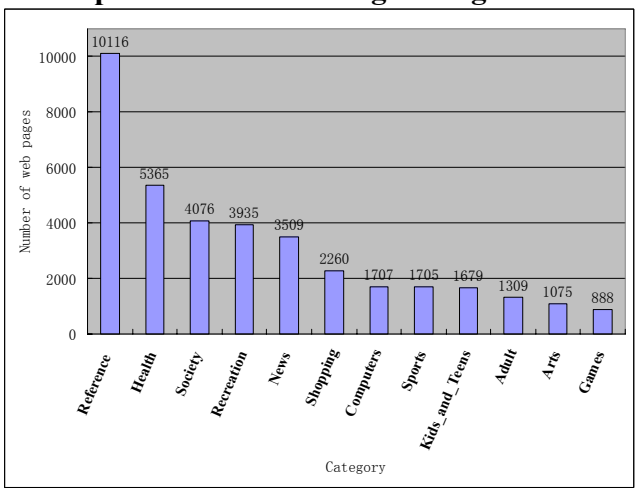

Figure 6. The web pages crawled from the ODP web directory in its 12 first-level categories.

Web page categorization is a typical multi-class and multi-label classification problem [13, 27]. Besides the inherent imbalance of web categories, imbalance effect is enhanced by the one-against-all learning strategy for transforming such a multi-class and multi-label classification problem into a binary classification problem, where training examples belonging to one category are taken as positive examples and training examples not belonging to this category as taken as negative examples. We also follow this popular way to train the investigated classifiers except *NB*. We first introduce the dataset with the stress of the imbalanced categories and then give the experimental results.

4.2.1 Dataset

The web pages used in our experiment are crawled from the ODP web directory. On the ODP, all the web pages have been manually classified into hierarchical taxonomy. We have crawled totally 37,624 web pages from its 12 first-level categories. The dataset exhibits highly imbalanced distribution as plotted in the Figure 6.

The maximum category contains 10116 pages while the minimum category only contains 888 pages. In the experiments, all the web pages are preprocessed by the following steps: (1) passing all words through the word stemmer; (2) tossing out all the stop tokens; (3) normalizing the documents into TFIDF vectors.

4.2.2 Result

The dataset is randomly split into 10 groups. Eight groups are used as the training set and the remained two ones are used as the validation and testing sets respectively. We repeat such trails for 20 times using the three-level decision-tree feature. The experimental results averaged over both 12 categories and 20 runs are reported in Table 2. We can see that:

Table 2. The experimental results on the ODP dataset.

	P	R	F1	FN	FP	ER	Time
NB	0.3115	0.7293	0.4365	0.2707	0.1742	0.1704	3501
SVM	0.4526	0.6255	0.5252	0.3745	0.0770	0.0929	3913
WSVM	0.4483	0.6552	0.5324	0.3448	0.0733	0.0959	4200
Ada	0.4761	0.6617	0.5537	0.3383	0.0808	0.0931	1750
AA	0.4618	0.6946	0.5548	0.3054	0.0736	0.0929	1718
AC	0.3904	0.7344	0.5098	0.2656	0.1043	0.1177	1313
FC	0.4999	0.6167	0.5522	0.3833	0.0762	0.0959	593
Improv			**8.33%**			**18.52%**	**54.84%**

- **AdaBoost and SVM vs. NB**. AdaBoost and SVM both significantly outperform NB in classification accuracy. It may suggest that a discriminative model is more suitable for imbalanced web page categorization than a generative model.
- **AdaBoost vs. SVM**. AdaBoost is slightly better than SVM in terms of *F1* and *ER*. However, AdaBoost is faster than SVM more than 2 (*3913/1750*) times, which benefits from simple threshold decision in its weak classifier rather than expensive kernel mapping in SVM. It implies that AdaBoost is a better choice than SVM for cascade learning since the classification speed of ensemble classifiers is very important to it.
- **IC vs. balanced classification (BC)**. IC is much better than BC in recall (from 66.17% to 69.46% when AA vs. Ada and from 62.55% to 65.52% when WSVM vs. SVM). It verifies that IC can surely retrieve more positive examples than BC.
- **Cascade vs. Non-cascade**. Cascade method can greatly reduce the classification time. The classification speed of AC is raised by at least 23.57% (*405/1718*) comparing to Ada and AA and at least 66.45% (*2600/3913*) comparing to SVM and WSVM. It mainly benefits from the fact that the considerable number of negative examples is excluded at the early stages in AC just using a small number of features.
- **FC vs. other methods**.

