Knowledge-Based Systems 51 (2013) 15–26

Contents lists available at SciVerse ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Focused crawling enhanced by CBP–SLC

Tao Peng, Lu Liu *

College of Computer Science and Technology, Jilin University, Changchun 130012, China
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

ARTICLE INFO

Article history:
Received 1 January 2013
Received in revised form 24 May 2013
Accepted 13 June 2013
Available online 12 July 2013

Keywords:
Focused crawling
DOM tree
TFIPNDF
CBP–SLC
WVC
Tunneling

ABSTRACT

The complexity of Web information environments and multiple-topic Web pages are negative factors significantly affecting the performance of focused crawling. In a Web page, anchors or some link-contexts may misguide focused crawling, and a highly relevant region also may be obscured owing to the low overall relevance of that page. So, partitioning Web pages into smaller blocks will significantly improve the performance. In view of above, this paper presents a heuristic-based approach, CBP–SLC (Content Block Partition–Selective Link Context), which combines Web page partition algorithm and selectively uses link-context according to the relevance of content blocks, to enhance focused Web crawling. For guiding crawler, we build a weighted voting classifier by iteratively applying the SVM algorithm based on a novel TFIDF-improved feature weighting approach. During classifying, an improved 1-DNF algorithm, called 1-DNFC, is also proposed aimed at identifying more reliable negative documents from the unlabeled examples set. Experimental results show that the performance of the classifier using TFIPNDF outperforms TFIDF, and our crawler outperforms Breadth-First, Best-First, Anchor Text Only, Link-context, SLC and CBP both in *Harvest rate* and *Target recall*, which indicate our new techniques are efficient and feasible.

1. Introduction

With the rapid growth of information and the explosion of electronic text from the complex World Wide Web, more and more knowledge you need is included. But, the massive amount of text also takes so much trouble to people for finding useful information. For example, users often visit a search engine site to find relevant information. But, the results may be mixed with a great deal of irrelevant Web pages which include a lot of noise. The main reason of above problem is the WWW information overload. Therefore, an appropriate method to obtain relevant information can solve the problem effectively. Focused crawlers [5,30] (also called topical crawlers) are designed to collect pages on specific topic, which are widely used today. Focused crawlers carefully decide which URLs to scan and in what order to pursue based on previous downloaded pages information, which rely on the fact that pages about a topic tend to have links to other pages on the same topic. Focused crawlers have been used in a variety of applications such as search engines [16], competitive intelligence [6], and digital libraries [33]. They allow a higher-level application to gather from the Web, a focused collection rich in information about a topic or theme.

To design an efficient focused crawler collecting relevant pages from the WWW, the choice of strategy for prioritizing unvisited URLs is crucial. In general, texts and links about the same topic are usually grouped into one region in a Web page. Motivated by this, we partition Web pages into some content blocks in this paper. Our solution approach works with the DOM (Document Object Model) trees, rather than raw HTML markup. Instead of treating a whole Web page as a unit of relevance calculating, we evaluate each content block, respectively. Anchor text is a key component of the Web page. It is a powerful navigation for people browsing the WWW, and it also helps search engines understand the relationship among Web pages. But, anchor text is not informative enough, so, it sometimes needs to be expanded to link-context. Link-context is the critical contextual information of anchor text for retrieving domain-specific resources. While some link-contexts may misguide focused crawling and obtain the inappropriate Web pages, because several relevant anchor texts become irrelevant or several irrelevant anchor texts become relevant after calculating the relevance (e.g., our previous work on selectively using link context presented in IAJIT). For irrelevant content blocks, we adopt a heuristic-based approach to deal with anchor text and the link-context to make the topic of out-link page clearer and more accurate. Although there are many advantages using link-context to crawl the Web pages, it is not known whether every anchor text should combine with several bytes link-contexts in a text window. Therefore, in the light of this, this paper presents a novel

* Corresponding author at: College of Computer Science and Technology, Jilin University, Changchun 130012, China.
E-mail addresses: tpeng@jlu.edu.cn, taopeng@illinois.edu (T. Peng), liulu0804@gmail.com (L. Liu).

0950-7051/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.knosys.2013.06.008

method combining Content Block Partition and Selective Link Context (CBP–SLC) to enhance focused crawling guided by text classifier.

The strategy of text classification is also another direct impact on the performance of focused crawling. In order to select links that lead to documents of interest, and avoid links that lead to off-topic regions, focused crawlers use the page classifiers to guide the search. For more domain-specific information, focused crawler gives priority to links that belong to pages classified as relevant. However, it is generally known that the primary challenge of text classification problem is that no labeled reliable negative documents are available in the training example set, and it is quite costly, exhausted and difficult to obtain training data. So, PU (learning from Positive and Unlabeled examples) text classifying is a very effective strategy. Text classification is the task of automatically applying labels to new documents, which is used widely [18,25,36,41], based on the classifier learnt from training examples. This paper also presents a TFIDF-improved approach, TFIPNDF (Term Frequency Inverse Positive–Negative Document Frequency), for weighting the terms in the positive and negative training example set, respectively. According to TFIPNDF, during the process of training classifier, a term plays different roles in training set. That is, the more documents a positive feature appears in, the more significant it is in positive training set. Also, our method has higher requirements on the quality of the training set, especially the negative data set. Therefore, how to identify more reliable negative documents is also studied in this paper.

The rest of the paper is organized as follows. We review the related work in Section 2. Section 3 illustrates how to represent and use CBP–SLC to enhance the focused crawling. Section 4 describes how to identify more reliable negative documents from the unlabeled example set using 1-DNFC and presents the process of our text classifying. The whole crawling procedure is proposed in Section 5. Several comprehensive experiments are performed to evaluate the effectiveness of our method in Section 6 in which experimental settings, performance metrics and results are provided. Section 7 draws the conclusions.

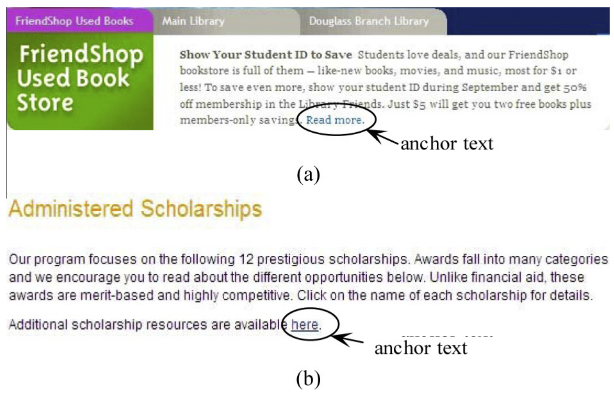
2. Related work

Since the early days of the WWW, researchers have explored different methods of Web information organization and retrieval. In what follows, we briefly review some work on Web crawling, text classification and TFIDF.

Compared to the standard Web crawlers, focused crawlers yield good recall as well as good precision by restricting themselves to a limited domain. They allow a higher-level application to gather domain-specific information from the web, a focused collection rich in information about a topic or theme. Many techniques have been proposed to enhance focused Web crawling [2,4,9,39,44]. Early, there are Breadth-First crawler [32], Depth-First crawlers such as Fish Search [7] and other focused crawlers include [14,24,29]. Due to the topic information and abstract provided by anchor text, the hyperlink structures in Web pages are utilized by many researchers to search the Web [11]. Ozel [26] put forward a genetic algorithm based automatic Web page classification system which used both HTML tags and terms belonging to each tag as classification features and learned optimal classifier from the positive and negative Web pages in the training dataset. Iwazume et al. [15] combined anchor text and ontology theory in order to guide Web crawler. Brin and Page [3], the founders of Google search engine, also used anchor text to build index for URLs. Jung [17] proposed the context-based focused crawler architecture to discover local knowledge from interlinked systems and a knowledge alignment process to integrate the discovered local knowledge. Eiron and McCurley [10] presented a statistical study of the nature of anchor text and real user queries on a large corpus of corporate intranet documents. Tateishi et al. in [38] evaluated Web retrieval methods using anchor text. Li et al. [20] proposed a focused crawler guided by anchor texts using a decision tree. BioCrawler developed by Batzios et al. [1] employed the principles of BioTope's intelligent agents on the semantic web, learned which sites were rich in semantic content and which sites linked to them and adjusted its crawling habits accordingly. SharkSearch applied not only anchor text but also its texts that appear in adjacent area to evaluate the benefit of crawling along the anchor text [14]. Chakrabarti et al. [4] suggested the idea of using DOM offset, based on the distance of text tokens from an anchor on a tag tree, to score links for crawling. By using link-context, the crawler [30] performed well in forecasting and prioritizing unvisited URLs. The approach initially looks for some referring pages by traversing backward from seed URLs, and then builds initial term-based feature set by parsing the link-contexts extracted from those reference Web pages. Web page content block partition was also used for tunneling, which traversed irrelevant page and reached a relevant one [22,31]. And, tunneling improved the effectiveness of focused crawling by expanding its reach. But, the method omits the links in the irrelevant content blocks, in which there may be some anchors linking the relevant Web pages. Pant [28] introduced a framework to study link-context derivation methods. The framework included a number of metrics to understand the utility of a given link-context.

