Knowl Inf Syst (2008) 16:281–301
DOI 10.1007/s10115-007-0107-1

REGULAR PAPER

SVM based adaptive learning method for text classification from positive and unlabeled documents

Tao Peng · Wanli Zuo · Fengling He

Received: 5 April 2006 / Revised: 29 May 2007 / Accepted: 5 August 2007 /
Published online: 6 September 2007

Abstract Automatic text classification is one of the most important tools in Information Retrieval. This paper presents a novel text classifier using positive and unlabeled examples. The primary challenge of this problem as compared with the classical text classification problem is that no labeled negative documents are available in the training example set. Firstly, we identify many more reliable negative documents by an improved 1-DNF algorithm with a very low error rate. Secondly, we build a set of classifiers by iteratively applying the SVM algorithm on a training data set, which is augmented during iteration. Thirdly, different from previous PU-oriented text classification works, we adopt the weighted vote of all classifiers generated in the iteration steps to construct the final classifier instead of choosing one of the classifiers as the final classifier. Finally, we discuss an approach to evaluate the weighted vote of all classifiers generated in the iteration steps to construct the final classifier based on PSO (Particle Swarm Optimization), which can discover the best combination of the weights. In addition, we built a focused crawler based on link-contexts guided by different classifiers to evaluate our method. Several comprehensive experiments have been conducted using the Reuters data set and thousands of web pages. Experimental results show that our method increases the performance (F_1-measure) compared with PEBL, and a focused web crawler guided by our PSO-based classifier outperforms other several classifiers both in *harvest rate* and *target recall*.

Keywords Text classification · Machine learning · Improved 1-DNF algorithm · SVM · PSO · Focused web crawling

T. Peng (✉) · W. Zuo · F. He
College of Computer Science and Technology, Jilin University, No.2699 Qianjin Road, Changchun, Jilin 130012, China
e-mail: taopengpt@yahoo.com.cn

W. Zuo · F. He
Key Laboratory of Symbol Computation and Knowledge Engineering of the Ministry of Education, Changchun 130012, China

1 Introduction

Text classification is the process of assigning predefined category labels to new documents based on the classifier learnt from training examples. Text classification is of great practical importance today given the massive amount of text available. With the rapid growth of information and the explosion of electronic text from the World Wide Web, one way of organizing this overwhelming amount of documents is to classify them into descriptive or topical taxonomies. Text categorization is used to automatically catalog news articles [15], web pages [4,19], automatically learn the reading interests of users [16,23]. In recent years, a number of statistical classification and machine learning techniques have been applied to text categorization, including regression models [28], nearest neighbor classifiers [11], decision trees [17], Bayesian classifiers [17], Network informative combinations of features [29], support vector machines [26], etc. One key difficulty with traditional approachs is that they require a large, often prohibitive, number of labeled training examples to learn accurately. Labeling must often be done manually, which is a painfully time-consuming process. Collecting negative training examples is especially delicate and arduous because (1) negative training examples must uniformly represent the universal set excluding the positive class, and (2) manually collected negative training examples could be biased because of human's unintentional prejudice, which could be detrimental to classification accuracy. Recently, researchers investigated the idea of using a small labeled set of every class and a large unlabeled set to help learning [2,8,18,20,27], and this is referred as PU-Oriented text classification, which will reduce the manual labeling effort.

PU-oriented classification finds its application in topic-oriented crawling or focused crawling, which is particularly helpful for no labeled negative documents are available in the training example set. A topic-specific classifier is required to automatically identify whether the retrieved web page belongs to the specific category during the crawling process. Instead of collecting a negative example set for each specific domain, it is more desirable to construct a universal unlabeled example set for building any topic-specific classifiers due to the scale and diversity of negative examples.

This paper describes a new text classification process that uses the Particle Swarm Optimization (PSO) algorithm to evolve the weights of all classifiers generated in the iteration steps. PSO is an intelligent evolutionary method. PSO and other evolutionary algorithms have been used to solve many optimization problems [5,13,21,22]. We apply the PSO algorithm to search out and identify the potential informative feature combinations for classification and then use the F_1-Measure to determine the fitness value. PSO is motivated by the social behavior of organisms, such as bird flocking and fish schooling. In our approach, not as usual, an individual is a combination of the weights of the sub-classifiers, and it is more natural to represent the optimization problem in the continuous domain. For comparing the performance of the focused crawling guided by several text classification techniques, we also built a focused crawler based on Link-context.

The rest of the paper is organized as follows. Section 2 analyzes the related work. Section 3 details the whole process of building the text classifier. Section 3.1 introduces the principle of particle swarm optimization briefly. Section 3.2 describes how to improve the 1-DNF algorithm and identify reliable negative examples. The process of building a set of classifiers by iteratively applying SVM algorithm on the training example set illustrated in Sect. 3.3. Section 3.4 describes evolving the weights with PSO algorithm. Section 4 describes our focused web crawling based on link-context. Section 5 reports the results of our experiments. Section 6 draws the conclusion.

2 Related work

A theoretical study of Probably Approximately Correct (PAC) learning from positive and unlabeled examples was done in [6]. Later Denis and co-authors performed experiments by using k-DNF and decision trees to learn from positive and unlabeled data [8,18]. Bing Liu [1] presents sample complexity results for learning by maximizing the number of unlabeled examples labeled as negative while constraining the classifier to label all the positive examples correctly, and Bing Liu [1] presents S-EM algorithm (identifying a set of reliable negative documents by using a Spy technique and Building a set of classifiers by iteratively applying Expectation Maximization (EM) algorithm with a Naive Bayes (NB) classifier) and Roc-SVM algorithm [27] (identifying a set of reliable negative documents by using a Rocchio algorithm and Building a set of classifiers by iteratively applying SVM algorithm) to learn from positive and unlabeled examples. Bing et al. [2] summarize the usual method for solving the PU-oriented text classification. Hwanjo et al. [10] presents an algorithm called PEBL that achieves classification accuracy (with positive and unlabeled data) as high as that of traditional SVM (with positive and negative data). The PEBL algorithm uses the 1-DNF algorithm to identify a set of reliable negative documents and builds a set of classifiers by iteratively applying the SVM algorithm.

Besides maximizing the number of unlabeled examples labeled as negative, other methods for learning from positive and unlabeled examples are possible. Denis et al. [7] presents a NB based method (called PNB) that tries to statistically remove the effect of positive data in the unlabeled set. The main shortcoming of the method is that it requires the user to give the positive class probability, which is hard for the user to provide in practice.

It is also possible to discard the unlabeled data and learn only from the positive data. This was done in the one-class SVM [14], which tries to learn the support of the positive distribution. We implement the one-class SVM. Through experiments, we show that its performance is poorer than learning methods that take advantage of the unlabeled data.

3 Text classification

The goal of text categorization is the classification of documents into a fixed number of predefined categories. Each document D can be in multiple, exactly one, or no category at all. Using machine learning, the objective is to learn classifiers from examples that do the category assignments automatically. Constructing our text classifier adopts three steps: Firstly, identify a set of reliable negative documents from the unlabeled set by using our improved 1-DNF algorithm (1-DNFII). Secondly, build a set of classifiers by iteratively applying the SVM algorithm and construct the final WV (Weighted Voting) Classifier by using the weighted voting method based on the precision of each individual corresponding classifier. Thirdly, construct the final PSO Classifier by using the weighted voting method based on PSO.