  FC vs. AC. Unfortunately, AC observably performs worse than non-cascade IC methods in terms of both *F1* and *ER*. For example, *F1* is decreased by 8.11% in *F1* (*0.045/0.5548*) and *ER* is decreased as much as 21.07% (*0.0248/0.1177*) compared with AA. The improvement in retrieving positive examples is counteracted by the misclassification of too many negative examples, and the overall classification accuracy is remarkably decreased. That is to say, AC achieves fast classification at the expense of classification accuracy. By contraries, the accuracy of FC is significantly improved compared with AC, with 8.33% increase in *F1* and 18.52% increase in *ER*. The misclassification of negative examples is effectively controlled. Moreover, FC successfully seeks a balance between the false positive rate and the false negative rate resulting in an evident improvement in overall classification accuracy. In fact, FC approaches the classification accuracy of AA in terms of *F1* and *ER*. Further, the classification time of FC is reduced by as much as 54.84% (*720/1313*) compared with AC. It attributes to the fact that FC successfully uses fewer and more effective features in the early stages to get rid of more negative examples.

  FC vs. Non-cascade. The most distinct advantage of FC over non-cascade IC methods is its distinct fast classification speed, being almost 3 (*1718/593*) times faster than Ada and AA and over 6 (*3913/593*) times faster than SVM and WSVM. At the same time, the classification accuracy is well comparable to AA and WSVM. Delightedly, FC is even slightly better than SVM and WSVM in *F1*. It manifests the effectiveness and efficiency of FC comparing to the state-of-the-art IC methods.

4.2.3 Efficiency Analysis

In order to better explain why FC is faster than AA, Ada and AC, we figure out the average number of the features used in these classifiers. These algorithms all take the decision-tree feature as the weak classifier. If we assume that each tree feature takes the equal time to classify an example, we can approximately compare the classification efficiency of these algorithms using the times of decision-tree classification. In the above experiment, there are averagely 43 features in Ada, 42 features in AA, 89 features in AC and 55 features in FC. As seen, the number of features in FC is reduced by 38.20% (*34/89*) compared with AC. This demonstrates that FC can effectively remove some unfavorable features from AC. Though Ada and AA use fewer features than AC and FC for classification, they treat all the examples indiscriminatingly. That is, all the positive and negative examples are evaluated using all the decision-tree features. As comparison, it is found that over 80% of negative examples are discarded in the first stage of AC and FC with less than 3 trees. Decision-tree classification in the later stages is only conducted on the minor promising examples. As a result, the required times of decision-tree classification in AC and FC are greatly reduced compared with AA and Ada.

FC is further faster than AC for classification. As an illustration, we examine the cascade classifier of FC and AC for the category of "News". The number of features at each stage of FC is 1, 2, 3, 4, 5, 5, 6, 7, and 8, while 3, 6, 7, 8, 9, 13, and 18 in AC. As observed, FC successfully uses fewer features than AC at all stages. Importantly, it is found that that some features in AC are replaced by FC with more effective ones for classification. Consequently, more negative examples are quickly excluded by FC than AC at earlier stages. In summary, the reduced number and the enhanced effectiveness of the features both contribute to the faster classification of FC than AC.

4.2.4 Experiment on Text Data and Feature Analysis

We also apply FloatCascade to the traditional text document for categorization in order to further investigate its imbalanced classification performance. The experiment is conducted on the Reuters-21578, a benchmark corpus for text categorization. The corpus consists of 7769 training documents and 3019 testing documents, both belonging to the common 90 categories. All the training and testing examples belonging to one category are merged to the larger category and then randomly split into 10 groups to construct the training, testing and validation sets like last experiment. We also repeat evaluation trials for 20 times in all.

In our work, we are mainly concerned with IC problem. In order to enhance the effect of data imbalance and analyze the effectiveness of FloatCascade to hard IC problems, we specially select 10 small categories from the Reuters-21578 for reporting experimental result averaged over these 10 categories. The selected categories are listed in the Table 3 and the experimental result is shown in the Table 3. Note that these results are also averaged over 20 runs. We can draw the similar conclusion to the last experiment, which indicates that FloatCascade is adaptable to various types of imbalanced data. In particular, FC outperforms AC in *F1* by 6.75% and in *ER* by 39.51%. One again, FC is two times faster than AC and many times than non-cascade methods.

Table 3. The 10 small categories selected from the Reuters-21578 corpus for the experiment.