Some semi-supervised learning methods from positive and unlabeled examples were done. Letouzey et al. [19] performed experiments by using k-DNF and decision trees to learn from positive and unlabeled data. Liu et al. [21] also summarized the usual method for solving the PU-oriented text classification. Yu et al. [42] presented an algorithm called PEBL that achieved classification accuracy (with positive and unlabeled data) as high as that of traditional SVM (with positive and negative data). And, the PEBL algorithm uses the 1-DNF algorithm to identify a set of reliable negative documents. Denis et al. [8] presented a NB based method (called PNB) that tried 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 [23], which tries to learn the support of the positive distribution. We implement the one-class SVM in this paper, and the experimental results show that the performance is poorer than learning methods that take advantage of the unlabeled data.

TFIDF is the most common weighting method used to describe documents in the Vector Space Model in applications related to text mining, and is also commonly used in applications related to IR to compare a query vector with a document vector using a similarity or distance function such as the cosine similarity function. There are many variants of TFIDF. Soucy and Mineau [37] introduced a new weighting method based on statistical estimation of the importance of a word for a specific categorization problem, which had the benefit to make feature selection implicit. Xu et al. [40] proposed an I_TFIDF term weighting approach for imbalanced binary class text classification, which additionally considered two imbalanced factors (imbalanced occurring of term in two classes and imbalanced class distribution of training set). According to their experiment, the method holds for an automated classification system to identify MEDLINE abstracts referring to host-pathogen protein–protein interactions. Han [12] presented a weighting method, in which vector components were weighted using an iterative approach and the weights are slightly modified. The weight is not designated by a specific one, instead, an optimal set of weights are used. The method is generally much too slow to


FriendShop Used Books Main Library Douglass Branch Library

FriendShop Used Book Store

Show Your Student ID to Save Students love deals, and our FriendShop bookstore is full of them – like-new books, movies, and music, most for $1 or less! To save even more, show your student ID during September and get 50% off membership in the Library Friends. Just $5 will get you two free books plus members-only savings. Read more.

Administered Scholarships

Our program focuses on the following 12 prestigious scholarships. Awards fall into many categories and we encourage you to read about the different opportunities below. Unlike financial aid, these awards are merit-based and highly competitive. Click on the name of each scholarship for details.

Additional scholarship resources are available here.

Fig. 1. Instances of some anchor texts that fit for extracting link-context.

be used, particularly for a large feature space problem. Hao et al.[13] also proposed TFIDF + LSI algorithm to guide the crawling based on combining their advantages of LSI and TF-IDF, in which the crawler using TFIDF + LSI performed the same crawl task and demonstrated the combination advantage of TF-IDF and LSI. Then, the experiment suggests that the crawler's performance using TFIDF + LSI is greatly superior to that using either TF-IDF or LSI respectively.

3. CBP–SLC

This paper segments Web pages into some smaller content blocks based on DOM tree structure using Content Block Partition (CBP) algorithm. In our previous work [22,31], Web page content block partition is also used for tunneling, which utilizes content block partition to calculate the relevance of different regions, respectively. Therefore, a highly relevant region in a low overall relevance Web page will not be obscured. But, the method omits the links in the irrelevant content blocks, in which there may be some anchors linking the relevant Web pages. So, in this paper, if a content block is relevant, focused crawler will fetch the pages linked by the anchors in the block. Otherwise, focused crawler will selectively use link-context (SLC) to expand the links in the block according to the relevance.

3.1. Content block partition

In an irrelevant Web page, there may be some regions that are relevant. So, in the process of focused crawling, segmenting multiple-topic Web pages into some smaller single topic content units can significantly retrieve more relevant pages. Before segmenting Web pages, we correct the HTML markup of a Web page and create a DOM tree. We tidy the HTML page into a well-formed Web page using HTML TIDY tool[1] because many Web pages are badly written. In this paper, some important tags are treated as content block tags, such as ⟨H1–H6⟩, ⟨P⟩, ⟨ADDRESS⟩, ⟨CENTER⟩, ⟨UL⟩, ⟨DT⟩, ⟨TABLE⟩, ⟨TH⟩, ⟨TR⟩, and ⟨TD⟩. These HTML tags are commonly used as the nodes of the sub-trees in Web pages, and the content blocks are just sub-trees in DOM tree. Algorithm 1 describes the process of CBP.

Algorithm 1. Content Block Partition algorithm. blockHeight (v, 0) refers to the height of sub-tree whose root is v in *Tp*. $s = \alpha \times$ blockHeight (*root*, 0), $0 < \alpha \leqslant 1$, α is a weight for content unit partition granularity adjustment. $k = 1$.

```
Algorithm of Content Block Partition (CBP)
Input: p: HTML parse tree, k: a integer parameter.

1. procedure CBP
2. Initialize
3.   Tp ← p
4.   q ← root of Tp
5.   repeat
6.     v ← top element in q
7.     if v has a child with at least k links and blockHeight
         (v, 0) >= s then
8.       push all the children of v to q
9.     else
10.      v is a content block
11.    end if
12.  until q is empty
13. end procedure
14. procedure blockHeight (Node iNode, int height)
15.   count ← height
16.   if iNode has child nodes then
17.     children ← the child nodes of iNode
18.     hTemp ← 0
19.     fTemp ← count
20.     for each child_node in children
21.       if the child_node is a content block tag then
22.         hTemp ← blockHeight (child_node, count + 1)
23.       else
24.         hTemp ← blockHeight (child_node, count)
25.       end if
26.       if hTemp > fTemp then
27.         fTemp ← hTemp
28.       end if
29.     end for
30.     if fTemp > count then
31.       count ← fTemp
32.     end if
33.   end if
34.   return count
35. end procedure
```

3.2. Selectively using link-context

Sometimes, some Web page designers do not summarize the destination Web pages in the anchor text. Instead, they use words such as "Click here", "here", "Read more", "more", "next" and so on to describe the texts around them in anchor text (shown in Fig. 1(a) and (b)). However, if we calculate every similarity or relevance between link-context and feature term about a specific topic, we may also omit some Web pages that are relevant indeed, or extract some Web pages that are not relevant. As Fig. 2(a) shown that, if we want to search "job" area and anchor text "Find Jobs with I-link" meets the requirement of relevance, but it becomes irrelevant after extracting link-contexts because the anchor text on the right of Fig. 2(a) does not belong to the "job" area. Then we will miss the anchor text. Another kind of circumstance is shown in Fig. 2(b). There are four anchor texts in Fig. 2(b). Suppose we want to search "business" area, when we search the first anchor text, and we find out it does not belong to the "business" area. However, after combining its link-context, it belongs to "business" area, which leads to extract wrong Web pages. Motivated by these circumstances, selectively using link-context to enhance focused crawling for domain-specific resources is utilized to improve the efficiency and accuracy of focused crawlers.

[1] http://www.w3.org/People/Raggett/tidy/.

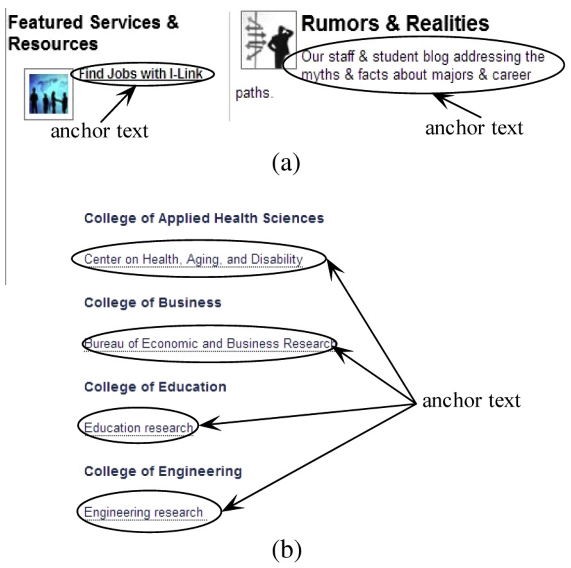

Featured Services & Resources
Find Jobs with I-Link
anchor text
Rumors & Realities
Our staff & student blog addressing the myths & facts about majors & career paths.
anchor text
(a)
College of Applied Health Sciences
Center on Health, Aging, and Disability
College of Business
Bureau of Economic and Business Research
College of Education
Education research
College of Engineering
Engineering research
anchor text
(b)

Fig. 2. Instances that do not fit for extracting link-context.

Text around links, which help people surf the Internet, are key components of the Web pages. Fig. 3 shows a framework of a simple Web consisting three Web pages from the Web crawler's perspective.