3.1 Brief introduction of PSO

Particle Swarm Optimization, originally developed by Kennedy and Elberhart [12], is a method for optimizing hard numerical functions on metaphor of social behavior of flocks of birds and schools of fish. A swarm consists of individuals, called particles, which change their positions over time. These particles can be considered as simple agents "flying" through a problem space. A particle's location in the multi-dimensional problem space represents one solution for the problem. When a particle moves to a new location, a different problem solution

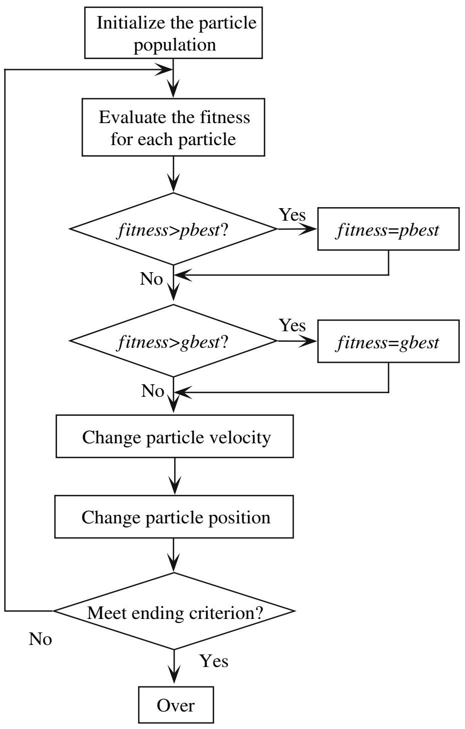


Fig. 1 The process of PSO algorithm for solving optimization problems

is generated. This solution is evaluated by a fitness function that provides a quantitative value of the solution's utility. Like the GA, the PSO is initialized with a population of random solutions. It also requires only the information about the fitness values of the individuals in the population. Let the ith particle in a d-dimensional space be represented as $X_i = (x_{i1}, x_{i2}, \ldots, x_{id}), i = 1, 2, \ldots, m$. The best previous position (which possesses the best fitness value) of the ith particle is denoted by $P_i = (p_{i1}, p_{i2}, \ldots, p_{id})$, which is also called $pbest_i$. The index of the best $pbest_i$ among all the particles is represented by the symbol g. The location $pbest_g$ is also called $gbest$. The velocity for the ith particle is represented as $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$. The concept of the particle swarm optimization consists of, at each time step, changing the velocity and location of each particle towards its $pbest_i$ and $gbest$ locations according to Eqs. (1) and (2), respectively:

$$V_i = wV_i + c_1r_1(pbest_i - X_i) + c_2r_2(pbest_g - X_i) \tag{1}$$
$$X_i = X_i + V_i \tag{2}$$

where w is the inertia coefficient which is a constant in the interval [0,1] and can be adjusted in the direction of linear decrease; two nonnegative constants, c_1 and c_2 are learning rates, which, some researchers have found out that setting c_1 and c_2 equal to 2 gets the best overall performance; r_1 and r_2 are generated randomly in the interval [0,1]. Figure 1 describes the process of particle swarm search.

3.2 Identifying reliable negative documents

For identifying the reliable negative documents from the unlabeled example set, we must identify the features of the negative documents. For example, if the frequency of occurrence

Fig. 2 Algorithm for identifying a set of reliable negative documents

```
Algorithm Improved 1-DNF (1-DNFII)
Assume the word feature set be $\{x_1,x_2,...,x_n\}$, $x_i$ is a word
feature that occurs in $U \cup P$;
PF = NULL;

RN=U; (RN: Reliable negative documents)
for ( i=1, i<= n, i++ ){
  if ( freq($x_i$, P)/|P| > freq($x_i$, U)/|U| and freq($x_i$, P)/|P|> $\lambda$ )
    PF=PF $\cup$ {$x_i$};
  }
for ( each document $d \in$ U) {
    if ( $\exists x_j$ freq($x_j$, d) > 0 and $x_j \in PF$ )
     RN=RN – {d};
  }
```

of a feature in the positive example set exceeds 90%, whereas less than 10% in the unlabeled example set, then this feature will be regarded as a positive one. Using this method, we can obtain a positive feature set *PF*. If some documents in the unlabeled example set do not contain any feature in the positive feature set *PF*, these documents can be regarded as reliable negative examples. For describing expediently, we define the following notation: P represents the positive example set, U represents the unlabeled example set, NEG_0 represents the reliable negative document set produced by our improved 1-DNF algorithm (1-DNFII), $NEG_i (i \geq 1)$ represents the negative document set produced by the ith iteration of the SVM algorithm, and *PON* represents the training example set.

In the 1-DNF algorithm [10], a positive feature is defined as a feature that occurs in the positive set P more frequently than in the unlabeled set U. We found that this definition has an obvious shortcoming: it only considers the diversity of the frequency of the feature in P and U, and does not consider the absolute frequency of the feature in P. For example, the frequency of some feature is 0.2% in the positive data set and 0.1% in the unlabeled data set. This feature is obviously not a positive feature. If we use the 1-DNF algorithm to identify negative data, this feature will be regard as a positive one. Accordingly, the number of documents in NEG_0 identified by the 1-DNF algorithm is less. Sometimes, NEG_0 may even be empty. From above discussion, we improved the 1-DNF algorithm (1-DNFII) by considering both the diversity of the feature frequency in P and U and the absolute frequency of the feature in P.

Definition 1 A feature is regarded as positive only when it satisfies the following two conditions:

1. The frequency of the feature occurring in the positive data set is greater than the frequency of the feature occurring in the unlabeled data set;
2. The absolute frequency of the feature in the positive data set is greater than $\lambda (0 < \lambda < 1)$, where λ will be fixed through experiments.

Our improved 1-DNF algorithm (as shown in Fig. 2) can be depicted as follows, where $|P|$ is the number of the documents in the positive set P, $|U|$ is the number of the documents in the unlabeled set U, and $freq(x_i, P)$ is the number of feature x_i occurring in the positive set P, $freq(x_i, U)$ is the number of feature x_i occurring in the unlabeled set U. We use our improved 1-DNF algorithm to identify the reliable negative set NEG_0.

Fig. 3 Algorithm for constructing the final classifier by using weighted voting method

```
Algorithm Building Classifiers
$PP = 10\% \times P$; $P = P - PP$; $PON = P \cup NEG_0$; $U=U-NEG_0$; $i = 0$; $allPrecision = 0$;

While ( true ) {
  $SVM_i$ = SVM_training ($PON$);
  $NEG_{i+1} = SVMClassify_i(U)$;
  $precision_i = SVMClassify_i$_predict($PP$)'s precision;
  $allPrecision = allPrecision + precision_i$;
  if ($NEG_{i+1}$= =NULL)
    Break;
  $PON = PON \cup NEG_{i+1}$;
  $U = U - NEG_{i+1}$;
  $i = i+1$;
};
$FinalWVClassifier = \sum_{i=0}^{d-1} \frac{precision_i}{allPrecision} SVMClassifier_i$

(The times of building sub-classifiers just are the number of classifiers that constitute the final classifier)
```

3.3 Building text classifiers by iteratively applying SVM

Unlike traditional approach that one specific classifier in the classifiers set generated during the iterative algorithm is designated as the final one, we make use of all of them to construct the final classifier based on voting method. Figure 3 describes the process of the constructing weighted voting classifier. Because we know nothing about the negative data in the PU problem, a new voting method is proposed. Using the improved 1-DNF algorithm (1-DNFII), we can obtain the more reliable negative document set NEG_0. Now the training sample set is $PON = P \cup NEG_0$, and the unlabeled sample set is $U = U - NEG_0$. We use the SVM algorithm on *PON* as the training set to construct the initial classifier SVM_0, and use SVM_0 to classify *PP* ($PP = 10\% \times P$)(the precision is $precision_1$) and U (the set of documents classified as negative is NEG_1). Then the training example set is increased to $PON = PON \cup NEG_1$, and unlabeled sample set is $U = U - NEG_1$. We use the training set *PON* to construct the second classifier SVM_1. SVM_1 is used to classify *PP* (the precision is $precision_2$) and U (the set of documents classified as negative is NEG_2). Then the training example set is increased to $PON = PON \cup NEG_2$, and the unlabeled set is $U = U - NEG_2$. This process iterates until no documents in U are classified as negative. Then we use the precision of each individual classifier to weight its corresponding classifier to construct the final WV Classifier.