Topic	corn	wheat	oilseed	soybean	sugar
Topic	nat-gas	cpi	bop	alum	dlr

Table 4. The experimental results on the Reuters-21578 corpus.

Text categorization using decision-tree feature							
	P	R	F1	FN	FP	ER	Time
NB	0.6845	0.5064	0.5821	0.4936	0.0038	0.0118	3015
SVM	0.7651	0.6433	0.6990	0.3567	0.0030	0.0078	3813
WSVM	0.8048	0.6936	0.7375	0.3064	0.0027	0.0071	4000
Ada	0.8331	0.6278	0.7160	0.3722	0.0027	0.0071	641
AA	0.7985	0.6823	0.7358	0.3177	0.0035	0.0071	625
AC	0.5819	0.8725	0.6982	0.1275	0.0149	0.0162	188
FC	0.6615	0.8534	0.7453	0.1466	0.0081	0.0098	94
Improv			**6.75%**			**39.51%**	**50.00%**
Text categorization using single feature							
	P	R	F1	FN	FP	ER	Time
Ada	0.5793	0.3315	0.4217	0.6685	0.0011	0.0105	328
AA	0.6335	0.5001	0.5590	0.4999	0.0031	0.0095	360
AC	0.4292	0.9300	0.5874	0.0700	0.1844	0.1830	125
FC	0.4398	0.9300	0.5880	0.0700	0.1843	0.1830	123

Besides, to all compared algorithms, the classification accuracy of text categorization is distinctly much better than that of web page categorization. It should attribute to the availability of good features on text data. The documents in Reuters-21578 corpus are news articles edited by human and generally have better quality than web pages. Thus, it is more likely to extract effective decision-tree features from such text data. To further validate the importance of decision-tree feature for the cascade learning, we also conduct the experiments of Ada, AA, AC and FC using just a single term feature on Reuters-21578 collection. The experimental results on the selected 10 categories are also listed in Table 3 for convenient comparison. It can be observed that: 1) all four algorithms perform much worse with a singe feature. Using features with poor discrimination capability, Ada and AA fail to achieve good accuracy though they combine multiple features. FC and AC are also hard to achieve a satisfactory accuracy though they combine multiple AA classifiers; 2) the classification time of Ada and AA is superficially reduced to almost half. This is only because a decision tree requires approximately 2 times time of a single feature for classifying an example; 3) FC merely approaches the accuracy and speed of AC. This is mainly owing to that FC almost cannot remove or replace the features used in AC. These experimental results bear out the significance of features to the success of FloatCascade learning.

4.3 Experiment on Citation Matching

Experiments on citation matching are done on the Cora dataset, an extensively used citation corpus. We first introduce this dataset and then describe the features extracted in our experiment. After that, we give the experimental results and analysis.

4.3.1 Dataset

William W. Cohen, Rob Schapire, and Yoram Singer. Learning to order things. 1997.

William W. Cohen, Robert E. Schapire, and Singer Yoram. Learning ro order things. 1997. In Advances in Neural Information Processing Systems 10.

W. W. Cohen, R. E. Shapire, and Y. Singer. Learning to order things. 1998. In Advances in Neural Information Processing Systems 10.

William W. Cohen, Robert E. Schapire, and Yoram Singer. Learning to order things. 1998. In Advances in NIPS 10.

W. W. Cohen, R. E. Shapire, and Y. Singer. Learning to order things. 1998. In Advances in Neural Information Processing Systems 10.

Figure 7. An example of citations that refer to the same paper.

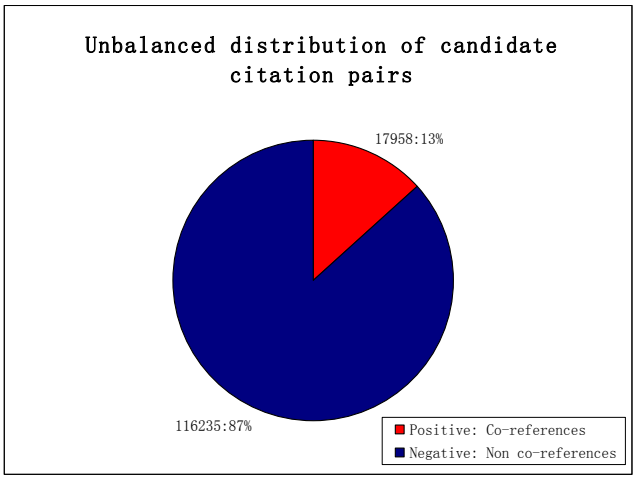

Figure 8. Unbalanced distribution of candidate citation pairs.

