International Journal of Software Engineering
and Knowledge Engineering
Vol. 25, No. 1 (2015) 147–168

DOI: 10.1142/S0218194015500011

World Scientific
www.worldscientific.com

Clustering-Based Topical Web Crawling for Topic-Specific Information Retrieval Guided by Incremental Classifier

Tao Peng* and Lu Liu

College of Computer Science and Technology
Jilin University, Changchun 130012, China

Department of Computer Science,
University of Illinois at Urbana-Champaign, 201 N Goodwin Avenue
Urbana, IL 61801, USA

Key Laboratory of Symbol Computation and
Knowledge Engineering of the Ministry of Education
Changchun 130012, China
**taopeng@illinois.edu*

Received 4 March 2013
Revised 3 May 2013
Accepted 29 June 2014

Today more and more information on the Web makes it difficult to get domain-specific information due to the huge amount of data sources and the keywords that have few features. Anchor texts, which contain a few features of a specific topic, play an important role in domain-specific information retrieval, especially in Web page classification. However, the features contained in anchor texts are not informative enough. This paper presents a novel incremental method for Web page classification enhanced by link-contexts and clustering. Directly applying the vector of anchor text to a classifier might not get a good result because of the limited amount of features. Link-context is used first to obtain the contextual information of the anchor text. Then, a hierarchical clustering method is introduced to cluster feature vectors and content unit, which increases the length of a feature vector belonging to one specific class. Finally, incremental SVM is proposed to get the final classifier and increase the accuracy and efficiency of a classifier. Experimental results show that the performance of our proposed method outperforms the conventional topical Web crawler in *Harvest rate* and *Target recall*.

Keywords: Web page classification; link-context; CFu-tree; comparison variation (CV); clustering; incremental SVM.

1. Introduction

With the explosive growth of text information (e.g. news, sports, games, and forum discussions) on the Web, more and more individuals and organizations pay attention

*Corresponding author.

to the Web pages and text information that focus on a specific topic. However, finding and monitoring text information on the Web and distilling the information that belongs to the topic that what we want is still a practical and important task. Besides, the proliferation of diverse sites increases the difficulty of obtaining information on a specific topic. So, one appropriate way of organizing this overwhelming amount of documents is necessary.

Probably the most common approach to analyzing big or complex data is called classification or categorization. A host of classification methods have been used to different areas (e.g. text indexing, document sorting and text filtering, and Web page categorization) [12]. Also, Web page categorization is one of the most common text categorization applications that many classification methods have made contributions to, such as Naïve Bayesian Classification [25], Rocchio [19], SVM [42], decision trees [32], associate rules [21], and so on. Compared with pure-text classification, Web page classification is more difficult because of the noisy information on the Web. For instance, the complex structure makes the texts of Web page harder to classify. Also, some texts are very concise, which means a long article can be replaced by just a few words. As a result, we may not be able to get a good classification performance. In such cases, it would be helpful if we group them into some clusters first.

To tackle this problem, a novel clustering-based topical Web crawling approach guided by classifier can be applied to organize data for domain-specific information retrieval. Classification and clustering algorithms are heavily used in most modern search engine [9]. Describe abstractly, classification, also called categorization, is a supervised process which applies labels to data, such as sorting news in Web pages [33], filtering spam [23], classifying multimedia information [17], analyzing users' opinion [15, 18] and so on. Different with classification, the task of clustering is an unsupervised process which classifies objects into groups called clusters. There are many clustering algorithms such as K-means (e.g. ISODATA [35] and bisecting K-means [16]), hierarchical clustering [13] and many applications about clustering such as information fusion, eliminating redundancy and so on.

Anchor text, which is similar to user's query, tends to be very short and probably succinctly describes the topic of a Web page. Link-context is the critical contextual information of anchor text for topical Web crawling. So, link-context is extracted to build feature vector of the information in the Web page. Due to the limited amount of information, some link-context, which belongs to the specific topic, is misjudged by the classifier. In this paper, the clustering approach, in combination with our classifier, is presented to make domain-specific information retrieval more accurate, that is, clustering-based Web crawling could obtain more information in Web pages that is related to the specific domain. Given an example, in a Web page, "click here" as an anchor text, can not reflect the topic of its target Web page. "Football and boxing…" and "football and basketball…" are both the link-context of "click here" and they belong to the "sport" class. Assuming that the feature vector built by the classifier does not contain "boxing" or some more words of the sentence "football and boxing…", then the sentence "football and boxing…" may not be classified to the

positive example set. Suppose that the feature vector built by the classifier contains "football, basketball" and some more words of the sentence "football and basketball…". Also, the sentence "football and basketball…" is classified to the positive example set. After clustering the two sentences, the feature vector of the two sentences may belong to the "sport" class. So, the anchor text, which should belong to the "sport" class, would not be classified mistakenly due to fewer features. This method can be applied to text classification, especially Web page classification