The anchor text is the visible, clickable text in a hyperlink [10]. Information provided by the same content block where the link appears is an effective way to enrich the context of the anchor text. Pant [28] presented two link-context derivation techniques. We use the method of deriving link-context from Aggregation Nodes, with some modifications. After segmenting the HTML page, we insert missing tags and reorder tags in the "dirty" pages 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 appear at leave nodes on the tag tree. This preprocessing simplifies the analysis.

If we want to extract the link-context of an anchor text, then first, we locate the anchor. Next, we treat each node, which is on the path from the root of the tree to the anchor, as a potential aggregation node. From these candidate nodes, we choose the parent node of anchor, which is the grandparent node of the anchor text as the aggregation node. Then, all of the texts in the sub-tree rooted at the node are retrieved as the contexts of the anchor. If one anchor appears in many different blocks, combine the link-context in every block as its full link-context.

Compared with [28], we fixed the aggregation node. In fact, for each block, we have an optimal aggregation node that provides the highest context similarity for the corresponding anchor with different aggregation node. It is very labor-some 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. We set a tradeoff on quantity and quality.

4. Building text classifier using SVM

Constructing our text classifier adopts two steps: first, identify a set of reliable negative documents from the unlabeled set by using our improved constrained 1-DNF algorithm (1-DNFC). Then, build a set of sub-classifiers by iteratively applying the SVM, the most common algorithm used to text classification [27,34,43], and construct the final text classifier by using the weighted voting method.

4.1. Identifying reliable negative documents using improved 1-DNF

For identifying the reliable negative documents from the unlabeled examples set, we must select the features of the negative documents. For example, if the frequency of a term which occurs in the positive examples set exceeds 80%, whereas less than 5% in the unlabeled examples set, then this term is considered to be positive. Motivated by this idea, we can obtain a positive feature set *PF*. A document, which does not contain any positive features, in the unlabeled examples set can be regarded as a reliable negative example. For describing expediently, we define the following notation: *P*, *U*, *PF*, *RN*, *PON*, and NEG_0 represent the positive example set, unlabeled example set, positive feature set, reliable negative documents, training example set, and the reliable negative document set produced by our Constrained 1-DNF algorithm (1-DNFC), respectively. $NEG_i\ (i \geqslant 1)$ represents the negative document set produced by the *i*th iteration of the SVM algorithm. Compared with 1-DNF algorithm [42], the constrained 1-DNF considers not only the diversity of the feature frequency in *P* and *U*, but also a constrained frequency (λ) of the feature in *P*. Therefore, constrained 1-DNF algorithm (1-DNFC) is described as follows.

Algorithm 2. Algorithm of Constrained 1-DNF. $|P|$ is the number of the documents in *P*, $|U|$ is the number of the documents in *U*, *freq* (t_i, P) is the number of documents where feature t_i occurs in *P*, and *freq* (t_i, U) is the number of documents where feature t_i occurs in *U*.

```
Algorithm of Constrained 1-DNF
Input: term set $t = \{t_1, t_2, \ldots, t_m\}$, $t_i \in U \cup P$. unlabeled example
set $U = \{d_1, d_2, \ldots, d_n\}$

1. procedure 1-DNFC
2.   Initialize
3.   $PF \leftarrow$ null
4.   $RN \leftarrow U$
5.   for each term $t_i$ in $t$
6.     if $freq(t_i, P)/|P| > freq(t_i, U)/|U|$ and $freq(t_i, P)/|P| > \lambda$ then
7.       $PF \leftarrow PF \cup \{t_i\}$
8.     end if
9.   end for
10.  for each document $d_i$ in $U$
11.    for each $t_j$ in $PF$
12.      if $freq(t_j, d_i) > 0$ then
13.        $RN \leftarrow RN - \{d_i\}$
14.      end if
15.    end for
16.  end for
17.  return $RN$
18. end procedure
```

4.2. Building classifier by iteratively applying SVM

Text classifier embeds the documents into some feature space, which may be extremely large, especially for very large vocabularies. And, the size of feature space affects the efficiency and effectiveness of text classifiers. Therefore, pruning the feature space is necessary and significant. In this paper, we prune the feature space according to the term frequency, which is based on the hypothesis that those very rare words are unimportant for the classifying, and they will not affect the performance of the overall situation.

Thus, when representing the weight of term t_j in document d_i, x_{ij} is equal to 0 if term t_j does not occur in document d_i. So, the feature vector for document d_i is written as $d_i = (t_1{:}\ x_{i1}, \ldots, t_j{:}\ x_{ij}, \ldots, t_{im}{:}x_{im})$. And an example is also illustrated in Fig. 4 after feature weighting.

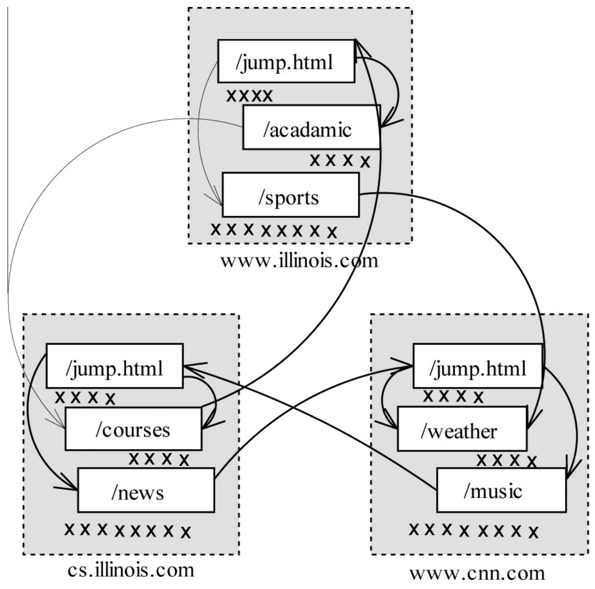
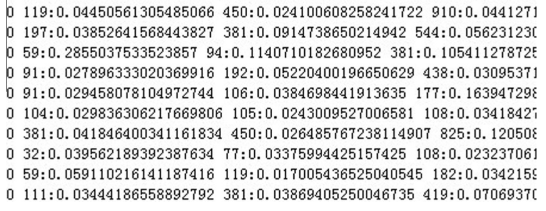


Fig. 3. A simple Web consisting three Web pages. The arrows denote links between the pages.

In this paper, we adopt an improved TFIDF term weighting method, TFIPNDF (Term Frequency Inverse Positive–Negative Document Frequency), which is based on statistical term frequency and inverse document frequency components from positive and negative training examples, respectively. Compared with TFIDF, term frequency component of TFIPNDF is the same with TFIDF. However, when weighting inverse document frequency component, TFIPNDF method calculates the IPDF (Inverse Positive Document Frequency) and INDF (Inverse Negative Document Frequency) weight values in the positive and negative training examples, according to the distribution of the terms, respectively. In other words, IPNDF (or, IPDF and INDF) reflects the importance of the term in the positive and negative training examples, respectively. Therefore, TFIPNDF is composed of two parts:

$$TFIPNDF = \begin{cases} f_{ki} \times \frac{P_i}{S_P} \times \log\left(\frac{N}{n_i}\right), & (document_k \in P) \\ f_{ki} \times \frac{N_i}{S_N} \times \log\left(\frac{N}{n_i}\right), & (document_k \in N) \end{cases} \tag{1}$$

where f_{ki} is the frequency of word i in document k, N is the number of documents in the collection, n_i is the number of documents where word i occurs in the collection, P_i is the number of positive documents where word i occurs, N_i is the number of negative documents where word i occurs, S_P and S_N are the numbers of positive and negative documents in the collection, respectively.



```
0 50:0.17030419900915347 197:0.04711820778838992 320:0.21643716
0 119:0.044505613054850066 450:0.02410060825824172 910:0.0441271
0 197:0.03852641568443827 381:0.0914738650214942 544:0.0562312
0 59:0.2855037533523857 94:0.1140710182680952 381:0.1054112787
0 91:0.027896333020369916 192:0.0522040019665062 9 438:0.0309537
0 91:0.029458078104972744 106:0.0384698441913635 177:0.1639472
0 104:0.029836306217669806 105:0.0243009527006581 108:0.0341842
0 381:0.041846400341161834 450:0.026485767238114907 825:0.12050
0 32:0.039562189392387634 77:0.03375994425157425 108:0.0232370
0 59:0.059110216141187416 119:0.017005436525040545 182:0.034215
0 111:0.03444186558892792 381:0.038694052500467 35 419:0.0706937
```

Fig. 4. Illustration of a snippet of data matrix that stores the feature vectors (each row represents a feature vector).