In Cora, there are 1295 distinct citations to 122 computer science research papers. Each citation was composed of multiple fields including author, title and venue etc. Figure 7 shows several citations referring to the same paper in spite of many literal differences. We perform citation matching by identifying whether a pair of citations actually refers to the same one publication. The number of pair-wise similarity computations grows quadratically with the size of the dataset. It is prohibitively expensive to accomplish this computation on the Cora (C_{1295}^2 pairs) dataset. Fortunately, the majority of citation pairs are obviously dissimilar. We sort out candidate citation pairs sharing at least one term. Finally, we obtain 17,958 target co-references and 116,235 non co-references, which is under a distinct imbalanced distribution as illustrated in Figure 8.

4.3.2 Features

In this section, we briefly describe the features used in our experiments. These features are extracted from each citation pair

under the help of SecondString [32] and SimMetrics [41], two open source toolkit for calculating distance metrics between short strings. The adopted features can be loosely grouped into several categories. The first category refers to character-based metric, such as edit distance and Jaro metric. The second one is token-based metric, such as Jaccard metric and TFIDF cosine metric. The third one involves statistics-based metric, such as Jensen-Shannon metric and SFS metric. The last category corresponds to hybrid metric, including two-level metrics and SoftTFIDF metric. The specific definitions of these metrics can refer to [32]. Finally, we extract totally 37 similarity metrics as features.

Comparing to a text document and an image, it can only extract a minority number of features from a citation pair due to the lack of enough contexts. Like last experiment, we also adopt the decision-tree to enrich and enhance the feature space for citation matching.

4.3.3 Result

For evaluation, all the positive and negative examples are equally divided into the three subsets. These three subsets are alternately taken as the training, testing and validation set. We repeat all possible combinations for experiments, totally 6 times ($C_3^1 \times C_2^1 \times C_1^1$). The averaged results are shown in Table 4. The main conclusions made in the web page and text data still hold here. In specific, 1) AC is much faster than non-cascade methods (SVM, WSVM, Ada and AA), and FC is further faster than AC; 2) IC methods (AA and WSVM) retrieve more co-references than BC methods (Ada and SVM); 3) The overall accuracy of AC is significantly decreased in comparison with non-cascade methods; 4) FC achieves a much better classification accuracy than AC. It is comparable to or even better than the non-cascade methods.

Table 5. The experimental results on the Cora dataset.

	P	R	F1	FN	FP	ER	Time
SVM	0.9007	0.8922	0.8964	0.1078	0.0180	0.0319	9975
ASVM	0.8591	0.9390	0.8973	0.0610	0.0281	0.0332	10236
Ada	0.8126	0.9739	0.8860	0.0261	0.0411	0.0387	5563
AA	0.7923	0.9761	0.8745	0.0239	0.0469	0.0433	5408
AC	0.6838	0.9902	0.8080	0.0098	0.0848	0.0732	187
FC	0.7976	0.9702	0.8747	0.0298	0.0456	0.0431	171
Improv			**9.31%**			**41.06%**	**8.56%**

However, it also reveals some difference with the last experiments: 1) AA method performs worse than Ada methods in terms of *F1* and *ER*. As an asymmetric learning method, AA fails to seek a balance between false negative rate and false positive rate. 2) The classification time of FC is merely reduced by 8.56% (*16/187*) compared with AC, much smaller than 54.84% and 50% in the previous experiments. The main reason is that the metric feature used in citation matching has better discrimination capability than the term feature used in the last two experiments. In this case, AC itself contains just a small number of features and it leaves not much room for FC to reduce the number of the features. Even so, FC still makes a significant accuracy improvement by replacing some features with more effective ones.

5. CONCLUSIONS

The problem of imbalanced classification may occur very often on the web due to extensive impact of "Matthew Effect". Fast classification is in urgent need for those large-scale and on-line web IC applications. In this paper, we investigate the feasibility of cascade learning for fast imbalanced classification in web mining, and propose a novel asymmetric cascade learning algorithm called FloatCascade to improve the accuracy of AsyCascade. Our key insight is that feature refinement (namely feature deletion in FloatCascade) can remedy the accuracy loss caused by continuous feature addition in AsyCascade. FloatCascade successfully selects less yet more effective features at each stage of the cascade classifier by minimizing the corresponding false positive rate, which meets the actual objective of asymmetric cascade learning. Besides, the quality and quantity of available features is critically important to the success of FloatCascade learning. A decision-tree feature is adopted to enhance feature diversity and discrimination capability for FloatCascade learning.