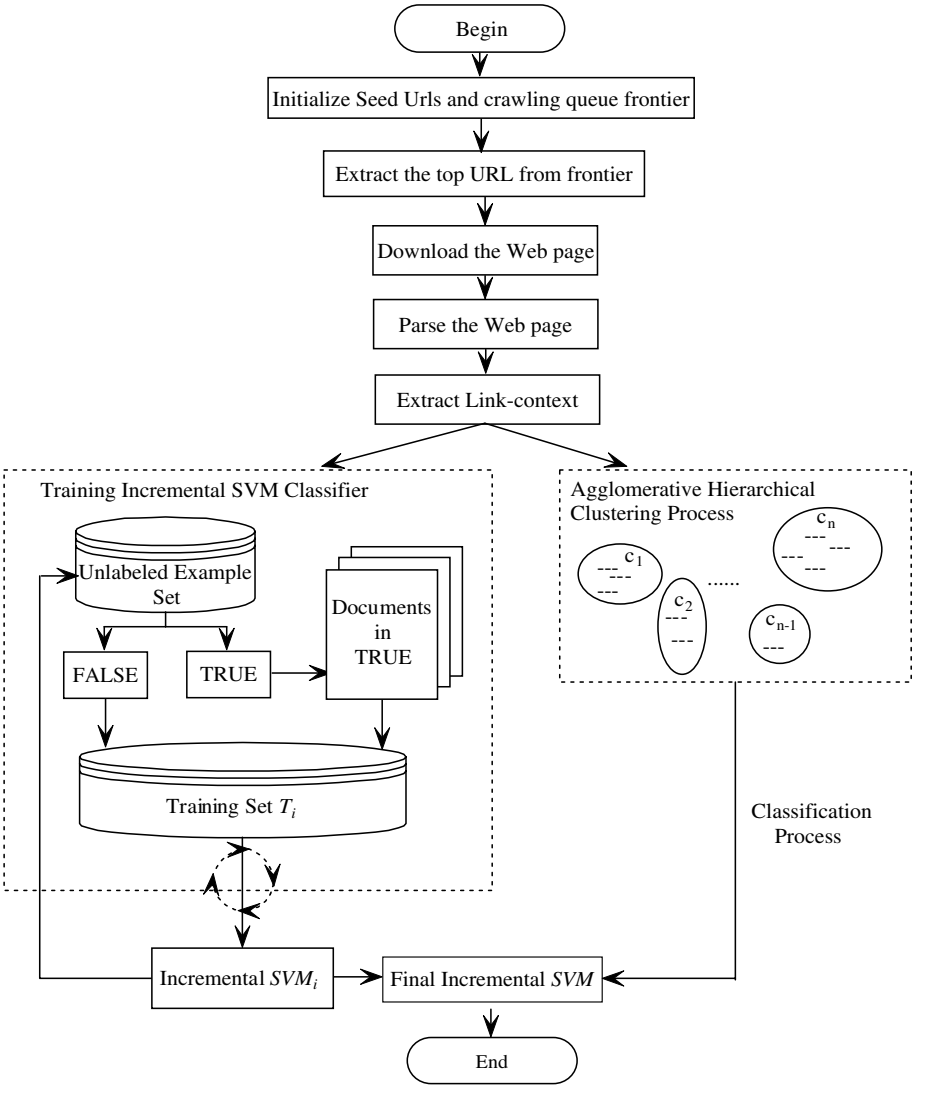

Fig. 1. The framework of clustering-based topical crawler guided by the incremental classifier.

because there are lots of anchor texts which contain almost fewer than 10 words. Clustering, in combination with classification, is a novel method to topical Web crawling. Since the number of clusters obtained by clustering is not fixed, we may find some categories that are not given by the classifier.

The training set of the classifier may be very large, which could spend lots of time and space training the final classifier. Incremental learning can not only choose the training samples heuristically, but also solve the problem that training process would cost too much time and space. The traditional SVM classifiers add all the samples to the training process. However, in this paper, we delve into the support vectors which are important factors to the final classifier. Only applying support vectors and some heuristic samples to the training process, also called incremental training process, can achieve good classification performance. Then, the final classifier is built by iteratively using incremental SVM, which is a new idea of this paper. Incremental SVM is similar to transfer learning, that is, it also can deal with the situation that training and future data does not have the same distribution. In the hyperplane, the support vectors play an important role on classification. Classifying the training example set is replaced with classifying the set of support vectors. When future data come, future data, in combination with the original set of support vectors, could get the new hyperplane, which is suitable for the future data. The framework of clustering-based topical crawler is shown in Fig. 1.

The outline of this paper is organized as follows. We review the related work in Sec. 2. Section 3 illustrates how to extract link-context from the Web pages to guide topical Web crawling. The hierarchical clustering process is proposed in Sec. 4. The incremental method used to build the final classifier is proposed in Sec. 5. Several comprehensive experiments are performed to evaluate the effectiveness and the efficiency of our method in Sec. 6. Section 7 draws the conclusions.

2. Related Work

In the past decade, a considerable amount of related research has been done in academia. In this section, we first review some previous works on topical Web crawling. After that, we discuss the related issues of link-context and clustering analysis. Finally, we will see the key issues about incremental techniques.

Research on topical Web crawling has attached much attention in data mining. Also, researchers have come up with many different kinds of methods for topical Web crawling [3, 4, 20, 30], such as probabilistic model, ontology model, support vector model, and so on. In probabilistic models, Liu and Milios [22] proposed two probabilistic models (MEMM and CRF) which captured sequential patterns along paths leading to the targets. It does not have the advantage of being a simple and intuitively appealing method for retrieving the target pages. However, once this kind of method provides a clear statement about the assumptions, it also can be a powerful model and have good performance. Mouton and Marteau [26] exploited link analysis to improve topical Web crawling, which is also a probabilistic model. The topical

graphs on the Web could guide the crawler toward highly relevant areas of the Web. Some researchers use ontology to get rid of the irrelevant information with the topic [11, 34]. In support vector model, Pant and Srinivasan [28] applied link contexts and support vector machine (SVM) to build topical Web crawlers. The goal of using link contexts is to obtain more semantic information of anchor texts. SVM also gets good performance. Compared with [28], in this paper, we use clustering technique to cluster link-context and get more semantic information, which has got good performance. Also, SVM is improved to incremental SVM. We analyze our results, which show that our proposed method is effective and efficient.

One word can have many different meanings. For example, state represents a country or a verb like "represent". Similarly, desert could be a nature landscape or discarding somebody or something. People could tell the difference through the context of the word, that is, the words that surround it. Link-context, as the words that appear in the text around a hyperlink in a Web page, has also been studied for many years. Pant and Srinivasan [28] investigated the effects of various definitions of link-contexts. Also, they tried to find how many link-contexts a crawler obtained could have the best crawling performance. Tian *et al.* [37] proposed a contextual dependency network (CDN) which made use of link-context and a dependency function to get rid of irrelevant links that did not provide useful or predictive information. Tang *et al.* [36] applied link-contexts to ranking the web pages. Researchers use many different methods to extract link-contexts. Kumar *et al.* [5] proposed a rule-based method to extract link-contexts. They categorized anchor texts into various concepts, such as named entity, class name and so on. And then SLR parser was applied to extract precise link-context of the Web page. In this paper, we use DOM tree to extract link-context, which is easier.

Clustering, as an approach to organize various data, has been applied to many different areas, especially in text information retrieval. Bouras and Tsogkas [5] proposed a clustering technique, the enhancement of standard k-means using the external knowledge from WordNet, for news articles. The implement of WordNet could add more semantic information. However, k-means fixed the number of clusters to some extent. Having Wordnet combine with k-means could indeed provide better results when it came to efficiency. Ma *et al.* [24] applied clustering to group research project selection. Cota *et al.* [10] presented a heuristic-based hierarchical clustering method to deal with name ambiguity. The method fuses clusters of citations of similar author names based on several heuristics and similarity measures on the components of citations. Clustering technique is also used in social bookmarks [39], detecting data records [14], categorization of texts [41], and so on. Clustering, in combination with classifier, applied to topical Web crawling, is a new research area and a main contribution in this paper.

Incremental learning is a machine learning paradigm where the learning process takes place whenever new examples emerge and adjusts what has been learned according to the new examples [1]. Ade and Deshmukh made a survey and defined five criteria for incremental learning algorithm [1]. Besides, researchers come up with

lots of different incremental algorithms and apply them to various research areas. Totad *et al.* [38] used a batch incremental processing algorithm to obtain a FP-tree, which took less time for constructing FP-tree. Zhang *et al.* [43] introduced an incremental method to decision support systems (DSS). The incremental method grouped the obtained results into equivalence classes so that the time and space complexity could be improved. Incremental technique is always applied to many research areas to improve efficiency. Incremental SVM is introduced in this paper to increase the efficiency and accuracy of classification.