3.4 Evolving the weights using PSO-based method

In the past several years, PSO has been proven to be both effective and quick to solve some optimization problems. It was successfully applied in many research and application areas [3,9]. Because the PSO algorithm can discover the best combination of the each individual classifier's weights, after building individual classifiers, we use the following function to construct another final text classifier based on PSO voting method.

$$FinalPSOClassifier = \sum_{i=0}^{d-1} \frac{\omega_i}{\sum_{i=0}^{d-1} \omega_i} SVMClassifier_i \tag{3}$$

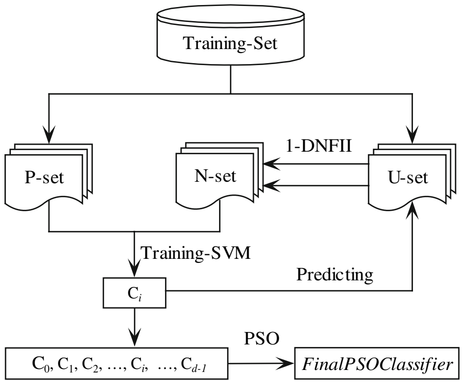


Fig. 4 Illustration of the procedure to build text classifiers from labeled and unlabeled examples based on PSO. C$_i$ represents the individual classifier produced by the ith iteration of the SVM algorithm

In this paper, it is possible to view the process of constructing the final classifier as an optimization problem that locates the optimal combination of the weights. This view offers us a chance to apply the PSO algorithm on the document categorization solution. Figure 4 describes the process of building text classifiers from labeled and unlabeled examples based on PSO.

In the PSO-based classification method, the multi-dimensional combination of sub-classifiers is modeled as a problem space. Each weight of sub-classifier in the combination represents one dimension of the problem space. In the PSO-based method, each individual is represented to determine a combination of sub-classifiers in the problem space. All of the combinations can be represented as a multidimensional space with a large number of particles in the space. Therefore, a swarm represents a number of candidate sub-classifier combinations. Each particle maintains a vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{id}), i = 1, 2, \ldots, m$, where x_{ij} represents the weight of corresponding sub-classifier and d is the number of dimension (the number of sub-classifiers).

According to its own experience and those of its neighbors, the particle adjusts the position in the vector space at each generation. We use F$_1$-Measure to determine the fitness value, which evaluates the combination represented by each particle. The fitness value is measured by the equation below:

$$fitness = F_1\text{-}Measure \tag{4}$$

As the F$_1$-Measure value increases as much as possible based on the guidance of the proposed fitness function, the classification system corresponding to the individual will satisfy the desired objective as well as possible. Subsequently, a PSO-based method is proposed to find an appropriate individual so that the corresponding classification system has the desired performance. The PSO-based procedure is described as follows:

Step 1 *Initialize the PSO-based method.*

- (a) *Set the number of individuals m, the number of dimensions d, the number of generations N, the constants for the PSO algorithm* (w, c_1, *and* c_2);
- (b) *Generate randomly initial population P. Generate randomly initial velocity vectors V.*

Step 2 *Calculate the fitness value of each individual based on Eq.* (4).

- (a) *If* $fitness_i > pBest_i$, *then set* $fitness_i = pBest_i$;
- (b) *If* $fitness_i > gBest$, *then set* $fitness_i = gBest$;

(c) *Using Eqs.* (1) *and* (2), *update the particle velocity and position to generate the next solutions.*

Step 3 *Repeat step* (2) *until the following termination condition is satisfied.*

(a) *The maximum number of iterations is exceeded.*

The behavior of the PSO-based method can be classed into two stages: a global searching stage and a local refining stage. At the initial iteration, based on the PSO algorithm's particle velocity updating Eq. (1), the particle's initial velocity v_{id}, the two randomly generated values (r_1, r_2) at each generation and the inertia weight factor w provide the necessary diversity to the particle swarm by changing the momentum of particles to avoid the stagnation of particles at the local optima. Multiple particles parallel searching, using multiple different solutions at a time, can explore more area in the problem space. The initial iteration can be classified as the global searching stage. After several iterations, the particle's velocity will gradually reduce and the particle's explore area will shrink while the particle will approach the optimal solution. The global searching stage gradually changes to the local refining stage. By selecting different parameters in the PSO algorithm, we can control the shift time from the global searching stage to the local refining stage. The later the particle shifts from the global searching stage to local refining stage, the greater the possibility that it can find the global optimal solution.

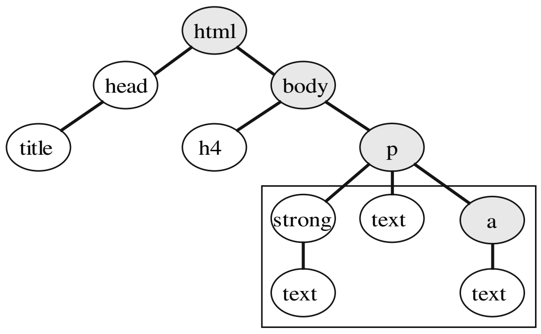
4 Building focused crawler guided by text classifier

With the rapid growth of information and the explosion of web pages from the complex WWW, it gets harder for search engines to retrieve the information relevant to a user. A web crawler is a program that retrieves web pages for a search engine, which is widely used today. Unlike general-purpose web crawlers that automatically traverse the web and collect all web pages, focused crawlers (also called topical crawlers) are designed to collect pages on specific topic, which try to "predict" whether or not a target URL is pointing to a relevant page before actually fetching the page. A focused crawler carefully decides which URLs to scan and in what order to pursue based on information about previously downloaded pages. To design an efficient focused crawler collecting relevant pages from the WWW, the choice of strategy for prioritizing unvisited URLs is crucial. In this section, we present a focused web crawling method to utilize link-context for determining the priorities guided by text classifiers. The link-context is a kind of extended anchor text. Anchor text is the "highlighted clickable text" in source code, which appears within the bounds of an <A> tag. Since the link-context tends to summarize information about the target page, we pursue the intuition that it is a good provider of the context of the unvisited URLs. By applying it in prioritizing the unvisited URLs and guiding the crawling, the crawler's performance will be maximized.

We use the method of deriving link-context from Aggregation Nodes, with some modifications. We tidy the HTML page into a well-formed web page beforehand (http://www.w3.org/People/Raggett/tidy/) because many web pages are badly written. We insert missing tags and reorder tags in the "dirty" page to make sure that we can map the context onto a tree structure with each node having a single parent and all text tokens that are enclosed between < text > ... < /text > tags, which are inserted by the "tidy" program, appear at leaf nodes on the tag tree. This preprocessing simplifies the analysis. Figure 5 illustrates a snippet of an HTML page and the corresponding tag tree. First, we locate the anchor. Next, we treat each node on the path from the root of the tree to the anchor as a potential aggregation node (shaded in Fig. 5). From these candidate nodes, we choose the parent node of anchor, which

Fig. 5 An HTML page and the corresponding tag tree

```
<html>
<head>
<title>Illustration</title>
</head>
<body>
<h4>Artificial Intelligence</h4>
<p>
<strong>
<text>SITE LISTINGS</text>
</strong>
<text>provides an overview of progress in the field of
artificial intelligence, focusing on the research and
development of novel computing hardware which
functions in ways similar to natural nervous systems.
</text>
<a href="http://www.artificialbrains.com/">
<text>ArtificialBrains.com<text>
</a>
</p>
</body>
</html>
```



is the grandparent node of the anchor text, as the aggregation node. Then, all of the text in the sub-tree rooted at the node is retrieved as the context of the anchor (showed in rectangle of Fig. 5). If there are several anchors, which appear in many different blocks, referring to the same URL, combine the link-context in every block as its full link-context. In fact, for each page we have an optimal aggregation node that provides the highest context similarity for the corresponding anchor with different aggregation node. It is very laborsome to tidy up web pages every time we analyze the web pages. Large size contexts may be too "noisy" and burdensome for some systems. Too small contexts, like single anchor texts, provide limited information about the topic. Accordingly, fixing an optimal aggregation node must set a tradeoff on quantity and quality, which needs more experiments. In this paper, we choose the parent node of anchor, which is the grandparent node of the anchor text, as the aggregation node. Link-context derived by simply choosing the parent of the anchor. We will later look into the possibility of choosing some other potential aggregation node.