Encouraging experimental results on web page categorization and citation matching demonstrate the effectiveness and efficiency of FloatCascade learning for imbalanced web classification. Some important experimental findings include: 1) FloatCascade can achieve much higher classification speed than AsyCascade. 2) FloatCascade can significantly improve the classification accuracy of AsyCascade. It is even comparable to some non-cascade methods. 3) FloatCascade can attain a favorable balance between the false negative rate and the false positive rate and ensures the significant increase in the overall classification accuracy.

The main contributions of this paper are summarized as follows:

1. We propose to study the problem of imbalanced web mining, and investigate the applicability of cascade learning for fast imbalanced classification in web mining.
2. We propose a new asymmetric cascade learning method called FloatCascade to achieve higher classification speed and better classification accuracy than AsyCascade, and we also highlight the importance of feature for FloatCascade learning.

Considering the popularity of web IC problems and the generality of our FloatCascade learning, we expect that FloatCascade is very promising for many imbalanced web mining applications. In the future, we will go on improving FloatCascade in two directions: better classification accuracy and higher classification speed. And we will apply FloatCascade to more web IC problems for fast and accurate classification.

6. REFERENCES

[1] N. Japkowicz. Learning from Imbalanced Data Sets: A Comparison of Various Strategies, In *Learning from imbalanced data sets: The AAAI Workshop 10-15*. Technical Report WS-00-05, Menlo Park, CA: AAAI Press, 2000.

[2] H. Liu and H. Motoda. On Issues of Instance Selection. In *Journal of Data Mining and Knowledge Discovery*, pp. 115-130, 2002.

[3] D. Fragoudis, D. Meretakis, and S. Likothanassis. Integrating Feature and Instance Selection for Text Classification. In *Proc. of ACM SIGKDD 2002*, pp. 501-506, Canada, 2002.

[4] M. Kubat and S. Matwin. Addressing the Curse of Imbalanced Training Sets: One Sided Selection. In *Proc. of ICML 1997*, pp. 179-186, 1997.

[5] C. Chen, H. Lee, and M. Kao. Multi-class Svm with Negative Data Selection for Web Page Classification. In *Proc. of IEEE Joint Conf. on Neural Networks*, pp. 2047- 2052, Budapest, Hungary, 2004.

[6] J. Brank, M. Grobelnik, N. M. Frayling, and D. Mladenic. Training Text Classifiers with SVM on Very Few Positive

Examples. *Technical Report MSR-TR-2003-34*, Microsoft Research, April 2003.

[7] G. Wu and E. Y. Chang, Kba: Kernel Boundary Alignment Considering Imbalanced Data Distribution. *IEEE Trans. on Knowledge and Data Engineering (TKDE)*, 17(6): pp. 786-795, June 2005.

[8] E. S. Robert. A Brief Introduction to Boosting. In *Proc. of IJCAI 1999*, pp. 1401-1405, Stockholm, Sweden, 1999.

[9] V. Paul and J. Michael, Fast and Robust Classification Using Asymmetric AdaBoost and a Detector Cascade. In *Proc. of NIPS 2001*. pp. 1311-1318, 2001.

[10] V. Paul and J. Michael, Robust Real-Time Face Detection. In *Journal of International Journal of Computer Vision (IJCV)*, pp. 137-154. Kluwer Academic Publishers, Netherlands, 2004.

[11] D. Shen, J. Sun, Q. Yang, and Z. Chen. A Comparison of Implicit and Explicit Links for Web Page Classification. In *Proc. of WWW 2006*, pp. 643-650, 2006.

[12] E. J. Glover, K. Tsioutsiouliklis, S. Lawrence, D. M. Pennock, and G. W. Flake. Using Web Structure for Classifying and Describing Web Pages. I*n Proc. of WWW 2002*, pp. 562-569, Honolulu, Hawaii, USA, 2002.

[13] H. Oh, S Myaeng, and M. Lee. A Practical Hypertext Categorization Method using Links and Incrementally Available Class Information. In *Proc. of SIGIR 2000*, pp. 264-271, Athens, Greece, 2000.

[14] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of ECML 1998*, pp. 137-142, Chemnitz, Germany, 1998.

[15] A. McCallum and K. Nigam. A Comparison of Event Models for Naive Bayes Text Classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pp. 22-28, 1998.