3. Extracting Link-Context

Anchor text, as a powerful navigation in the Web page, could not only help people browse the Internet, but also understand the semantic information of the Web page. For example, the word "cnn" could probably be the anchor text of the link http://edition.cnn.com/. It is easy to understand that "cnn" is a short description of the Web page that "http://edition.cnn.com/" links to. However, anchor text is usually very short and not informative enough to well illustrate the topic of the Web pages. So, it could be expanded to words around it, that is link-context. Link-context, which appears around a clickable hyperlink (anchor text), has been applied to a number of Web resource discovery and information retrieval applications such as categorization, search engine, digital libraries and so on. In one Web page, texts and links (URLs and hyperlinks) about the same topic are usually grouped into one content block, also called content unit which is assumed to be a rectangle shape and be closely packed in original Web page. The size of the content unit is varied. The big one may cover the whole web page, while the small one only takes 1/8 or 1/16 of the web page's total space, and the smallest one is an anchor text. Each HTML page has a corresponding DOM tree which provides each Web page with a pre-defined syntactic structure where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes (as shown in Fig. 2).

Before we derive link-context from DOM tree, we tidy the HTML page into a well-formed one web page beforehand using HTML TIDY tool[a] because many Web pages are badly written. 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> $\cdots$ </text> tags appear at leave nodes on the tag tree. This preprocessing simplifies the analysis.

We use the method of deriving link-context from Aggregation Nodes [27], with some modifications. 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 (shaded in Fig. 2). 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

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

```
<html>
<head>
<title>CS@Illinois | Department of Computer Science at Illinois</title>
</head>
<body>
<h1 class="cs">Department of Computer Science</h1>
<a href="http://cs.illinois.edu/csillinois/overview">
<text>Overview</text></a>
<p>
<a href="/csillinois/history"><text>History</text></a>
<text>have a look at awards<a href="/csillinois/awards">
<text>Read more</text></a></text>
<h3>Connect with Us</h3>
</p>
</body>
</html>
```

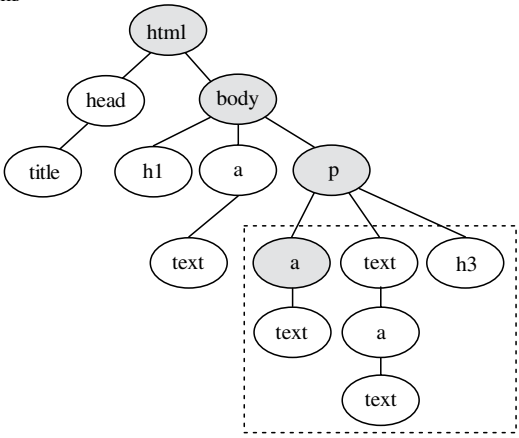

Fig. 2. An HTML page and its corresponding tag tree.

anchor (showed in rectangle of Fig. 2). If one anchor appears in many different blocks, combine the link-context in every block as its full link-context.

4. Clustering Process

In this paper, a novel conceptual clustering method is used for finding and characterizing each cluster. Our method is an agglomerative hierarchical clustering method, which uses a *bottom-up* strategy. That is, it starts by letting each object be a cluster and iteratively merges clusters into larger clusters, until certain termination conditions are satisfied. The similarity between clusters C_i and C_j is defined as Eq. (1) below so that the similarity calculation can be standardized, that is, the range of values allowed for the similarity is from 0 to 1.

$$Sim(F_i, F_j) = 1 - \sqrt{\frac{\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^2}{2}}, \quad (i \neq j). \tag{1}$$

To measure whether two closest pairs of clusters can be merged, we build a binary model, also known as a *Comparison Variation* ($CV(C_i, C_j) \in \{0, 1\}$), to reach the goal of quantitative calculation, as shown in Eq. (2) below.

$$CV(C_i, C_j) = \begin{cases} 1, & \left(\prod_{k=1}^{n} (1 - Sim(u_k, C_j))^{\lambda} \leq \max\{Sim(u_k, C_j) | u_k \in C_i\} \right) \\ 0, & (\text{otherwise}) \end{cases} \quad (2)$$

where C_i and C_j are the closest pair of clusters, and $\{u_k\}$ is the collection of content units which belong to cluster C_i. $Sim(u_k, C_j)$ is the clusters similarity calculated by Eq. (1). We compare $\prod_{k=1}^{n} (1 - Sim(u_k, C_j))^{\lambda}$ and all $Sim(u_k,C_j)$ to select the maximum. If a $Sim(u_k, C_j)$ is the largest, i.e. $CV\,(C_i, C_j) = 1$, then C_j is similar with u_k, and C_i and C_j can be merged. However, if $\prod_{k=1}^{n} (1 - Sim(u_k, C_j))^{\lambda}$ is the largest, i.e. $CV\,(C_i, C_j) = 0$, then C_j is not similar with any u_k, and C_i and C_j can not be merged. λ ($\lambda > 0$) is a weight for reflecting the relative importance of similarity value when merging clusters. Obviously, the higher the value of λ is, the lower the similarity value of merging clusters is required. In other words, if $\lambda \to 0$, $\prod_{k=1}^{n} (1 - Sim(u_k, C_j))^{\lambda}$ is close to 1, then each content unit is a cluster. However, if λ is infinitely great, $\prod_{k=1}^{n} (1 - Sim(u_k, C_j))^{\lambda}$ is close to 0, then all the content units will merge into a whole cluster. For different specific similarity value request, an appropriate λ is selected.

Defining *Comparison Variation* can reduce the difficulty of setting up similarity threshold, which is uncertain. We need to perform multiple experiments to test different results calculated by different thresholds, then we can get a proper threshold. In addition, the threshold can not change with the method of calculating similarity. While *Comparison Variable* is calculated by the calculating similarity method and it changes with the method, so it is a reasonable metric.

Initially, each content unit is treated as a single cluster after preprocessing (eliminating *stopwords* and stemming) and feature extraction. At the beginning, every CFu-tree only has one node representing a cluster. Then we add every CFu-tree into the *arrayList*. Clusters are merged or spilt according to the similarity criterion after scanning the CFu-tree and calculating the similarity. The scanning process is repeated until clusters do not change. Each cluster is treated as a tree including its information.

The CFu-tree is built dynamically as a single cluster (also is a node) is inserted. Thus, the method is incremental. The node is inserted into the closest leaf node or non-leaf node. If the similarity between any nodes is less than the requirement after insertion, then the node or other nodes are split. After the merging of the new nodes, clustering features and the set of content units of the root node will be also merged.