A document collection may contain documents of many different lengths. It is useful to use normalized weight assignments. A vector length normalization of TFIPNDF is defined as:

$$x_{ki} = \begin{cases} \dfrac{f_{ki} \times \frac{P_i}{S_P} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{kr} \times \frac{P_r}{S_P} \times \log\left(\frac{N}{n_r}\right)\right]^2}}, & (document_k \in P) \\ \dfrac{f_{ki} \times \frac{N_i}{S_N} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{kr} \times \frac{N_r}{S_N} \times \log\left(\frac{N}{n_r}\right)\right]^2}}, & (document_k \in N) \end{cases} \tag{2}$$

where M is the number of all the features.

After document pre-processing, feature extracting and term weighting, we build a set of sub-classifiers by iteratively applying the SVM algorithm and construct the final text classifier. We do not designate one specific classifier in the classifier set as the final one, instead, all of them are used to construct the final classifier based on voting method. Algorithm 3 is the implementation of constructing the text classifier (WVC: Weighted Voting Classifier) by using weighted voting method.

Algorithm 3. Constructing the final classifier by using weighted voting method.

```
Algorithm of Constructing the Weighted Voting Classifier
  (WVC)
Input: positive example set $P$, unlabeled example set $U$,
  reliable negative example set $NEG_0$.
1. procedure Classifying
2.   Initialize
3.   $PON \leftarrow P \cup NEG_0$, $U \leftarrow U - NEG_0$
4.   $i \leftarrow 0$, $allP \leftarrow 0$
5.   repeat
6.     $SVM_i \leftarrow SVM\_training\ (PON)$
7.     $p_i \leftarrow$ precision of $SVM_i\_predict\ (P)$
8.     $allP \leftarrow allP + p_i$
9.     $NEG_{i+1} \leftarrow$ the negative documents classified by
       $SVM_i\_predict\ (U)$
10.      $PON \leftarrow PON \cup NEG_{i+1}$
11.      $U \leftarrow U - NEG_{i+1}$
12.      $i \leftarrow i + 1$
13.   until $NEG_{i+1}$ is empty
14.   $FinalClassifier = \sum_{k=0}^{i} \frac{p_k}{allP} SVM_k$
15. end procedure
```

5. Focused crawling architecture

The Web crawler has two jobs: downloading pages and finding URLs. The crawler begins with a group of seed URLs, which are provided to the crawler as starting parameters. During the process of crawling, after being downloaded and preprocessed (eliminating *stopwords* and stemming), a Web page is first classified by a text classifier with a high dimensional dictionary. The relevant pages are added into the relevant page set. At the same time, the Web page is parsed into the page's DOM tree and partitioned into many content blocks according to HTML content block tags based on content block partition strategy. And then all anchors in the content blocks will be picked. If the page linked by the anchor has been crawled or the anchor is in the queue of crawling (*frontier*), the anchor is omitted. We compute the relevance of content blocks with the term-based features using text classifier. If a content block is relevant, all unvisited URLs are extracted and added into *frontier*,

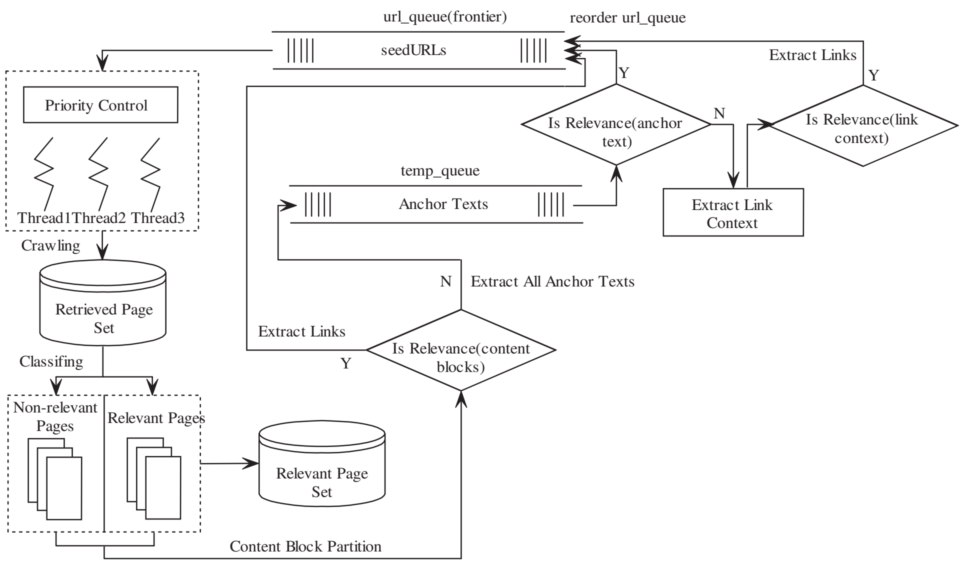


Fig. 5. The architecture of topical crawler enhanced by selectively using link-context.

and the content block relevance is treated as priority. If the content block is not relevant, in which, the anchors are computed to get the relevance. If an anchor text is relevant, it is added into the *url_queue* or *frontier*. Otherwise, its link-context needs to be extracted and the relevance is calculated again. And if the link-context is relevant, the anchor is also added into *frontier*. The unvisited URL that has highest priority will be first selected for crawling. Whenever a new anchor is inserted into the waiting queue, the queue will be readjusted to create its new *frontier*. Fig. 5 illustrates the architecture of our focused crawler.

Although we consider a tradeoff on quantity and quality, we might make many mistakes when extracting every link-context of its anchor. One kind of mistakes is that we may extract a host of Web pages that are irrelevant to the specific domain. Because the anchor text itself is irrelevant, however, it becomes relevant after combining with its link-context by mistake. Another kind of mistakes is that a host of relevant Web pages are not extracted still caused by extracting link-context inappropriately. Our approach, which selectively makes use of link-context, can solve this problem quite well. Before extracting every anchor's link-context from irrelevant content block, we compute the relevance of anchor text first. If it is relevant, then the corresponding URL is added into the *frontier*. There is no need to calculate the relevance of its link-context. As for the irrelevant anchor texts, we just extract their corresponding link-contexts to judge whether or not they indeed link to relevant pages. To some extent, it not only improves the efficiency, but also reduces the risk of making mistakes. Algorithm 4 is the implementation of the focused crawling procedure enhanced by CBP–SLC.

The relevance of anchor text, link-text, content blocks and Web pages is computed using our WVC classifier. In our focused crawler, we compute the weight of each term based on *tfc* weighting scheme (Eq. (3)) [35] after preprocessing which includes removing *stopwords* and stemming.

$$x_{ki} = \frac{f_{ki} \times \log\left(\frac{N}{n_i}\right)}{\sqrt{\sum_{r=1}^{M}\left[f_{kr} \times \log\left(\frac{N}{n_r}\right)\right]^2}} \quad (3)$$

where f_{ki} is the frequency of word i in text unit k (anchor text, link-context, content block or Web page), N is the number of text units in the collection, M is the number of all the features. n_i is the number of text units where word i occurs.

Algorithm 4. Focused Crawling Enhanced by CBP–SLC.

```
Algorithm of Focused Crawling Enhanced by CBP–SLC
Input: seedURLs: a queue for seed URLs
1. procedure CBP–SLC
2.   enqueue seedURLs into url_queue //append element at the end of queue
3.   repeat
4.     url ← dequeue(url_queue) //remove the element at the beginning of //queue and return it
5.     page ← crawl url and get the corresponding Web page
6.     if page is relevant then
7.       put page into relevance_pageDB
8.     end if
9.     block_list ← CBP (page)
10.    for each block in block_list
11.      WVC (block)
12.      if block is relevant then
13.        enqueue its unvisited urls into url_queue
14.      else
15.        temp_queue ← extract all anchor texts
16.        for each anchor text in temp_queue
17.          WVC (anchor text)
18.          if anchor text is relevant then
19.            enqueue its unvisited url into url_queue
20.            dequeue url in temp_queue
21.          else
22.            extract link-context of the corresponding anchor text
23.            WVC (link-context)
24.            if link-context is relevant then
25.            enqueue its unvisited url into url_queue
26.            end if
27.            dequeue url in temp_queue
28.          end if
29.        end for
30.      end if
31.    end for
32.  until url_queue is empty
33. end procedure
```

6. Experiments and results

In this section, several tests have been used to verify whether the methods presented in this paper hold for domain-specific information retrieval. The experiments comprise two parts: The comparison of text classifying performance and domain-specific information retrieval.