In the process of crawling, whenever a web page is downloaded, we pick up all URLs and anchors. If the page linked by the URL has been crawled or the URL is in the queue of crawling, the URL is omitted. Then we parse the page's DOM (Document Object Model) tree to extract link-context, compute the similarity with the term-based features using crawling classifier. Then the similarity is treated as a priority. The unvisited URL that has the highest

Fig. 6 Procedure of focused web crawling based on link-context guided by crawling classifier

```
Focused Web Crawling Procedure Based On Link-Context
  Input: starting_urls: seed URLs
  Procedure:
  1.  enqueue(url_queue, starting_url)
  2.  while (not empty(url_queue))
  3.    url = dequeue(url_queue)
  4.    page = crawl_page(url)
  5.    enqueue(crawled_queue, url)
  6.    LinkContext_list = Extract_Context(page)
  7.    for each Link-context in LinkContext_list
  8.      for each u in Link-context
  9.        prioritize u by Classify(Link-context )
  10.       add u to url_list
  11.   for each u in url_list
  12.     if ( $u \notin url\_queue$ and $u \notin crawled\_queue$ )
  13.     enqueue(url_queue, u)
  14.     reorder_queue(url_queue)

Key function description:
enqueue(queue,element):append element at the end of queue
dequeue(queue):       remove the element at the beginning of
                          queue and return it
reorder_queue(queue): reorder queue by link's priority
```

priority will be first fetched to crawl. Whenever a new batch of URLs is inserted into the waiting queue, the queue will be readjusted to create its new frontier. After parsing, the pages will be classified by a conventional classifier. Some "good" relevant pages' in-link contexts will be preprocessed (eliminating *stopwords* and stemming) and introduced to the training set. Figure 6 describes our focused web crawling procedure.

5 Experiments and results

5.1 Text classification

In the experiment, we used the Reuters-21,578 dataset, which has 21,578 documents collected from the Reuters newswire, as our training sample set. Of the 135 categories in Reuters 21,578, only the most populous 10 are used. In data pre-processing, we applied *stopword* removal and *tfc* (function 5) [25] feature selection, and removed the commoner morphological and inflexional endings from words using Porter Stemming Algorithm (http://www.tartarus.org/~martin/PorterStemmer/). Each category is employed as the positive class, and the rest as the negative class. For each category, 30% of the documents are randomly selected as test documents, and the rest are used to create the training sets as follows: γ of the documents from the positive class are first selected as the positive set P. The rest of the positive documents $(1 - \gamma)$ and negative documents are used as the unlabeled set U. We range γ from 0.1 to 0.5 to create a wide range of scenarios. The experimental results showed in this paper are the average on different γ setting.

$$a_{ik} = \frac{f_{ik} * \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{j=1}^{M}\left[f_{jk} * \log\left(\frac{N}{n_j}\right)\right]^2}}. \tag{5}$$

where, f_{ik} is the frequency of word i in document k, N is the number of documents in the collection, n_i is the total number of times word i occurs in the whole collection.

In our experiment we used LIBSVM (version 2.71), an integrated tool for support vector classification, which can be downloaded at http://www.csie.ntu.edu.tw/~cjlin/libsvm/. We used the standard parameters of the SVM algorithm in one-class SVM classifier, PEBL classifier and our two classifiers (Weighted voting classifier and PSO voting classifier).

To evaluate our final classifiers, we use the F_1-it Measure, which is a commonly used performance measure for text classification. This measure combines *precision* and *recall* in the following way:

$$Precision = \frac{\#\, of\ correct\ positive\ predictions}{\#\, of\ positive\ predictions}. \tag{6}$$

$$Recall = \frac{\#\, of\ correct\ positive\ predictions}{\#\, of\ positive\ examples}. \tag{7}$$

$$F_1\text{-}Measure = \frac{2 * precision * recall}{precision + recall}. \tag{8}$$

For comparing the performance of the classifiers based on different techniques, we implemented the PEBL algorithm, one-class SVM algorithm and our two classifiers (weighted voting classifier, PSO voting classifier) in the experiments. In the step of identifying reliable negative documents from the unlabeled set U by the improved 1-DNF algorithm, we ranged λ from 0.10 to 0.90, an selected λ which results in best performance as the final value.

At first, we compared the improved 1-DNF algorithm with the 1-DNF algorithm in the number of identifying reliable negative documents and the error rate. Let *RN* be the number of the reliable negative documents, the ERR(%) is calculated as follows:

$$ERR(\%) = \frac{Number\ of\ the\ positive\ data\ in\ the\ strong\ negative\ data\ set}{Number\ of\ all\ positive\ data\ mixed\ in\ the\ unlabeled\ data\ set}. \tag{9}$$

Table 1 shows the average number of reliable negative documents and the error rate when $\lambda = 0.10$–0.90, and Fig. 7 compares the number of reliable negative documents between 1-DNF and 1-DNFII from $\lambda = 0.10$–0.90, respectively. Results show that the number of the reliable negative documents identified by the improved 1-DNF algorithm is more than that identified by the original 1-DNF. Computing the average of reliable negative data of the ten categories, we found that the numbers of reliable negative documents identified by the improved 1-DNF algorithm are 4.42, 7.6, 8.63 and 9.82 times greater than that identified by the original 1-DNF at $\lambda = 0.10$, 0.20, 0.30 and 0.40 setting, respectively. At the same time, as shown in Fig. 7, the error rates of identifying the positive data as negative are 0.19, 1.30, 2.27 and 3.83%, respectively. So the comparisons indicate that the improved 1-DNF algorithm can identify more reliable negative documents with a low error rate.

Table 2 shows the average times of building sub-classifiers iteratively on $\gamma = 0.10$–0.50 setting. Fig. 8 shows the average times of building sub-classifiers iteratively on the ten categories. As shown in Table 2, all of the values of 1-DNFII are less than 1-DNF for each λ setting. That is, the time for our classifier training is less than PEBL and decreases as λ increases. This is because the greater λ means the smaller *PF*, which means the greater NEG_0, so that the algorithm of Fig. 3 requires fewer iterations to stop. The number of the times of building sub-classifiers just is the number of classifiers that constitute the final classifier.