A CFu-tree represents a cluster, even though a single node. A leaf node represents a document which is as a sub-cluster. A non-leaf node, which represents cluster C_i, in the CFu-tree has descendants which merge into C_i. The non-leaf nodes store sums of the cluster features and documents of their children, and thus summarize clustering information about their children.

Initially, our method places each document into a cluster of its own. The clusters are then merged step-by-step according to the criterion. The major steps of clustering represented by a CFu-tree include the following:

Step 1. A document is viewed as a single cluster C_i. After preprocessing and feature extracting, a *CFu-tree* is built including only one node, that is C_i.
Step 2. Add the *CFu-tree* to the *CFu-tree* list, and cluster for each pair of clusters.
Step 3. Scan the*CFu-tree* list to calculate the similarity.
Step 4. Merge or split clusters according to similarity criteria.
Step 5. Repeat *Step 3* until no cluster meets the conditions.

The CFu-tree is built dynamically when a single cluster (also is a node) is inserted. Thus, the method is incremental. The node is inserted into the closest leaf node or non-leaf node. If the similarity between any leaf nodes after insertion is less than the requirement, then the leaf node and possibly other nodes are split. After the merging of the new nodes, clustering features and the set of documents of the root node will be also merged.

5. Building Incremental Classifier for Topical Web Crawling

In this paper, the text classifier is mainly used to judge the topic relevance of Web pages obtained by clustering. Web page classification not only plays an important role in topical Web crawling, but also is widely used in digital library, personal information retrieval, and search engine, and so on. Today's Web search engines incorporate hundreds of documents and hyperlinks (e.g. anchor text). Anchor texts relating to page contents are undoubtedly important analysis during the crawling process. Since the number of the features of anchor text is few, the features are clustered first, and then can be classified by the classifier.

After applying stopwords removal, feature selection, term weighting, in this section, we mainly introduce the incremental SVM training process. This incremental SVM modifies SVM [8] formulation so that the final classifier has higher precision. In this paper, our previous work 1-DNFII algorithm [29], which considers both the diversity of the feature frequency in positive feature set and unlabeled example set and the absolute frequency of the feature in positive feature set, is used for collecting the positive and negative feature sets. If the frequency of occurrence of a feature in the positive example set is more frequently than in the unlabeled example set, then this feature is regarded as a positive example. If the documents in the unlabeled example set do not contain any feature in the positive feature set, the documents can be regarded as reliable negative examples. First, the initial set of positive and negative examples (P_0 and RN_0) are chosen from the training set T, in which all the samples are labeled beforehand. P_0 and RN_0 constitute the initial training sample set T_0. The initial SVM_0 is run using P_0 and RN_0. That is, the initial hyperplane between P_0 and RN_0 is determined. And at the same time, the set of

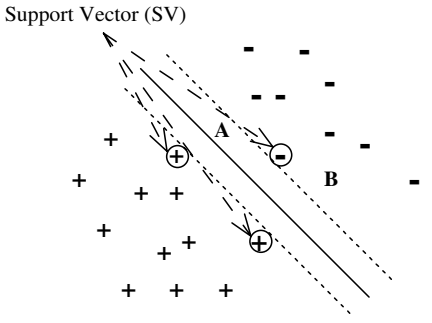

Fig. 3. The hyperplane and support vector (SV).

support vectors (AV_0) is obtained. As Fig. 3 shown, two positive examples(+) and one negative example(−) are all support vectors and they make up the set of support vectors. Then, the training set T can be classified using SVM_0 and we can get the correct classification results (identified by $TRUE$) and the wrong classification results (identified by $FALSE$) because all the training examples have their labels. Before classifying, each document is represented as a vector $(x_1 : w_1; x_2 : w_2; \ldots; x_i : w_i; \ldots; x_n : w_n)$, where x_i is the ith feature in the document and w_i is the corresponding weight of the feature i. Once they are classified by a classifier, the correctness of their labels can be verified. If the set of wrong classification results is not null, then it means the samples in this set and the samples in $TRUE$ are not in the same side of the hyperplane. So, when training the incremental SVM again, adding the samples in $FALSE$ could affect the original set of support vectors, which is a positive factor for the final classifier. Also, as for the correct classification results, once the features of a document are not clear, which means this document might not be classified correctly next time, then this kind of documents also should be added to the training set for the next training process (identified by $DARK$). The evaluation value is calculated by Eq. (3) as follows.

$$Q(d_j) = \sum_{i=1}^{n} \frac{N(T_{ij})}{N(T_i)} \tag{3}$$

where $N(T_{ij})$ is the number of word i in document j. $N(T_i)$ is the number of word i in the training example set. n is the number of words in the training example. For the documents which are classified correctly, $\mu\%$ documents in $TRUE$ are selected according to the Q value. That is, we choose the document whose Q value is low.

Now we get the training example set $T_1 = AV_0 \cup FALSE \cup DARK$. Repeat the mentioned process above until the set $FALSE$ is equal to null. Therefore, for the ith training, the training example set would be $T_i = AV_{i-1} \cup FALSE \cup DARK$. And the algorithm of incremental SVM training process is described as follows.

Algorithm 1. Incremental SVM training process.

Algorithm of **Incremental SVM training process**
Input: RN_0: the initial set of negative examples, P_0: the initial set of positive examples U: the set of unlabeled examples
Output: f: final SVM classifier

1. **procedure Incremental SVM training process**
2. $RN \leftarrow RN_0$, $P \leftarrow P_0$, $SVM_0 \leftarrow$ trainSVM(RN, P)
3. $i \leftarrow 1$
4. TRUE $\leftarrow$ the examples in U that are classified by SVM_0 correctly
5. FALSE $\leftarrow$ the examples in U that are not classified by SVM_0 correctly
6. **while** FALSE $\neq$ NULL
7. DARK $\leftarrow$ choose Q samples from TRUE
8. $SVM_i \leftarrow$ trainSVM(AV_{i-1}, FALSE, DARK)
9. TRUE $\leftarrow$ the examples in U that are classified by SVM_i correctly
10. FALSE $\leftarrow$ the examples in U that are not classified by SVM_i correctly
11. $i \leftarrow i + 1$
12. **end while**
13. $f \leftarrow SVM_{i-1}$
14. **end procedure**

In automatic text classification, applying appropriate examples to the further training process could reduce much time and work for evaluating them. SVM, as a learning algorithm, could achieve very good classification performance under the condition of the number of training examples is small. As shown in Fig. 3, the final