6.1. The comparison of text classifying

In the experiment, we used the Reuters-21,578 dataset, Reuters Corpus Volume 1 (RCV1),[2] 20 Newsgroups,[3] and Open Directory Project,[4] as our training and test data set. Of the 135 topics in Reuters-21,578, 9980 documents from the most populous 10 topics are used in this paper. RCV1, which has about 810,000 Reuters, English language news stories collected from the Reuters newswire. We use "topic codes" set in category codes (topic, industry, and region). In the "topic" hierarchy, four hierarchical groups are included: CCAT (Corporate/Industrial), ECAT (Economics), GCAT (Government/Social), and MCAT (Markets). Among 789,670 documents, 10,000 documents are used in this paper. The 20 Newsgroups dataset, collected by Ken Lang, consists about 20,000 newsgroup documents, which partition evenly across 20 different newsgroups. In this paper, 10 classes are applied to be our dataset, 9840 documents in total. ODP is the largest, most comprehensive human-edit directory of the Web and is available in RDF. The data structure of ODP is organized as a tree, where nodes contain URLs that link to the specific topical Web pages, thus we use the first three layers and consider both hyperlink text and the corresponding description, and ignore all non-English URLs. We choose ten topics as samples to test the performance of our method instead of crawling all the URLs from the first layer of *Dmoz*. Besides, 1000 samples are chosen from each class.

6.1.1. Performance metrics

As a supervised or semi-supervised learning algorithm, the performance evaluation of text classifying can reflect the availability of the system directly. Most classification tasks are evaluated using *Precision*, *Recall* and *F-Measure*. *Precision* for text classifying is the fraction of documents assigned that are relevant to the class, which measures how well it is doing at rejecting irrelevant documents. *Recall* is the proportion of relevant documents assigned by classifier, which measures how well it is doing at finding all the relevant documents. We assume that *T* is the set of relevant documents in test data set, *C* is the set of relevant documents assigned by classifier. Therefore, we define *Precision* and *Recall* as follows:

$$Precision = \frac{|C \cap T|}{|C|} \times 100\% \qquad (4)$$

$$Recall = \frac{|C \cap T|}{|T|} \times 100\% \qquad (5)$$

F-Measure, which is defined as the harmonic mean of *Precision* and *Recall* value, is also used to measure the performance of our method. For different specific request, according to the importance of the *Precision* and *Recall*, we define *F-Measure* as follows:

$$F\text{-}Measure = \frac{(\beta^2 + 1)Precision \times Recall}{\beta^2 \times Precision + Recall} \qquad (6)$$

where β is a weight for reflecting the relative importance of *Precision* and *Recall* value. Obviously, if $\beta > 1$, then the*Recall* value is more important then *Precision* value. In this paper, β is assigned a constant 1.

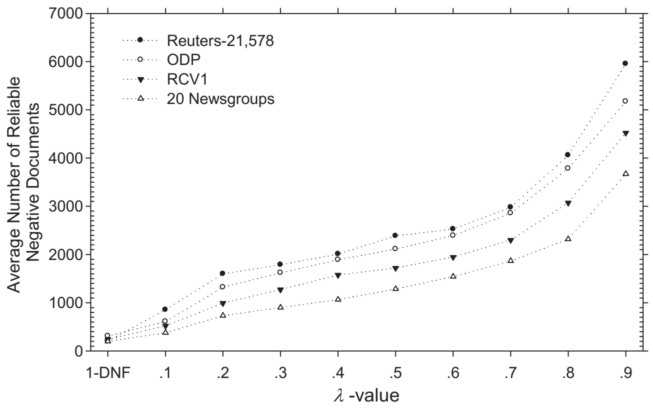


Fig. 6. The average number of reliable negative documents obtained by 1-DNF and 1-DNFC for each λ setting on four datasets.

We also define *ERR* to compare the improved 1-DNF algorithm with the 1-DNF algorithm in the number of identifying reliable negative data and the error rate. Let *RN* (P_t) be the set of the positive documents in the reliable negative data, P_t be the set of positive documents in the unlabeled example set. The *ERR*(%) is calculated as follows:

$$ERR(\%) = \frac{|RN(P_t)|}{|P_t|} \times 100\% \qquad (7)$$

6.1.2. Evaluation of text classifying

For evaluating the performance of text classifying, we implemented PEBL algorithm, one-class SVM algorithm and our classifier in java. In the experiment, we used LIBSVM[5] (version 2.71). We used the standard parameters of the SVM algorithm in one-class SVM, PEBL and WVC. The low frequency features (assigned a constant 5 in this paper) and stopwords in the dictionary are filtered out. The features in all documents are reordered and mapped using "ID" in accordance with the order of the features in dictionary.

Fig. 6 compares the average number of reliable negative documents for each topic on the four datasets between 1-DNF and 1-DNFC at $\lambda = 0.1–0.9$, respectively. Results show that the number of the reliable negative documents identified by the 1-DNFC algorithm is significantly more than that identified by the original 1-DNF. And, as shown in Fig. 7, the average error rates of identifying the positive data as negative are lower on $\lambda = 0.1–0.4$ settings. So the comparisons indicate that the improved 1-DNF algorithm can identify more reliable negative documents with very low error rates, especially on $\lambda = 0.1$ and 0.2 settings. Therefore, in the light of the plots as shown in Figs. 6 and 7, λ is assigned a constant 0.2 in the rest of the experiments.

In order to verify the effectiveness of TFIPNDF, we run the classifying system using different term weighting methods. Fig. 8 compares the performance of *F-Measure* achieved by our classifying method using TFIPNDF and TFIDF weightings for each topic on the four datasets. We observe that the performance of classifying using TFIPNDF weighting outperforms TFIDF on each dataset. For comparing the performance of the text classifying based on different techniques clearly, Fig. 9 plots *F-Measure* performance of OCS, PEBL and our weighted voting classifying method using TFIPNDF weighting for each dataset. The result indicates that the *F-Measures* of WVC are higher than other two methods. For comparing the effi-

[2] http://trec.nist.gov/data/reuters/reuters.html.
[3] http://qwone.com/~jason/20Newsgroups/.
[4] ODP, http://www.droz.org.
[5] http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

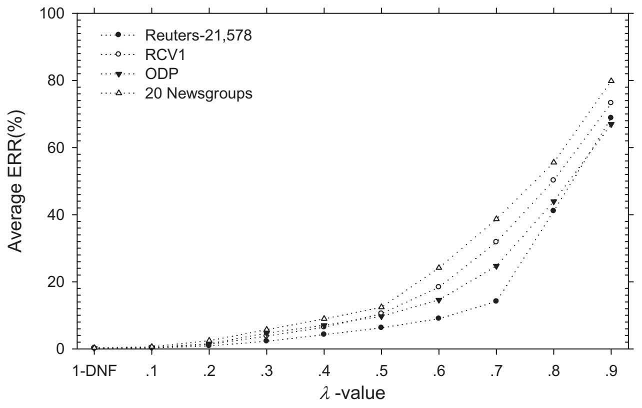


Fig. 7. The average error rates of identifying the positive data as negative using 1-DNF and 1-DNFC for each λ setting on four datasets.

ciency of classifiers, Table 1 also shows the average time consuming of building classifiers on four datasets. All of the values of WVC are less than other two classifiers on the four datasets.

6.2. Domain-specific information retrieval

In the experiment, we built crawlers that use different techniques for crawling the Web, and tested our method using multiple crawlers over 10 topics covering hundreds of thousands of pages. Under the guide of each method, crawler downloaded the page according to the strategies of Web page processing. Our crawler is multi-threaded and implemented in java, which provides for reasonable speed-up. We use 10 threads of execution starting from 100 relevant URLs (Seed URLs) on each topic picked from Open Directory Project while running the crawler. We first download 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 select 10 categories (same as Section 6.1) and the associated ODP relevant sets. These selected categories serve as topics for our crawling experiments. We select some ODP relevant sets as the seeds (contain 100 URLs), which are used to initialize the crawl, for each topic.

6.2.1. Performance metrics

The performance evaluation of focused Web crawling can reflect the availability of the system directly. Perhaps, the most crucial evaluation of focused crawling is to measure the rate at which relevant pages are acquired, and how effectively irrelevant pages are filtered out from the crawler. With this knowledge, *Harvest rate* and *Target recall* are used to evaluate the results after crawling *n* pages, which are running average, over different time slices of the crawl, of page relevance assessed using classifiers. For testing the performance of conquering and traversing irrelevant pages to