Building PSO-based voting classifier, we set the maximal times of iteration $N = 500$, the population size $m = 30$, the inertia coefficient $w = 0.8$. Usually $c_1 = c_2 = 2$ is chosen to take the same weight. The performance of the WVC (Weighted Voting Classifier), PEBL and OCS (one-class SVM) is shown in Table 3. Table 4 illustrates the performance of the PSOC (PSO Voting Classifier), PEBL and OCS (one-class SVM). For comparing the performance

Table 1 The average number of reliable negative documents and the error rate

λ		Acq	Corn	Crude	Earn	Grain	Interest	Money	Ship	Trade	Wheat
1-DNF	RN	161.0	203.4	116.6	269.6	128.0	299.0	269.6	324.4	70.8	200.2
	ERR(%)	0.00	0.00	0.00	0.02	0.00	0.00	0.00	0.33	0.00	0.00
0.10	RN	792.4	802.2	480.2	787.0	890.8	1250.0	1308.6	1356.4	383.4	975.6
	ERR(%)	0.98	0.00	0.22	0.26	0.00	0.11	0.00	0.33	0.00	0.00
0.20	RN	1,003.8	1,343.0	1,189.6	1,200.4	1,570.4	2,185.2	2,087.2	2,128.2	1,034.2	1,791.0
	ERR(%)	2.52	0.00	0.63	0.66	0.91	2.88	2.67	2.28	0.44	0.00
0.30	RN	1,115.4	1,577.8	1,296.6	1,689.0	1,660.0	2,267.6	2,245.2	2,600.8	1,164.6	2,017.0
	ERR(%)	4.07	0.00	0.63	3.22	0.91	3.32	5.19	4.51	0.90	0.00
0.40	RN	1,151.8	1,731.4	1,678.8	1,706.8	2,523.6	2,330.6	2,393.0	2,680.8	1,305.8	2,562.0
	ERR(%)	4.51	0.48	0.63	3.35	2.26	3.32	7.83	12.50	3.24	0.25
0.50	RN	1,603.0	2,570.4	2,599.8	2,622.8	2,535.0	2,356.8	2,491.6	2,683.2	1,813.8	2,568.2
	ERR(%)	6.77	0.48	4.53	5.60	7.59	4.52	9.93	13.10	3.58	0.25
0.60	RN	2,593.6	2,586.4	2,601.6	2,636.6	2,557.8	2,392.2	2,560.8	2,683.2	2,635.8	2,589.0
	ERR(%)	9.37	2.64	6.34	5.91	19.10	4.67	13.18	13.10	3.58	0.25
0.70	RN	2,593.6	2,619.6	2,601.6	4,198.8	2,557.8	2,478.4	2,629.8	2,683.2	2,635.8	2,617.2
	ERR(%)	9.37	16.91	6.34	15.93	19.10	7.31	17.18	13.10	3.58	0.25
0.80	RN	2,593.6	5,844.6	2,601.6	5,023.2	6,314.0	5,482.6	2,629.8	2,683.2	2,635.8	2,617.2
	ERR(%)	9.37	85.75	6.34	100.00	100.00	70.23	17.18	13.10	3.58	0.25
0.90	RN	4,543.2	6,652.4	6,246.4	5,023.2	6,314.0	6,622.2	6,481.2	6,755.0	5,517.2	6,476.2
	ERR(%)	64.14	100.00	10.13	100.00	100.00	100.00	100.00	100.00	6.52	1.50

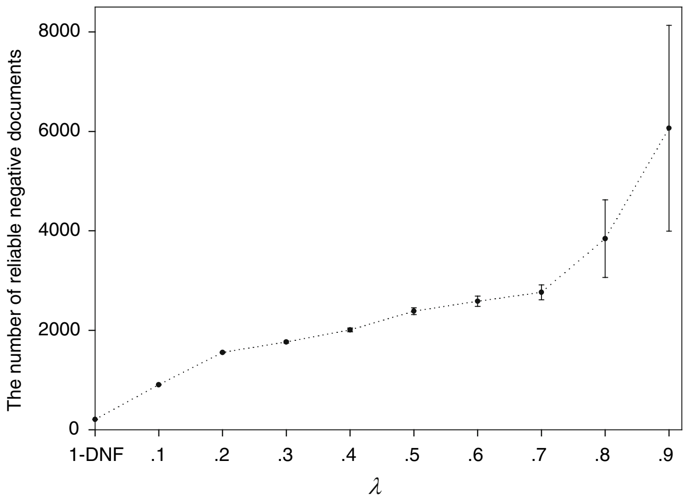


Fig. 7 The average number of reliable negative documents obtained by 1-DNF and 1-DNFII for each λ setting on ten categories. The error bars correspond to the number of positive documents in the reliable negative documents

Table 2 Average times of building sub-classifiers iteratively on $\gamma = 0.10$–0.50 setting

λ	Acq	Corn	Crude	Earn	Grain	Interest	Money	Ship	Trade	Wheat	Average
OCS	12.1	9.5	15.2	11.0	12.7	9.6	14.2	9.3	12.8	11.5	10.7
PEBL	9.8	7.2	9.4	7.4	7.8	7.6	10.4	8.6	9.0	9.0	8.62
0.10	8.0	6.8	9.2	6.6	6.8	6.6	9.2	7.2	7.6	8.0	7.6
0.20	8.8	6.2	7.6	7.2	6.6	6.6	10.4	6.8	6.6	7.0	7.38
0.30	8.8	6.2	7.6	6.2	6.0	6.4	10.2	6.8	6.2	7.0	7.14
0.40	8.4	6.0	7.4	6.4	6.0	6.4	9.4	5.6	6.6	7.8	7
0.50	9.0	6.0	7.2	6.6	6.6	6.4	8.4	5.0	6.2	6.2	6.76
0.60	8.4	5.6	7.8	6.2	5.8	6.4	8.8	5.0	6.6	6.4	6.7
0.70	8.4	5.6	7.8	2.6	5.8	5.6	9.2	5.0	6.6	6.0	6.26
0.80	8.4	2.2	7.8	1.0	1.0	2.4	9.2	5.0	6.6	6.0	4.96
0.90	4.2	1.0	8.6	1.0	1.0	1.0	1.0	1.0	6.0	5.0	2.98

of the classifiers based on different techniques clearly, average F_1-Measure performance of OCS, PEBL, WVC and PSOC on the ten categories is shown in Fig. 9. We observed that the performance of our WVC outperforms PEBL for $\lambda = 0.10$–0.60 setting by 1.73, 5.026, 4.98, 3.60, 2.49 and 0.76%, respectively, and outperforms OCS even more for $\lambda = 0.10$–0.90 setting. The F_1-Measure performance of PSO-based method, which was presented in this paper, outperforms WVC for each λ setting by 6.544, 6.23, 6.424, 6.542, 6.689, 7.059, 7.457, 8.115 and 9.467%, respectively, as shown in Fig. 9. The results in Fig. 9 indicates that F_1-Measure performance of PSOC and WVC are best and higher than PEBL by 11.256 and 5.026% for $\lambda = 0.20$ setting, respectively. Therefore, Figure 10 illustrates the performance of the four classification methods for $\lambda = 0.20$ setting. The corresponding number of reliable negative documents and the error rate are also shown in Table 5.

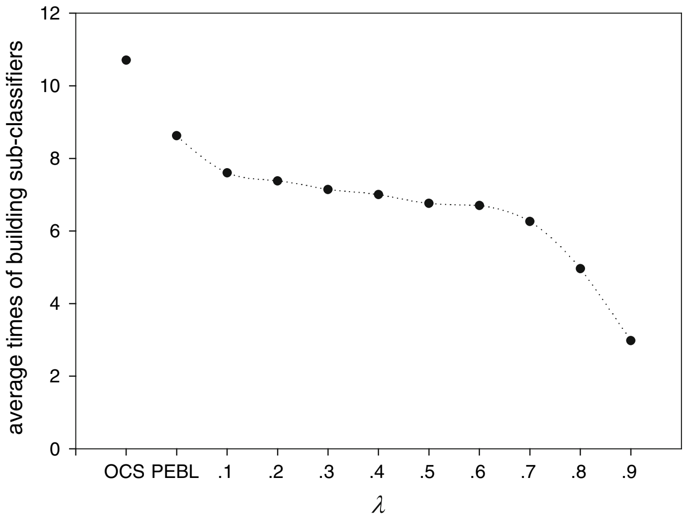


Fig. 8 The average times of building sub-classifiers iteratively on the ten categories

Figure 11 shows average performance comparisons of WVC and PSOC on $\lambda = 0.10$–0.90 setting for each category dataset. The result indicates that the F_1-Measure of PSOC on each category dataset is higher than WVC.