[16] S. Lessmann. Solving Imbalanced Classification Problems with Support Vector Machines. In *Proc. of the Int. Conf. on Artificial Intelligence (IC-AI'04)*, pp. 214-220, Las Vegas, Nevada, USA,

[17] A. Sun, E. Lim, B. Benatallah, and M. Hassan. FISA: Feature-Based Instance Selection for Imbalanced Text Classification. In *Proc. of PAKDD 2006*. pp. 250-254, 2006.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. In *Journal of Artificial Intelligence Research*, 16: pp. 321-357, 2002.

[19] S. Z. Li, Z. Zhang, H. Shum, and H. Zhang. FloatBoost Learning for Classification. *In Proc. of NIPS 2002*, pp. 993-1000, 2002.

[20] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. Adacost: Misclassification Cost-sensitive Boosting. In *Proc. of ICML 1999*, pp. 97-105, 1999.

[21] Y. Ma and X. Q. Ding. Robust Real-time Face Detection based on Cost-sensitive AdaBoost Method. In *Proc. of ICME 2003*, pp. 465-468, 2003.

[22] K. M. Ting and Z. Zheng. Boosting Trees for Cost-sensitive Classifications. In *Proc. of the ECML 1998*. pp. 190-195, 1998.

[23] K. Morik, P. Brockhausen, and T. Joachims, Combining Statistical Learning with a Knowledge-based Approach - A Case Study in Intensive Care Monitoring. In *Proc. of ICML 1999*, pp. 268-277, 1999.

[24] A. K. McCallum, K. Nigam, and L. Ungar. Efficient Clustering of High-dimensional Data Sets with Application to Reference Matching. In *Proc. of KDD2000*, pp. 169–178, Boston, MA, 2000.

[25] R. Akbani, S. Kwek, and N. Japkowicz. Applying Support Vector Machines to Imbalanced Datasets. *In Proc. of ECML 2004*, pp. 39-50, 2004.

[26] X. Hou, C. Liu, and T. Tan. Learning Boosted Asymmetric Classifiers for Object Detection. In *Proc. of CVPR 2006*. pp.330-338, , New York, 2006.

[27] N. Bobb. BiBoost for Asymmetric Learning. *Technical Report*, University of California, 2006.

[28] J. Wu, J. M. Rehg, and M. D. Mullin. Learning a Rare Event Detection Cascade by Direct Feature Selection. In *Proc. of NIPS 2003*, pp. 1523-1530, 2003.

[29] J. Wu, M. D. Mullin, and J. M. Rehg. Linear Asymmetric Classifier for Cascade Detectors. In *Proc. of ICML 2005*. pp. 988-995, 2005.

[30] P. Pudil, J. Novovicova, and J. Kittler. "Floating Search Methods in Feature Selection". In *Journal of Pattern Recognition Letters*, (11): pp. 1119–1125, 1994.

[31] H. Pasula, B. Marthi, B. Milch, S. Russell, and I. Shpitser. Identity Uncertainty and Citation Matching. In *Proc. of NIPS 2002*. pp. 1401-1408, 2002.

[32] W. W. Cohen. P. Ravikumar, and S. E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proc. of IJCAI 2003 Workshop on Information Integration on the Web*, pp. 73-78, 2003

[33] M. Kubat, R. Holte, and S. Matwin. Machine Learning for the Detection of Oil Spills in Satellite Radar Images. In *Journal of Machine Learning*, pp.195-215, 1998.

[34] K. Sung and T. Poggio. Example-based Learning for View-based Human Face Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI)*, 20(1): pp. 39-51, 1998.

[35] http://www.dmoz.org

[36] http://dir.yahoo.com

[37] http://directory.google.com

[38] http://scholar.google.com/

[39] http://citeseer.ist.psu.edu/

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

[41] http://www.dcs.shef.ac.uk/~sam/simmetrics.html

[42] M. Richardson, and A. Prakash, and E. Brill. Beyond PageRank: Machine Learning for Static Ranking. *In Proc. of WWW2006*, pp. 707-715. May 23-26, 2006.

[43] H. Drucker, D. Wu, and V. N. Vapnik, Support Vector Machines for Spam Categorization, *IEEE Trans. on Neural Networks*, 20(5): pp. 1048-1054, 1999.

[44] W. Yih, J. Goodman, and V. R. Carvalho. Finding Advertising Keywords on Web Pages. In *Proc. of WWW 2006*, pp 213-222. 2006.