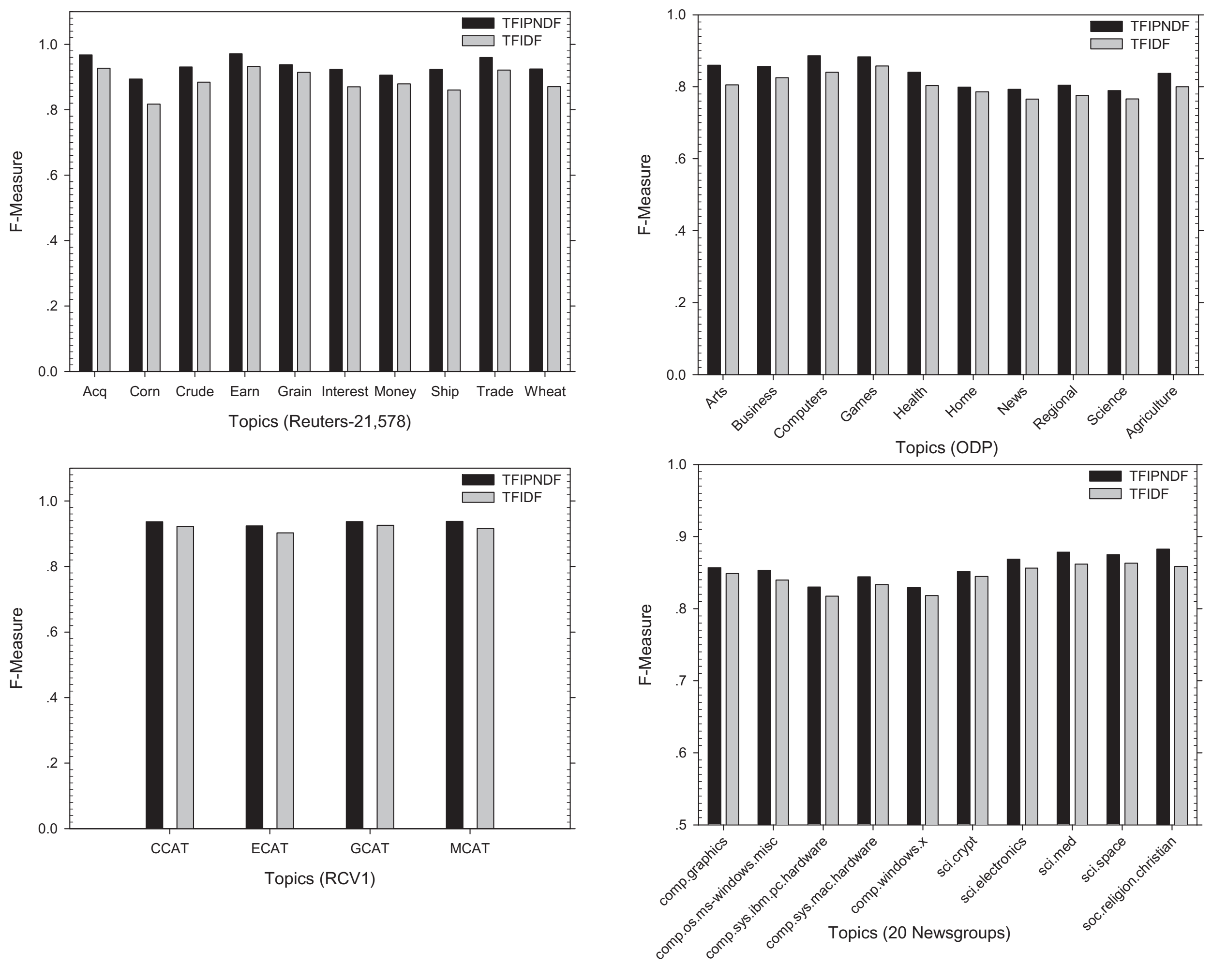


Fig. 8. The comparison of *F-Measures* achieved by our weighted voting classifying method using TFIPNDF and TFIDF weightings for each dataset.

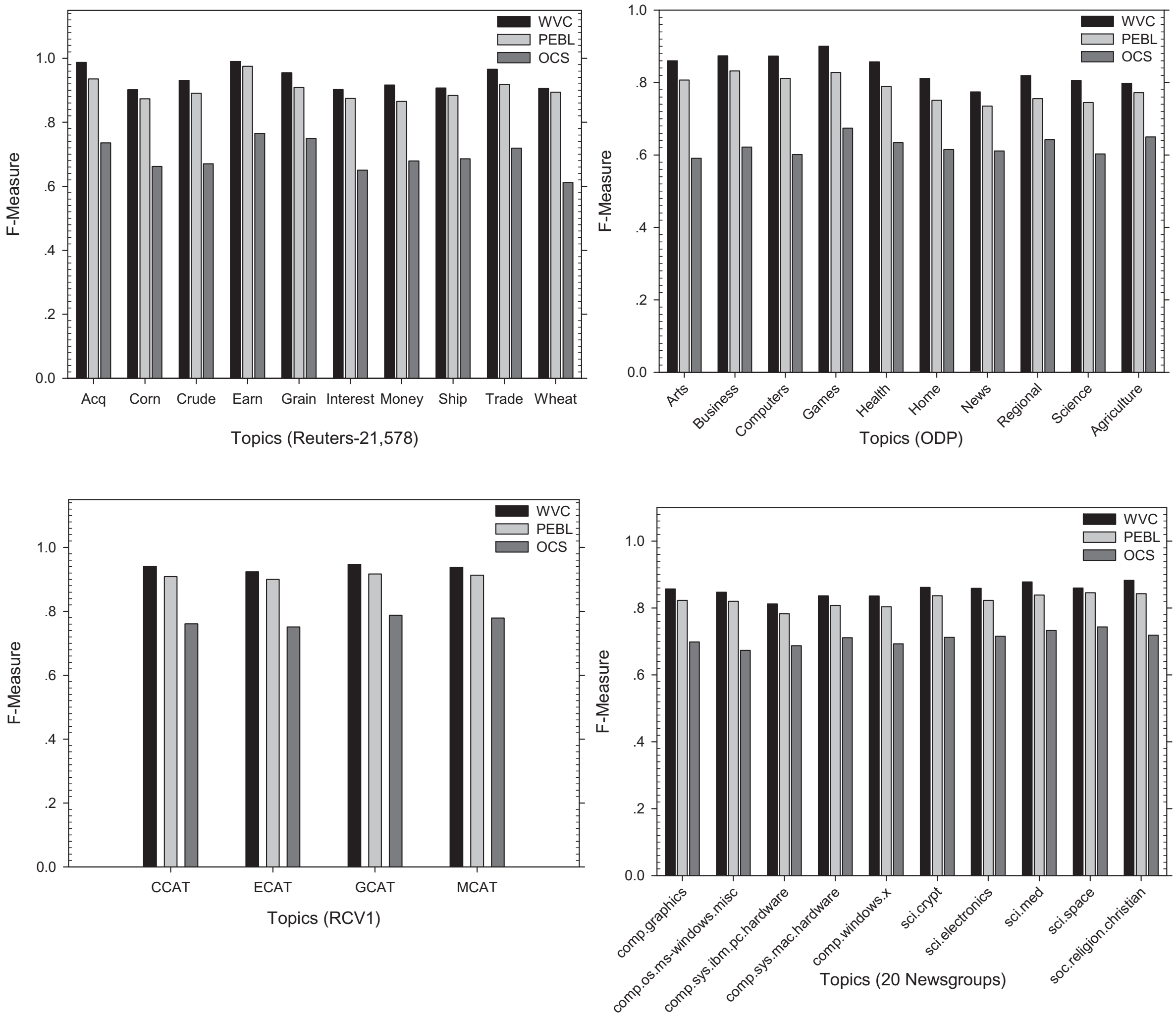


Fig. 9. The performance of three text classifying methods using TFIPNDF weighting for each dataset.

Table 1
Average time consuming of building classifiers iteratively across topics on four datasets (seconds).

	OCS	PEBL	WVC
Reuters-21,578	11.261	9.237	8.015
RCV1	15.352	11.248	9.892
ODP	13.460	10.788	9.027
20 Newsgroups	11.439	9.593	8.305

reach relevant ones, we also adopt *Target length* metrics in this paper.

The *Harvest rate* is the fraction of pages crawled that are relevant to the topic, which measures how well it is doing at rejecting irrelevant Web pages. We make this decision by using our classifier instead of manual relevance judgment, which is costly. *Target recall* is the fraction of relevant pages crawled, which measures how well it is doing at finding all the relevant Web pages. However, the relevant set for any given topic is unknown in the Web, so the true *Target recall* is hardly to measure. In view of this situation, we delineate a specific network, which is regard as a virtual WWW in the experiment. Given a set of seed URLs and a certain depth, the range reached by a crawler using Breadth-First crawling strategy is the virtual Web. We assume that the target set T is the relevant set in the virtual Web, $C(t)$ is the set of first t pages crawled. Therefore, we define *Harvest rate* and *Target recall* as follows:

$$Harvest\ rate = \frac{|C(t) \cap T|}{|C(t)|} \times 100\% \quad (8)$$

$$Target\ recall = \frac{|C(t) \cap T|}{|T|} \times 100\% \quad (9)$$

Target length [31] is the distance from seed URL to target relevant Web page, which measures how well it is doing at conquering and traversing irrelevant pages to reach relevant ones (also called *Tunneling* in our previous work [31]). Fig. 10 illustrates how to compute *Target length L*. The shaded rectangle represents a relevant page, and the white one represents an irrelevant page, in which, some content blocks may be relevant. The solid lines indicate the path of crawling. According to Fig. 10, after crawling a specified number of relevant pages, the lesser the sum of the relevant page's target length L, the more irrelevant pages are conquered and traversed. So, we define *Target length* as follows:

$$Target\ length = \sum_{i=0}^{R} L_i \quad (10)$$

where R is the number of the relevant pages.