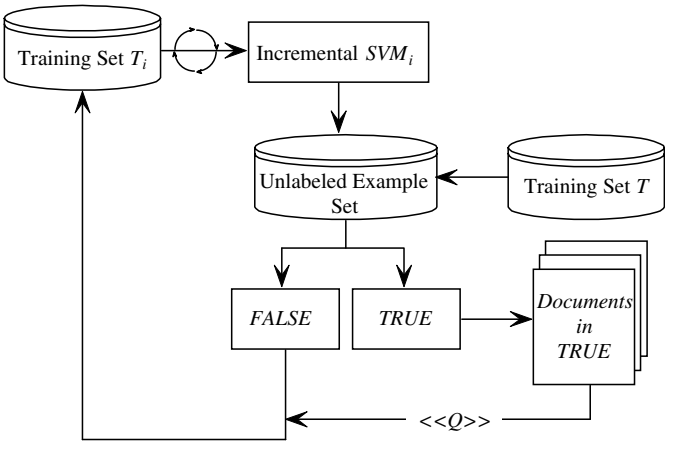

Fig. 4. The process of incremental SVM training process.

classifier mainly depends on the support vectors, that is, the points in the circle. So, the support vectors can describe the features in the training example set. The partition of the support vectors is equal to the partition of the whole training example set. In this paper, classifying the training example set is replaced with classifying the set of support vectors, which can save time and space, and does not affect the classification accuracy. Usually, the support vectors are only a small portion of the training set. If the incremental samples contain the classification information which is not included by the original training set, then the support vectors must change to reflect the influence brought by the incremental samples which are not classified correctly by the original classifier. Through the study of the set of support vectors, incremental learning is effective and efficient. Figure 4 illustrates the process of incremental SVM training process.

6. Experiments and Results

This section is mainly about the examination of the effectiveness and efficiency of our proposed method with the existing experimental data. After introducing some common performance metrics and the datasets we used, first, the clustering performance is evaluated to get the value of parameter lambda. In this setting, clustering could achieve its best performance. Then, several comparisons are given to show that our clustering method is feasible. Finally, we use Open Directory Project (ODP, http://www.dmoz.org) and Yahoo! Directory as our datasets to test our incremental SVM method. Through different iteration numbers, the growth of the F-Measure also validates our proposed method.

6.1. *Performance metrics*

The two most common effectiveness measures, *Harvest rate* and *Target recall*, are introduced to summarize and compare search results. Intuitively, *Harvest rate* is the fraction of crawled pages that are relevant to the topic, which measures how well it is doing at rejecting irrelevant Web pages. *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 can be 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:

$$Harvest\ rate = \frac{|C(t) \cap T|}{|C(t)|} \times 100\% \tag{4}$$

$$Target\ recall = \quad \frac{|C(t) \cap T|}{|T|} \times 100\% \tag{5}$$

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

$$F\text{-}Measure = \frac{(\beta^2 + 1) Harvest\ rate \times Target\ recall}{\beta^2 \times Harvest\ rate + Target\ recall} \times 100\% \tag{6}$$

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

Next, to demonstrate the effectiveness of clustering, we choose *Precision*, *Recall*, and *F-Measure* to evaluate the clustering performance. *Precision* for clustering is the fraction of objects assigned that belong to an explicit class, which measures how well it is doing at rejecting irrelevant class. Let $C = \{C_1, \ldots, C_r\}$ be a set of clusters after clustering, and $K = \{K_1, \ldots, K_t\}$ be the set of the true classes of the collection. For each cluster C_i, $MaxC_i(K_j)$ is the set of objects associated with class label K_j in C_i, and the number of $MaxC_i(K_j)$ is the most in the set of C. i.e.

$$|MaxC_i(K_j)| = \max\{|C_i \cup K_j||C_i \in C, K_j \in K\}. \tag{7}$$

So, the precision of cluster C_i is shown as follows:

$$Precision(C_i) = \frac{|MaxC_i(K_j)|}{|C_i|}. \tag{8}$$

Recall is the proportion of objects associated with label K_j and all of the K_j class, which measures how well it is doing at finding all the objects in cluster with the label K_j. Similarly, we define *Recall* as follows:

$$Recall(C_i) = \frac{|MaxC_i(K_j)|}{|K_j|}. \tag{9}$$

$C_{F\text{-}Measure}$, 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 $C_{F\text{-}Measure}$ as follows:

$$C_{F\text{-}Measure} = \frac{(\beta^2 + 1) Precision \times Recall}{\beta^2 \times Precision + Recall}. \tag{10}$$

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.

6.2. *The comparison of different techniques*

First, we define a variable lambda in Sec. 4 to adjust the relative importance of similarity value when merging clusters. As we mentioned before, the range of $Sim(u_k,C_j)$ is from 0 to 1, and the range of 1-$Sim(u_k, C_j)$ is also from 0 to 1, the higher the value of λ is, so the lower the similarity value of merging clusters is required. Before comparing the efficiency and accuracy of automatically classifying Web pages, we first analyze how the different *lambda* values impact on the performance of topical Web crawling. In the experiment, we use two online datasets: ODP and Yahoo! Directory to test the effectiveness of our proposed method. Also, 10 threads are applied to provide for reasonable speed-up. *Lambda* is increased from 0.2 to 6 during the process of crawling 10 topics covering thousands of pages for each topic. Figure 5 shows the average *Harvest rate*, *Target recall* and *F-Measure* on two datasets. With the increase of λ, *Harvest rate* decreases, *Target recall* slowly increases and *F-Measure* first increases then decreases. It means that the overall performance first increases and then decreases. Experimental results show *F-Measure* reaches its peak when λ is around 3. So, we assign the corresponding lambda as a constant 3 in the remainder of the experiments.

In what follows, we analyze our cluster performance using three metrics *Precision*, *Recall* and $C_{F\text{-}Measure}$.

For comparing the performance of the text clustering based on different techniques clearly, $C_{F\text{-}Measure}$ values for each topic on two datasets are shown in Fig. 6. We observed that the average $C_{F\text{-}Measure}$ values of our method outperform HAC [13] and K-means [16] on ODP and Yahoo! Directory datasets. In summary, our method shows significant performance improvement over HAC and K-means. Consequently, the experimental results prove that our method is effective and feasible.

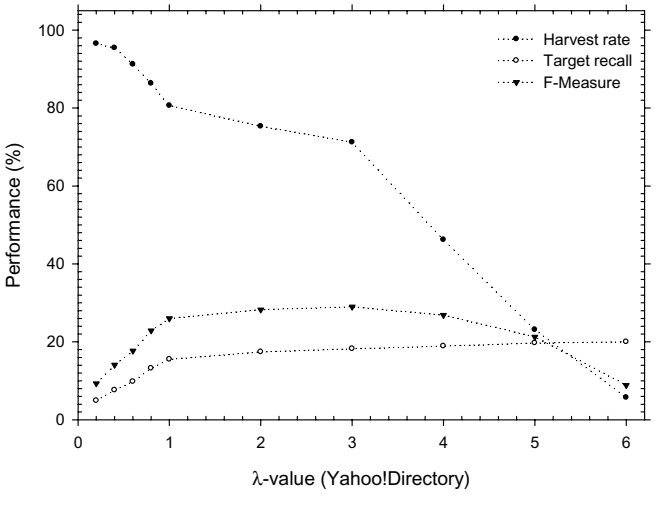

Fig. 5. Dynamic plot of average *Harvest rate*, *Target recall* and *F-Measure* versus weight λ on ODP and Yahoo!Directory. Performance is averaged across ten topics of each dataset.