The experimental results from above prove that the PSO-based text classification method can obtain a higher F_1-Measure performance compared with other three methods. The results also prove that the performance of the classifier constructed only using the positive set is poorer than that takes advantage of the unlabeled data.

5.2 Focused crawling based on link-context guided by classifiers

In this section, we built focused crawlers that use different classifiers for crawling the web, and tested our method using multiple crawls over 37 topics covering hundreds of thousands of pages. Under the guide of each method, crawler downloaded the pages judged as relevant. We used a conventional classifier to decide whether the page is relevant. *Harvest rate* and *Target recall* are used to evaluate the results. Our focused crawler is multi-threaded and implemented in java. Multithreading provides for reasonable speed-up when compared with a sequential crawler. We use 50 threads of execution starting from 100 relevant URLs (Seed URLs) on each topic picked from Open Directory Project (ODP, http://dmoz.org/) while running a crawler.

The ODP is a categorical directory of URLs that is manually edited and relatively unbiased by commercial motivations. The ODP provides the data contained in its directory in RDF format through its Web site. We first downloaded the RDF formatted content file from the ODP Web site. The content file contains a list of ODP categories and the external URLs or ODP relevant set corresponding to each category. We treat ODP categories as potential topics for our crawling experiments. The ODP relevant set consists of URLs that have been judged relevant to the category by human editors of ODP. We randomly select 37 categories and the associated ODP relevant sets. These selected categories will serve as topics for our crawling experiments. We further divide the ODP relevant set for a selected topic into two random disjoint subsets. The first set is the seeds (contain 100 URLs), which will be used to initialize the crawl.

Table 3 The F_1-Measure performance of OCS, PEBL and WVC for each λ setting on the ten categories

λ	PEBL	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	OCS
Acq	0.9319	0.9501	0.9648	0.9500	0.9390	0.9180	0.9030	0.9030	0.9030	0.7890	0.7103
Corn	0.7207	0.7350	0.7976	0.8200	0.7690	0.7750	0.7410	0.6600	0.5970	0.4298	0.4298
Crude	0.8367	0.8580	0.8891	0.8930	0.9000	0.8850	0.8650	0.8650	0.8650	0.7850	0.6759
Earn	0.9701	0.9792	0.9813	0.9772	0.9750	0.9640	0.9510	0.9310	0.8997	0.8997	0.8997
Grain	0.9101	0.9178	0.9416	0.9400	0.9270	0.9160	0.8660	0.8660	0.6591	0.6591	0.6591
Interest	0.8099	0.8207	0.8528	0.8520	0.8570	0.8410	0.8280	0.8020	0.7660	0.7036	0.7036
Money	0.8357	0.8640	0.8856	0.8620	0.8540	0.8460	0.8020	0.7870	0.7870	0.6972	0.6972
Ship	0.7641	0.7860	0.8767	0.8560	0.8300	0.8080	0.8080	0.8080	0.8080	0.4319	0.4319
Trade	0.8802	0.8903	0.9136	0.9020	0.8920	0.8770	0.8900	0.8900	0.8900	0.7310	0.6798
Wheat	0.7601	0.7918	0.8571	0.8650	0.8360	0.8380	0.8410	0.8420	0.8420	0.8110	0.5612
Average	0.8420	0.8593	0.8960	0.8917	0.8779	0.8668	0.8495	0.8354	0.8017	0.6937	0.6449

Table 4 The F_1-Measure performance of OCS, PEBL and PSOC for each λ setting on the ten categories

λ	PEBL	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	OCS
Acq	0.9319	0.9793	0.9940	0.9912	0.9802	0.9592	0.9507	0.9563	0.9599	0.8669	0.7103
Corn	0.7207	0.8070	0.8696	0.8960	0.8480	0.8597	0.8257	0.7600	0.7120	0.5698	0.4298
Crude	0.8367	0.9100	0.9411	0.9481	0.9551	0.9401	0.9201	0.9222	0.9278	0.8635	0.6759
Earn	0.9701	0.9927	0.9948	0.9952	0.9930	0.9825	0.9695	0.9506	0.9331	0.9304	0.8997
Grain	0.9101	0.9711	0.9949	0.9933	0.9803	0.9726	0.9282	0.9310	0.7317	0.7391	0.6591
Interest	0.8099	0.9047	0.9299	0.9248	0.9298	0.9138	0.9008	0.8860	0.8589	0.8015	0.7036
Money	0.8357	0.9492	0.9650	0.9380	0.9300	0.9220	0.8934	0.8784	0.8897	0.8056	0.6972
Ship	0.7641	0.8780	0.9607	0.9435	0.9263	0.9043	0.9043	0.9043	0.9043	0.5612	0.4319
Trade	0.8802	0.9743	0.9976	0.9860	0.9760	0.9662	0.9792	0.9786	0.9786	0.8390	0.6798
Wheat	0.7601	0.8810	0.9356	0.9435	0.9145	0.9165	0.9290	0.9323	0.9323	0.9070	0.5612
Average	0.8420	0.9247	0.9583	0.9560	0.9433	0.9337	0.9201	0.9100	0.8828	0.7884	0.6449

The result of PSOC is obtained under the condition that $N = 500$, $m = 30$, $w = 0.8$ and $c_1 = c_2 = 2$

The seeds also serve as the initial positive example set extracted from in-link's link-context for training the classifiers. The second set contains the training sets for building classifier, which guides focused web crawling. In data pre-processing, we applied *stopword* removal and *tfc* feature selection, and removed the commoner morphological and inflexional endings from words using The Porter Stemming Algorithm (http://www.tartarus.org/∼martin/PorterStemmer/).

The output of a crawler is a temporal sequence of pages crawled. Any evaluation of crawler performance is hence based on this output. With this knowledge, we could estimate the *precision* and *recall* of a crawler after crawling n pages. The *precision* would be the fraction of pages crawled that are relevant to the topic and *recall* would be the fraction of relevant pages crawled. However, the relevant set for any given topic is unknown in the web, so the true recall is hard to measure. In the experiments, *harvest rate* will be used as an estimate of precision and *target recall* will be used as an estimate of recall.

Harvest rate. *Harvest rate* (function 10) estimates the fraction of crawled pages that are relevant to a given topic. We make this decision by using a set of SVM evaluation classifiers

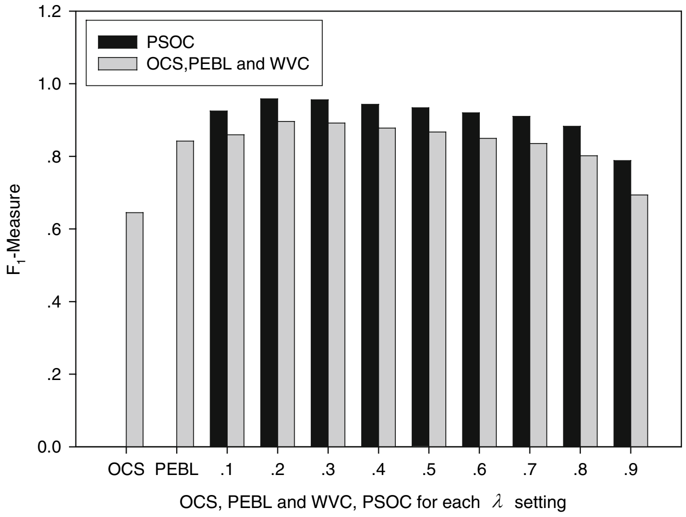


Fig. 9 Average performance comparisons of OCS, PEBL, WVC and PSOC on the ten categories