6.2.2. Evaluation of focused Web crawling

For comparing CBP–SLC with other focused crawling strategies, in this section, we built crawlers that used different techniques (*Breadth-First*, *Best-First*, *Anchor text only*, *Link-context only*, *Content*

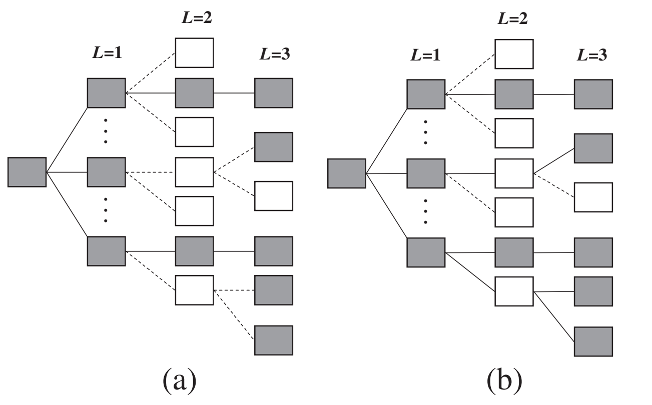


Fig. 10. The comparison of traversing irrelevant pages, in which some content blocks may be relevant. The more irrelevant pages are conquered and traversed in (b) than (a).

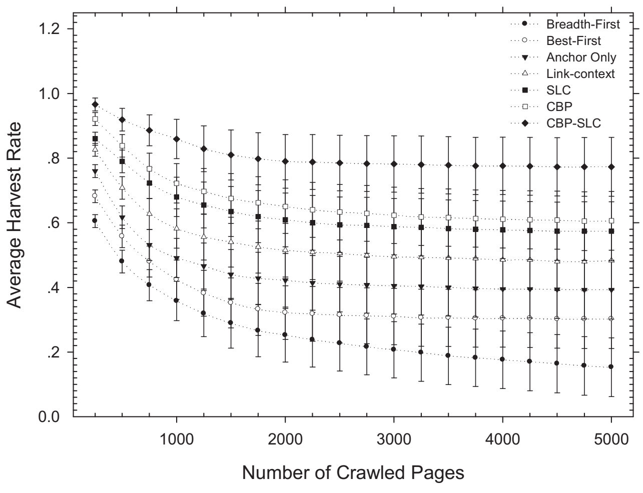


Fig. 11. Dynamic plot of *Harvest rate* versus the number of crawled pages. Performance is average across topics and standard error bars are also shown. The error bars correspond to ± standard error.

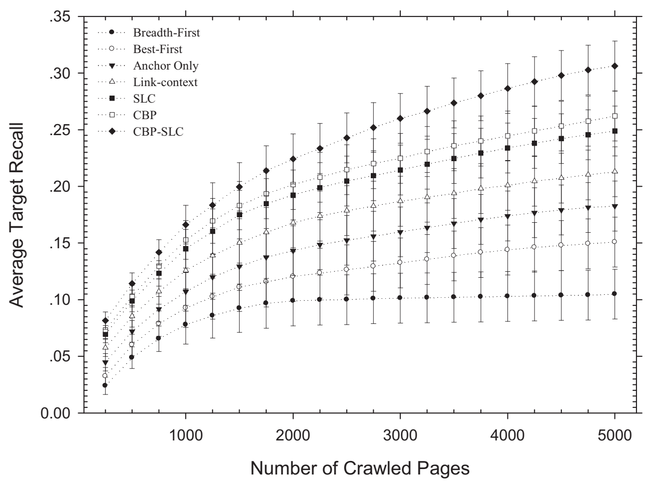


Fig. 12. Dynamic plot of *Target recall* versus the number of crawled pages. Performance is average across topics and standard error bars are also shown. The error bars correspond to ± standard error.

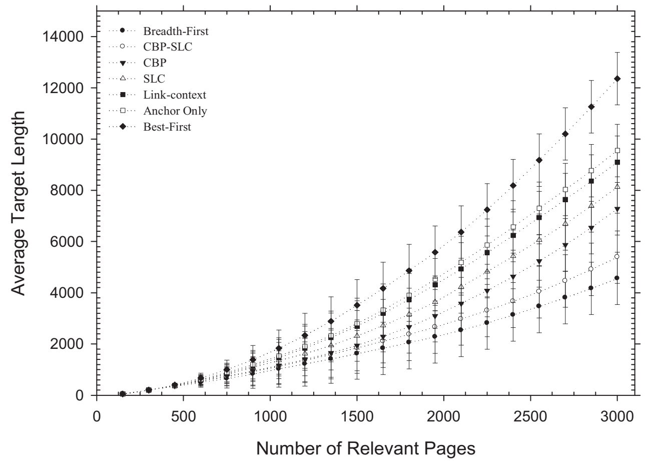


Fig. 13. Dynamic plot of *Target length* versus the number of relevant pages. Performance is average across topics and standard error bars are also shown. The error bars correspond to ± standard error.

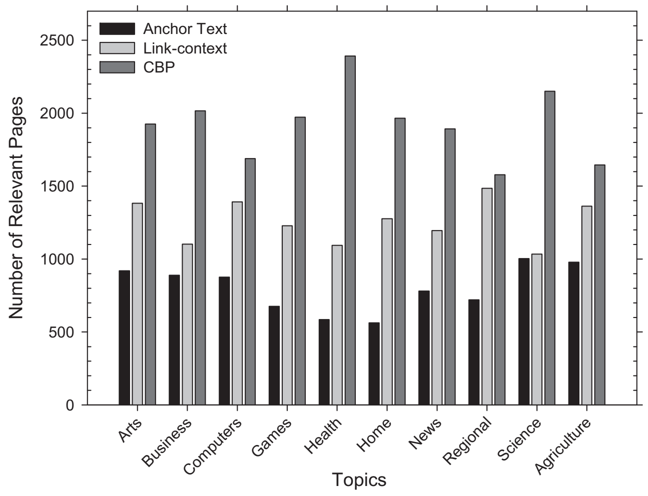


Fig. 14. The comparison of using anchor text, link-context and CBP in fetching relevant pages for each topic.

Block Partition (*CBP*) and *Selective Link-context* (*SLC*)), which are described in the following, to crawl the Web. Different Web page content block partition methods have different impacts on focused Web crawling performance. According to the experimental result in [31], alpha in Algorithm 1 is assigned a constant 0.5 in this paper.

In order to reflect the comprehensive of our method, Figs. 11 and 12 show the average *Harvest rate* and average *Target recall* on ten topics for each crawling strategy, respectively. Fig. 13 plots the average *Target length* on ten topics for each strategy. In the light of the results, the *Target length* of Breadth-First is the least. That is, it has a good performance in traversing irrelevant pages. Also, CBP–SLC has the second best performance. But, *Breadth-First* without judging the unvisited URLs has the worst performance in *Harvest rate* and *Target recall*. Therefore, *Breadth-First* fetched large numbers of irrelevant pages. And, its performance mainly depends on the localization of the relevant pages. Based on all results, CBP–SLC has the best performance among other methods. Also, CBP and SLC have nice results in *Harvest rate*, *Target recall* and*Target length*. *Best-First* predicts the relevance and groups the unvisited URLs based on the whole page picked up from. Therefore, it has a bad performance when meeting multiple topic Web pages, in which there may be a lot of noise. *Best-First* will also miss a lot of relevant pages. And, the *Target length* is the longest among all methods. Only analyzing the anchor text may omit much useful textual information because it is short and not informative enough. However, extracting its link-contexts every time can not only reduce the efficiency but also cause multiple mistakes. Their weakness can be just overcome by our method. In summary, the focused crawler enhanced by CBP–SLC shows significant performance improvement over the crawlers mentioned above.

In the experiment, we also set three counters to calculate the times of using anchor text only, combining link-contexts, and using content block partition, respectively. The comparison results (as shown in Fig. 14) show that there are indeed some relevant pages can be judged only using anchor text, and most relevant pages fetched using CBP and SLC.

7. Conclusions

In this paper, we presented a novel focused Web crawling approach enhanced by CBP–SLC for domain-specific information retrieval guided by text classifier. The methods presented in this paper and the experimental results draw the following conclusions:

TFIDF does not take into account the difference of term IDF weighting in the positive and negative example sets when building text classifier. And, TFIPNDF can be considered to make up for the defect of TFIDF in text classification. And the experimental result proved that the performance of classifying using TFIPNDF weighting outperforms TFIDF for each dataset. We also built a weighted voting classifier by iteratively applying the SVM algorithm using TFIPNDF. Furthermore, for comparing the performance of the text classifying based on different techniques clearly, we also implement OCS and PEBL methods used for comparison. And, 1-DNFC effectively ensures and enhances the performance of classifying.