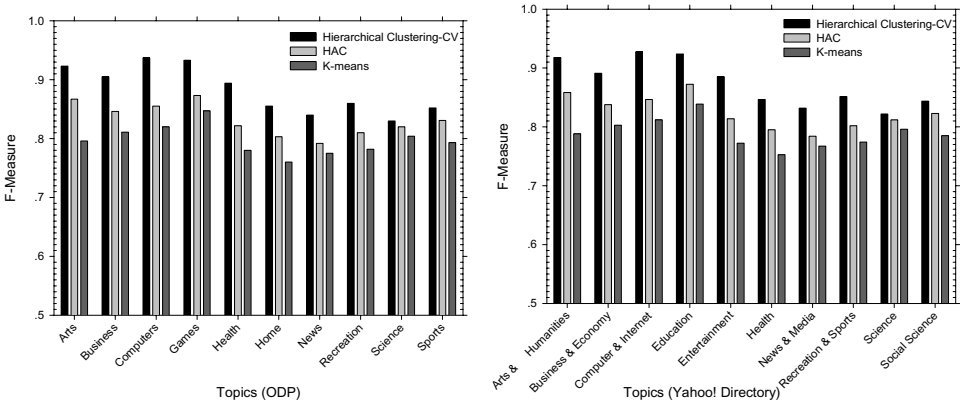

Fig. 6. The *F-Measure* of HAC, K-means and our proposed clustering method on two datasets.

We tested our incremental SVM algorithm on classification after clustering from 10 categories in ODP and Yahoo! Directory. For each class, we built a classifier using incremental SVM, and the iteration is up to six times. After six iterations, the *F-Measure* is close to reality. Tables 1 and 2 show the *F-Measures* on two datasets after each iteration. It can be seen that for each topic, *F-Measure* keeps increasing when another iteration happens. It is because that the number of samples is limited at the beginning. With the increasing number of the incremental samples and the support vector which is more and more closed to its true hyperplane.

Table 1. *F-Measures* on ODP after each iteration.$\mu = 10\%$.

	Arts	Business	Computers	Games	Health	Home	News	Recreation	Science	Sports
SVM_0	0.8516	0.8239	0.8227	0.8718	0.8968	0.7959	0.8258	0.8152	0.8111	0.8464
SVM_1	0.8708	0.8420	0.8415	0.8881	0.9141	0.8079	0.8363	0.8258	0.8294	0.8622
SVM_2	0.8833	0.8546	0.8602	0.8995	0.9226	0.8269	0.8484	0.8440	0.8413	0.8733
SVM_3	0.8958	0.8661	0.8779	0.9121	0.9361	0.8498	0.8622	0.8661	0.8531	0.8856
SVM_4	0.9073	0.8814	0.9066	0.9313	0.9543	0.8748	0.8776	0.8872	0.8641	0.9042
SVM_5	0.9322	0.9054	0.9551	0.9610	0.9802	0.9167	0.8989	0.9121	0.8878	0.9330

Table 2. *F-Measure* on Yahoo! Directory after each iteration. $\mu = 10\%$.

	Arts & Humanities	Business & Economy	Computer & Internet	Education	Entertainment	Health	News & Media	Recreation & Sports	Science	Social Science
SVM_0	0.8408	0.8296	0.8141	0.8600	0.8847	0.8028	0.8160	0.8074	0.8008	0.8350
SVM_1	0.8634	0.8453	0.8328	0.8826	0.9046	0.8160	0.8312	0.8189	0.8223	0.8569
SVM_2	0.8720	0.8538	0.8411	0.8914	0.9136	0.8242	0.8395	0.8271	0.8305	0.8655
SVM_3	0.8896	0.8709	0.8580	0.9093	0.9320	0.8407	0.8564	0.8437	0.8472	0.8829
SVM_4	0.8985	0.8796	0.8666	0.9184	0.9413	0.8491	0.8650	0.8522	0.8557	0.8917
SVM_5	0.9230	0.8964	0.9456	0.9515	0.9705	0.9076	0.8900	0.9031	0.8790	0.9238

Table 3. Different μ values versus *F-Measures* on ODP over each topic.

	Arts	Business	Computers	Games	Health	Home	News	Recreation	Science	Sports
$\mu = 10\%$	0.9322	0.9054	0.9551	0.9610	0.9802	0.9167	0.8989	0.9121	0.8878	0.9330
$\mu = 20\%$	0.9050	0.8790	0.9273	0.9330	0.9517	0.8900	0.8727	0.8855	0.8619	0.9058
$\mu = 30\%$	0.8787	0.8534	0.9003	0.9058	0.9239	0.8641	0.8473	0.8597	0.8368	0.8794
$\mu = 40\%$	0.8531	0.8286	0.8741	0.8795	0.8970	0.8389	0.8226	0.8347	0.8125	0.8538
$\mu = 50\%$	0.8282	0.8044	0.8486	0.8538	0.8709	0.8145	0.7987	0.8104	0.7888	0.8290
$\mu = 60\%$	0.8041	0.7810	0.8239	0.8290	0.8455	0.7908	0.7754	0.7868	0.7658	0.8048
$\mu = 70\%$	0.7807	0.7583	0.7999	0.8048	0.8209	0.7677	0.7528	0.7639	0.7435	0.7814
$\mu = 80\%$	0.7580	0.7362	0.7766	0.7814	0.7970	0.7454	0.7309	0.7416	0.7219	0.7586
$\mu = 90\%$	0.7288	0.7079	0.7467	0.7513	0.7663	0.7167	0.7028	0.7131	0.6941	0.7294

In Sec. 5, a parameter μ is defined to select $\mu\%$ documents in TRUE. These documents and the wrong classification results are all used to the next training process. Experimental results show that using less training examples could obtain the same or even higher accuracy compared with a large number of training examples, which achieves a reduction of training examples. Also, lower μ equate to better classification performance in tables showing quantitative comparisons. However, it needs more training times, which means the computational complexity is high. Considering the classification accuracy and computational complexity, an appropriate μ is needed to conduct Web page classification. Tables 3 and 4 show the different μ values versus *F-Measures* on two datasets over each topic. And μ is 10% in this paper.