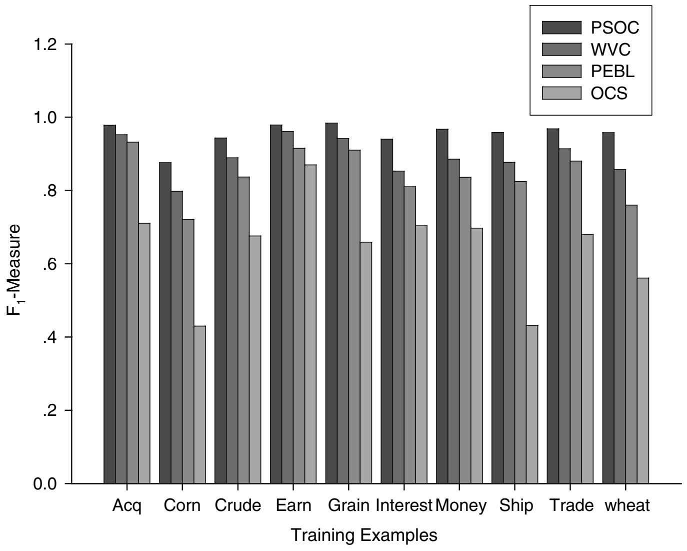


Fig. 10 The performance of the four classification methods for $\lambda = 0.20$ setting, which the performance of the final classifier is best

instead of manual relevance judgment, which is costly. The SVM evaluation classifiers are trained on a larger set from ODP, which are more "informed" about the topic. To explain, for each topic, we take one classifier at a time and train it using the pages corresponding to the entire ODP relevant set (instead of just the seeds) as the positive examples. The negative examples are obtained from the ODP relevant sets of the other topics. The negative examples are again twice as many as the positive examples.

$$harvest\_rate = \frac{relevant\_pages}{pages\_downloaded}. \tag{10}$$

Table 5 The number of reliable negative documents and the error rate ($\lambda = 0.20$)

	Acq	Corn	Crude	Earn	Grain
Improved 1-DNF					
RN	1,003.8	1,343.0	1,189.6	1,200.4	1,570.4
ERR(%)	2.52	0.00	0.63	0.66	0.91
1-DNF					
RN	161.0	203.4	116.6	269.6	128.0
ERR(%)	0.00	0.00	0.00	0.02	0.00
	Interest	Money	Ship	Trade	Wheat
Improved 1-DNF					
RN	2,185.2	2,087.2	2,128.2	1,034.2	1,791.0
ERR(%)	2.88	2.67	2.28	0.44	0.00
1-DNF					
RN	299.0	269.6	324.4	70.8	200.2
ERR(%)	0.00	0.00	0.33	0.00	0.00

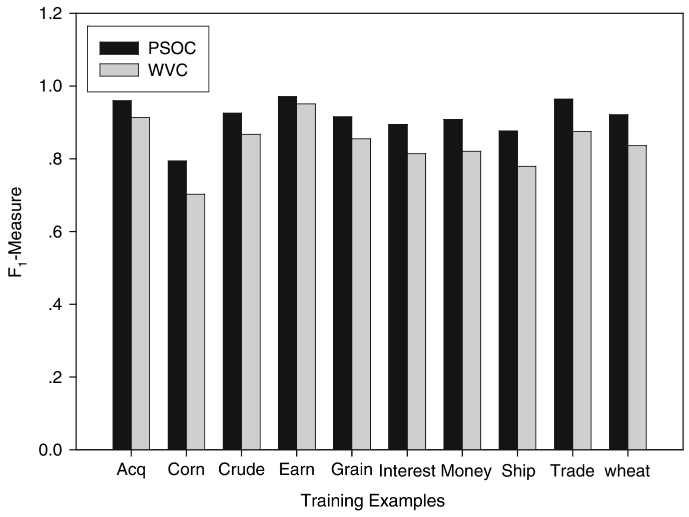


Fig. 11 Average performance comparisons of WVC and PSOC on $\lambda = 0.10–0.90$ setting

Target recall. *Target recall* is an estimate of the fraction of relevant pages that are fetched by a crawler. As described earlier, true recall is hard to measure since we cannot identify the true relevant set for any topic over the web. So in the experiments, we treat the recall of the target set, i.e., *target recall*, as an estimate of true recall for different techniques comparing. If targets are a random sample of the relevant pages on the Web, then we can expect target recall to give us a good estimate of the actual recall. We assume that the target set T is a random sample of the relevant set R. Therefore, for recall, $\frac{|C(t) \cap R|}{|R|} \approx \frac{|C(t) \cap T|}{|T|}$. The *target recall* [24], $R(t)$, after first crawling t pages for a given topic is computed as:

$$R(t) = \frac{|C(t) \cap T|}{|T|}. \tag{11}$$

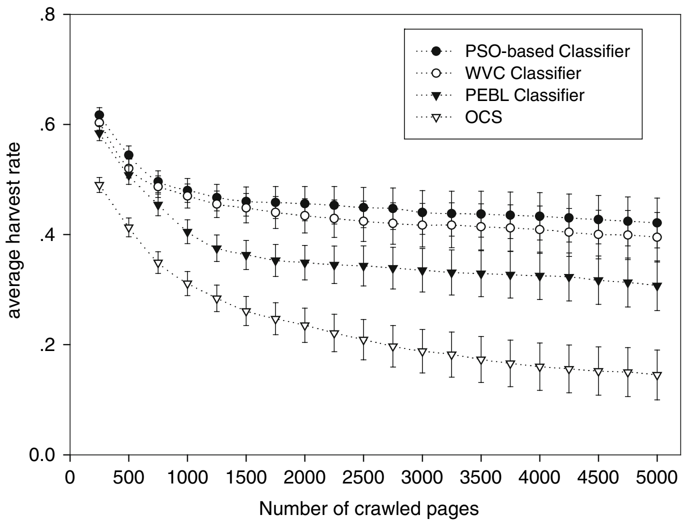


Fig. 12 Dynamic plot of average *harvest rate* versus number of crawled pages. Performance is averaged across topics and standard errors are also shown. The *error bars* correspond to ±standard error

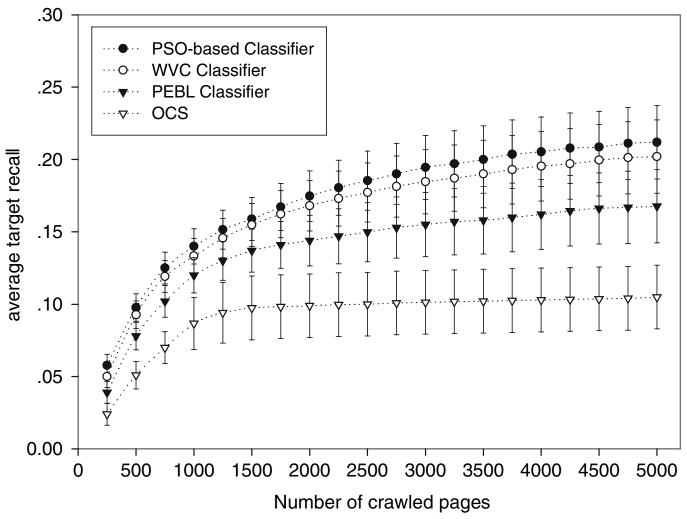


Fig. 13 Dynamic plot of average *target recall* versus number of crawled pages. Performance is averaged across topics and standard errors are also shown. The *error bars* correspond to ±standard error

where $C(t)$ is the set of first t pages crawled, T is the set of targets, and $|T|$ is the number of targets.