CBP–SLC method, which works with the DOM tree as opposed to raw HTML markup, makes Web page content more refined, and the highly relevant regions in an excessive clutter Web page are not obscured. Treating anchor text and its link-context respectively can both improve the efficiency and bring the error rate down caused by the misguidance of crawler. The comparison between using anchor text, link-context and content block partition also shows that many Web pages can be retrieved only using anchor text. Accordingly, partitioning the Web pages into smaller blocks and selectively using anchor and link-context improve the focused crawling performance significantly.

Acknowledgment

We are grateful to NIST for permitting and mailing RCV 1, which is distributed on two CDs, to us.

References

[1] A. Batzios, C. Dimou, A.L. Symeonidis, P.A. Mitkas, BioCrawler: an intelligent crawler for the semantic web, Expert Systems with Applications 35 (1-2) (2008) 524–530.
[2] P. Bedi, A. Thukral, H. Banati, A. Behl, V. Mendiratta, A multi-threaded semantic focused crawler, Journal of Computer Science and Technology 27 (6) (2012) 1233–1242.
[3] S. Brin, L. Page, The anatomy of a large-scale hypertextual Web search engine, Computer Networks and ISDN Systems 30 (1-7) (1998) 107–117.
[4] S. Chakrabarti, K. Punera, M. Subramanyam, Accelerated focused crawling through online relevance feedback, in: Proc. of WWW, 2002, pp. 148–159.
[5] S. Chakrabarti, M. van den Berg, B. Dom, Focused crawling: a new approach to topic-specific Web resource discovery, Computer Networks 31 (11-16) (1999) 1623–1640.
[6] H. Chen, M. Chau, D. Zeng, Ci spider: a tool for competitive intelligence on the web, Decision Support Systems 34 (1) (2002) 1–17.
[7] P.M.E. De Bra, R.D.J. Post, Information retrieval in the World Wide Web: making client-based searching feasible, in: Proceedings of the First International World Wide Web Conference, Geneva, 1994.
[8] F. Denis, R. Gilleron, M. Tommasi, Text classification from positive and unlabeled examples, in: Conference on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU), Annecy, France, 2002, pp. 1927–1934.
[9] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori, Focused crawling using context graphs, in: Proc. of VLDB, 2000, pp. 527–534.
[10] N. Eiron, K.S. McCurley, Analysis of anchor text for web search, in: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'03, 2003, pp. 459–460.
[11] E.J. Glover, K. Tsioutsiouliklis, S. Lawrence, D.M. Pennock, G.W. Flake, Using web structure for classifying and describing web pages, in: WWW'02 Proceedings of the 11th International Conference on World Wide Web, ACM Press, 2002, pp. 562–569.
[12] E.H. Han, Text Categorization using Weight Adjusted k-Nearest Neighbor Classification, PhD Thesis, University of Minnesota, 1999.
[13] H.W. Hao, C.X. Mu, X.C. Yin, S. Li, Z.B. Wang, An improved topic relevance algorithm for focused crawling, in: IEEE International Conference on Systems, Man and Cybernetics (SMC), Anchorage, AK, OCT 09-12, 2011, pp. 850–855.
[14] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalhaim, S. Ur, The shark-search algorithm-an application: tailored web site mapping, Computer Networks and ISDN Systems 30 (1-7) (1998) 317–326.
[15] M. Iwazume, K. Shirakami, K. Hatadani, H. Takeda, T. Nishida, Iica: an ontology-based internet navigation system, in: AAAI-96 Workshop on Internet Based Information Systems, 1996, pp. 65–71.
[16] P. Jin, H. Chen, X. Zhao, Indexing temporal information for web pages, Computer Science and Information Systems 8 (3) (2011) 711–737.
[17] J.J. Jung, Towards open decision support systems based on semantic focused crawling, Expert Systems with Applications 36 (2) (2009) 3914–3922.
[18] M. Khashei, A.Z. Hamadani, M. Bijari, A fuzzy intelligent approach to the classification problem in gene expression data analysis, Knowledge-Based Systems 27 (2012) 465–474.
[19] F. Letouzey, F. Denis, R. Gilleron, Learning from positive and unlabeled examples, in: Proceedings of the 11th International Conference on Algorithmic Learning Theory, Sydney, Australia, 2000, pp. 71–85.
[20] J. Li, K. Furuse, K. Yamaguchi, Focused Crawling by Exploiting Anchor Text using Decision Tree, WWW2005, Chiba, Japan, 2005, pp. 1190–1191.
[21] B. Liu, Y. Dai, X. Li, W.S. Lee, P.S. Yu, Building text classifiers using positive and unlabeled examples, in: Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), Melbourne, Florida, USA, 2003, pp. 179–188.
[22] N. Luo, W. Zuo, F. Yuan, C. Zhang, A New method for focused crawler cross tunnel, in: First International Conference of Rough Sets and Knowledge Technology 2006, Chongquing, China, Lecture Notes in Computer Science, vol. 4062, 2006, pp. 632–637.
[23] L.M. Manevitz, M. Yousef, One-class SVMs for document classification, The Journal of Machine Learning Research 2 (2002) 139–154.
[24] F. Menczer, R. Belew, Adaptive retrieval agents: internalizing local context and scaling up to the web, Machine Learning 39 (2-3) (2000) 203–242.
[25] N. Mendoza, A. Gago-Alonso, J.E. Medina-Pagola, Frequent approximate subgraphs as features for graph-based image classification, Knowledge-Based Systems 27 (2012) 381–392.
[26] S.A. Ozel, A web page classification system based on a genetic algorithm using tagged-terms as features, Expert Systems with Applications 38 (4) (2011) 3407–3415.
[27] P.F. Pai, M.F. Hsu, M.C. Wang, A support vector machine-based model for detecting top management fraud, Knowledge-Based Systems 24 (2) (2011) 314–321.
[28] G. Pant, Deriving link-context from HTML tag tree, in: Eighth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery, San Diego, CA, 2003.
[29] G. Pant, F. Menczer, Topical crawling for business intelligence, in: Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL), Trondheim, Norway, 2003.
[30] T. Peng, F. He, W. Zuo, C. Zhang, Adaptive topical web crawling for domain-specific resource discovery guided by link-context, in: 5th Mexican International Conference on Artificial Intelligence, Mexico, LNAI 4293, 2006, pp. 963–973.
[31] T. Peng, C. Zhang, W. Zuo, Tunneling enhanced by web page content block partition for focused crawling, Concurrency and Computation-Practice and Experience 20 (1) (2008) 61–74.
[32] B. Pinkerton, Finding what people want: experiences with the WebCrawler, in: The Second International WWW Conference, Chicago, USA, 1994.
[33] J. Qin, Y. Zhou, M. Chau, Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method, in: 4th ACM/IEEE-CS Joint Conference on Digital Libraries, Tucson, AZ, 2004, pp. 135–141.
[34] J. Ren, ANN vs. SVM: which one performs better in classification of MCCs in mammogram imaging, Knowledge-Based Systems 26 (2012) 144–153.
[35] G. Salton, C. Buckley, Term weighting approaches in automatic text retrieval, Information Processing and Management 24 (5) (1988) 513–523.
[36] M.C. Saunders, T.J. Sullivan, B.L. Nash, K.A. Tonnessen, B.J. Miller, A knowledge-based approach for classifying lake water chemistry, Knowledge-Based Systems 18 (1) (2005) 47–54.
[37] P. Soucy, G.W. Mineau, Beyond TFIDF weighting for text categorization in the vector space model, in: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005), 2005, pp. 1130–1135.

[38] K. Tateishi, H. Kawai, S. Akamine, K. Matsuda, T. Fukushima, Evaluation of web retrieval method using anchor text, in: Proceedings of the 3rd NTCIR Workshop, 2002, pp. 25–29.

[39] J.A. Torkestani, An adaptive focused Web crawling algorithm based on learning automata, Applied Intelligence 37 (4) (2012) 586–601.

[40] G.X. Xu, X. Gao, X. Zhang, X. Zhao, Improved TFIDF weighting for imbalanced biomedical text classification, in: International Conference on Energy and Environmental Science (ICEES), Singapore, 2011, pp. 2360–2367.

[41] B. Yu, Z.B. Xu, C.H. Li, Latent semantic analysis for text categorization using neural network, Knowledge-Based Systems 21 (8) (2008) 900–904.

[42] H. Yu, J. Han, K.C.C. Chang, 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, 2002, pp. 239–248.

[43] W. Zhang, T. Yoshida, X. Tang, Text classification based on multi-word with support vector machine, Knowledge-Based Systems 21 (8) (2008) 879–886.

[44] H.T. Zheng, B.Y. Kang, H.G. Kim, An ontology-based approach to learnable focused crawling, Information Sciences 178 (23) (2008) 4512–4522.