For comparing the performance of Web page classification based on different techniques clearly, *F-Measures* of each topic on two datasets are shown in Fig. 7. We observed that the *F-Measures* of our proposed incremental SVM method outperform PSOC [29] and SVM which are two typical classification algorithms. Besides, without clustering, only using classifier to guide the topical crawling could get the results in shorter time. However, the performance of combining clustering and classifier is much better than only using classifier. Standard SVM has $O(n^3)$ time complexity, where n is the number of training examples [40]. PSOC is a little more complex than standard SVM, and the time complexity of PSOC is about seven or

Table 4. Different μ values versus *F-Measures* on ODP over each topic.

	Arts & Humanities	Business & Economy	Computer & Internet	Education	Entertainment	Health	News & Media	Recreation & Sports	Science	Social Science
$\mu = 10\%$	0.9230	0.8964	0.9456	0.9515	0.9705	0.9076	0.8900	0.9031	0.8790	0.9238
$\mu = 20\%$	0.8961	0.8703	0.9181	0.9238	0.9422	0.8812	0.8641	0.8768	0.8534	0.8969
$\mu = 30\%$	0.8700	0.8449	0.8913	0.8969	0.9148	0.8555	0.8389	0.8513	0.8285	0.8708
$\mu = 40\%$	0.8447	0.8203	0.8654	0.8708	0.8881	0.8306	0.8145	0.8265	0.8044	0.8454
$\mu = 50\%$	0.8201	0.7964	0.8402	0.8454	0.8623	0.8064	0.7908	0.8024	0.7810	0.8208
$\mu = 60\%$	0.7962	0.7732	0.8157	0.8208	0.8372	0.7829	0.7677	0.7790	0.7582	0.7969
$\mu = 70\%$	0.7730	0.7507	0.7919	0.7969	0.8128	0.7601	0.7454	0.7563	0.7361	0.7737
$\mu = 80\%$	0.7505	0.7289	0.7689	0.7737	0.7891	0.7380	0.7237	0.7343	0.7147	0.7511
$\mu = 90\%$	0.7216	0.7008	0.7393	0.7439	0.7588	0.7096	0.6958	0.7061	0.6872	0.7222

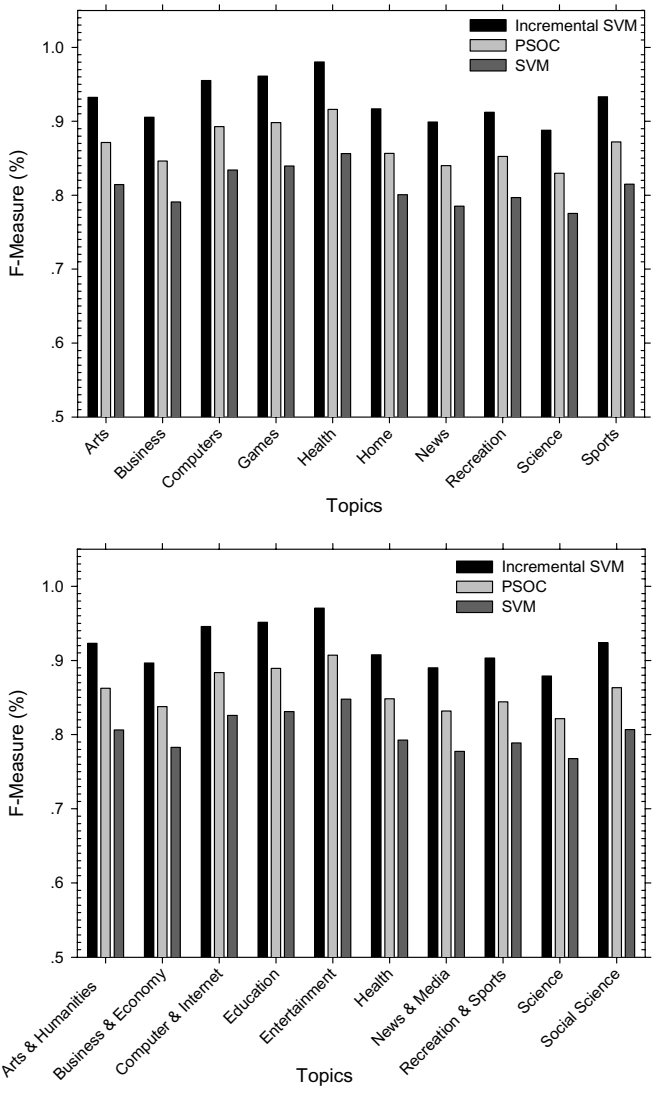

Fig. 7. *F-Measures* of our incremental SVM, PSOC and SVM method on two datasets.

eight times higher than that of standard SVM. In our clustering-based topical crawling method, the time complexity of clustering process is $O(n^2)$, and the time complexity of incremental SVM process is $O(sv^3)$, where sv is the number of examples which are used to train the hyperplane. Incremental SVM makes improvement on traditional SVM, which decreases the number of examples used in training process. So, the final time complexity is $O(sv^3)+ O(n^2)$. Because sv is smaller than the n, the final time complexity of our clustering-based topical crawling is not much higher than traditional topical crawling.

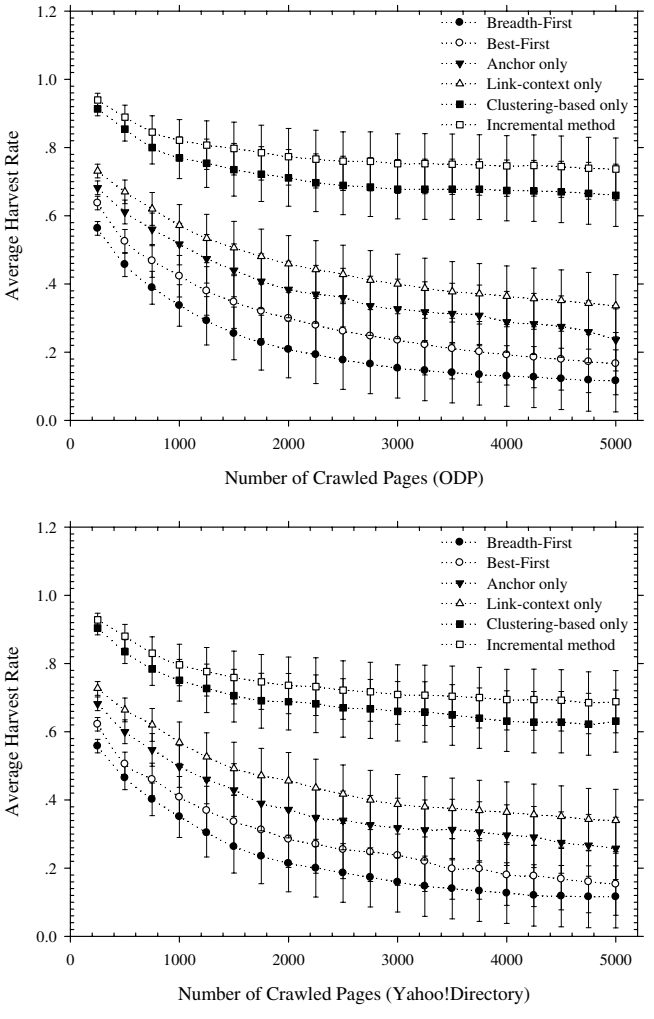

Fig. 8. Dynamic plot of average *Harvest rate* versus number of crawled pages on two datasets. Performance is averaged across ten topics on ODP and Yahoo! Directory. The error bars correspond to ± standard error.