Figures 12 and 13 plot average *harvest rate* and *target recall* of the focused crawlers guided by different classifiers. Focused crawlers predict the relevance of the potential URLs by referring to the link-context of the visited web page. We find that the focused crawler guided by PSO-based classifier shows significant performance improvement over the focused crawlers guided by other classifiers. Consequently, the experimental results are we expected.

6 Conclusion

This paper studied the PU-oriented text classification problem. A new algorithm based on the improved 1-DNF and weighted voting method to solve the PU classification problem in the text domain is proposed. Experimental results draw four important conclusions: firstly, compared with the 1-DNF algorithm, the improved 1-DNF can obtain more negative data with a lower error rate. Secondly, the performance of the classifiers that discard unlabeled data set and learn only from positive data set (one-class SVM algorithm) is much poorer than the classifiers that take advantage of the unlabeled data set. Thirdly, applying the weighted voting method to the PU-oriented text classification can increase the performance of the classifier. Lastly, compared the weighted voting method, the performance of the PSOC is better appreciably.

We also built a focused crawler that based on link-contexts guided by different classifiers to crawl the web, and evaluate our method using multiple crawls over 37 topics covering hundreds of thousands of pages. Experimental results show that the performance of the focused crawler guided by PSO-based classifier outperforms other classifiers.

Acknowledgments This work is sponsored by the national Natural Science Foundation of China (NSFC) under grant number 60373099.

References

1. Bing L, Wee SL, Philip S Yu, Xiaoli L (2002) Partially supervised classification of text documents. The nineteenth international conference on machine learning (ICML), Sydney, Australia, pp 384–397
2. Bing L, Yang D, Xiaoli L, Wee SL, Philip S Yu (2003) Building text classifiers using positive and unlabeled examples. In: Proceedings of the 3rd IEEE international conference on data mining (ICDM 2003), Melbourne, Florida, USA, pp 179–188
3. Carlisle A, Dozier G (2001) An Off-The-Shelf PSO. In: Proceedings of the workshop on particle swarm optimization, Indianapolis, pp 1–6
4. Craven M, DiPasquo D, Freitag D, McCallum A, Mitchell T, Nigam K, Slattery S (1998) Learning to extract symbolic knowledge from the World Wide Web. In: Proceedings of the fifteenth national conference on artificial intellligence (AAAI-98), Madison, USA, pp 509–516
5. De Falco I, Della Cioppa A, Iazzetta A, Tarantino E (2005) An evolutionary approach for automatically extracting intelligible classification rules. Knowl Inf Syst 7(2):179–201
6. Denis F (1998) PAC learning from positive statistical queries. In: Proceedings of the 9th international conference on algorithmic learning theory. Lecture Notes in Artificial Intelligence. vol 1501, Springer, Heidelberg, pp 112–126
7. Denis F, Gilleron R, Tommasi M (2002) Text classification from positive and unlabeled examples. Conference on information processing and management of uncertainty in knowledge-based systems (IPMU), Annecy, France, pp 1927–1934
8. DeComite F, Denis F, Gilleron R (1999) Positive and unlabeled examples help learning. In: Proceedings of the 10th international conference on algorithmic learning theory, Tokyo, Japan, pp 219–230
9. Eberhart RC, Shi Y (2000) Comparing inertia weights and constriction factors in particle swarm optimization. In: Proceedings of the 2000 congress on evolutionary computation. Washington, DC, vol 1, pp 84–88
10. Hwanjo Y, Jiawei H, Kevin Chen-Chuan Chang (2002) PEBL: Positive example based learning for Web page classification using SVM. In: Proceedings 8th international conference on knowledge discovery and data mining (KDD'02), Edmonton, Canada, pp 239–248
11. Han ES, Karypis G, Kumar V (2001) Text categorization using weight adjusted k-nearest neighbor classification. In: Proceedings of the 5th Pacific-Asia conference on knowledge discovery and data mining, Hong Kong, pp 53–65
12. Kennedy J, Eberhart R (1995) Particle swarm optimization, IEEE International Conference on Neural Networks, Perth, Australia, vol 4, 1942–1948
13. Lin WY, Kuo IC (2004) A genetic selection algorithm for OLAP data cubes. Knowl Inf Syst 6(1):83–102

14. Larry MM, MalikYousef (2001) One-Class SVMs for document classification. J Mach Learn Res 2:139–154
15. Lewis DD, Gale WA (1994) A sequential algorithm for training text classifiers. In: SIGIR '94: Proceedings of the seventeenth annual international ACM SIGIR conference on research and development in information retrieval, Dublin, Ireland, pp 3–12
16. Lang K (1995) NewsWeeder: Learning to Filter Netnews. In: Machine Learning: Proceedings of the twelfth international conference (ICML '95), San Francisco, CA, USA, pp 331–339
17. Levis D, Ringuette M (1994) A comparison of two learning algorithms for text categorization. In: Third annual symposium on document analysis and information retrieval, Las Vegas, US, pp 81–93
18. Letouzey F, Denis F, Gilleron R (2000) Learning from positive and unlabeled examples. In: Proceedings of the 11th international conference on algorithmic learning theory, Sydney, Australia, pp 71–85
19. Mukherjea S (2004) Discovering and analyzing World Wide Web collections. Knowl Inf Syst 6(2):230–241
20. Muslea I, Minton S, Knoblock CA (2002) Active + semi-supervised learning = robust multi-view learning. In: Proceedings of the nineteenth international conference on machine learning, Morgan Kaufmann Publishers Inc, San Francisco, USA, pp 435–442
21. Merwe VD, Engelbrecht AP (2003) Data clustering using particle swarm optimisation. In: Proceedings of IEEE congress on evolutionary computation (CEC 2003), Canbella, Australia, pp 215–220
22. Omran M, Salman A, Engelbrecht AP (2002) Image classification using particle swarm optimization. In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (SEAL 2002), Singapore, pp 370–374
23. Pazzani MJ, Muramatsu J, Billsus D (1996) Syskill & Webert: Identifying interesting Web sites. In Proceedings of the thirteenth national conference on artificial intelligence (AAAI-96), Portland, USA. AAAI Press/MIT Press, Cambridge, MA, pp 54–61
24. Srinivasan P, Menczer F, Pant G (2005) A general evaluation framework for topical crawlers. Inf Retr 8(3):417–447
25. Salton G, Buckley C (1988) Term weighting approaches in automatic text retrieval. Inf Process Manage 24(5):513–523
26. Thorsten Joachims (1998) Text Categorization with support vector machines: learning with many relevant features. In: Proceedings of ECML-98, 10th European conference on machine learning, Heidelberg, Germany, pp 137–142
27. Xiaoli L, Bing L (2003) Learning to classify text using positive and unlabeled data. In: Proceedings of eighteenth international joint conference on artificial intelligence (IJCAI-03), Acapulco, Mexico, pp 587–594
28. Yang Y, Pedersen JP (1997) Feature selection in statistical learning of text categorization. In: Proceedings of the fourteenth international conference on machine learning, Nashville, Tennessee, USA, pp 412–420
29. Zhihua Z, Shifu C, Zhaoqian C (2000) FANNC: A fast adaptive neural network classifier. Knowl Inf Syst 2(1):115–129

Authors Biography

Tao Peng received his PhD and MSc in computer science from the Jilin University in 2007 and 2004, respectively. He is currently a lecturer in the College of Computer Science and Technology, Jilin University, China. His research interests include data mining, web mining, and machine learning.

Wanli Zuo is currently a professor of computer science in the College of Computer Science and Technology, Jilin University. His research interests include database theory, data mining, web mining, machine learning, and web search engine.

Fengling He is currently a professor of computer science in the College of Computer Science and Technology, Jilin University. His research interests include data mining, web mining, and machine learning.