For comparing our method with other topical Web crawling strategies, in this section, we built crawlers that used different techniques (Breadth-First, Best-First, Anchor text only, Link-context only, clustering-based only and our incremental method). Figures 8 and 9 show the average Harvest rate and average Target recall on two datasets for each crawling strategy, respectively. In the light of the results, Breadth-First fetched large numbers of irrelevant pages. And, its performance mainly depends on the localization of the relevant pages. Best-First, Anchor text only, Link-context only and clustering-based only predict the relevance of the potential URLs by referring to the whole context of the visited Web page, Anchor text,

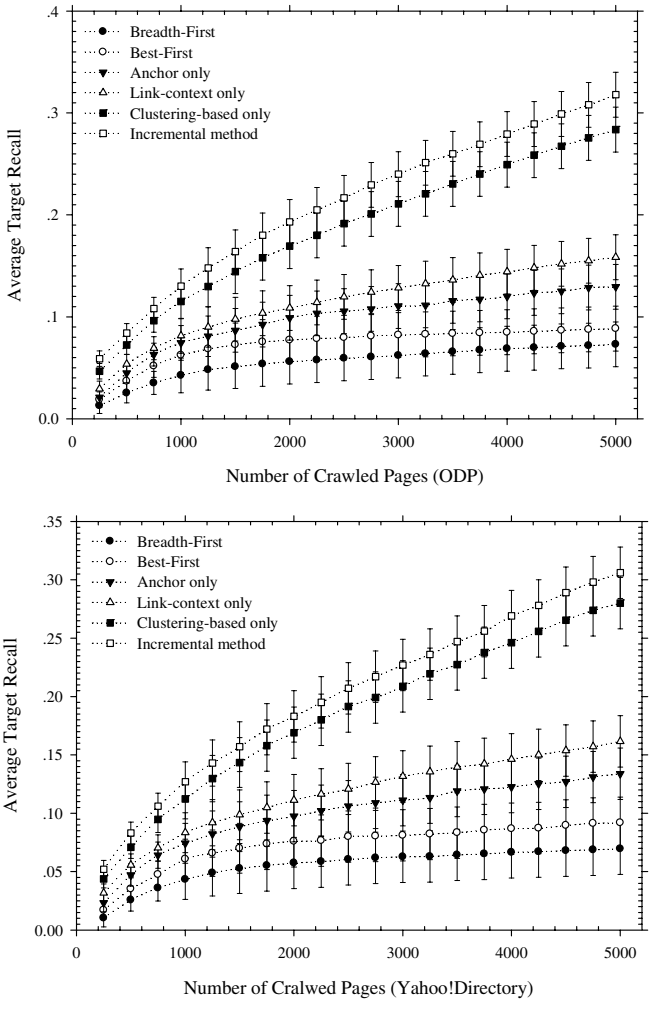

Fig. 9. Dynamic plot of average *Target Recall* versus number of crawled pages on two datasets. Performance is averaged across ten topics on ODP and Yahoo! Directory. The error bars correspond to ± standard error.

Link-context and the clusters of link-context, respectively. Based on all results, our incremental method has the best performance among other methods. Best-First has low accuracy when there is a lot of noise in the page or the page has multiple topics. Anchor text only crawler performs better than Best-First. Link-context crawler prioritizes the links picked up from the pages which improves the performance of crawler and makes it more efficient to crawl only using anchor text. Although clustering-based only method does not need training data, it can not solve the topic drift effectively. In summary, the clustering-based topical Web crawler guided by

incremental classifier shows significant performance improvement over the crawlers mentioned above.

7. Conclusions

In this paper, we present a novel incremental method for Web page classification using the combination of link-contexts, clustering and incremental SVM. Different from other methods about Web page classification, applying clustering to link-context could greatly improve the accuracy of classification because of the few features of each link-context. Moreover, incremental SVM dynamically classifies the documents, which heuristically selects the training samples instead of adding all the samples into the training process. In order to extract enough information covering sufficient features, grouping link-contexts from the Web is apparently meaningful and important. According to idea of agglomerative clustering, this paper proposed a more efficient method to group link-contexts from the Web, which built a binary model CV to quantifiably calculate the similarity between two clusters. The parameter lambda could adjust how closely between two clusters. Then we provide a series quantitative analysis for the effectiveness of our proposed method and the experimental results show that our method can achieve a relatively high precision and recall. Furthermore, incremental SVM could deal with the situation that various training samples do not share the same distribution, which could be used to applying a relatively small number of training data to obtain an accurate classifier.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grant Nos. 60903098, 60973040 and 71473035, MOE (Ministry of Education in China) Project of Humanities and Social Sciences (No. 14YJA870010), the Project of Jilin Provincial Industrial Technology Research and Development (JF2012c016-2), Jilin Provincial Science and Technology Key Project (20150204040GX).

References

1. R. R. Ade and P. R. Deshmukh, Methods for incremental learning: A survey, *International Journal of Data Mining & Knowledge Management Process* **3**(4) (2013) 119–125.
2. G. Attardi, A. Gullí and F. Sebastiani, Automatic web page categorization by link and context analysis, in *Proc. of First European Symp. Telematics, Hypermedia, and Artificial Intelligence*, 1999.
3. P. Bedi, A. Thukrai and H. Banati, A multi-threaded semantic focused crawler, *Journal of Computer Science and Technology* **27**(6) (2012) 1233–1242.
4. P. Bedi, A. Thukrai and H. Banati, Focused crawling of tagged web resources using ontology, *Computers & Electrical Engineering* **39**(2) (2013) 613–628.
5. C. Bouras and V. Tsogkas, A clustering technique for news articles using WordNet, *Knowledge-based Systems* **36** (2012) 115–128.

6. S. Brin and L. Page, The anatomy of a large-scale hypertextual web search engine, *Computer Networks and ISDN Systems* **30**(1–7) (1998) 107–117.
7. S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan and S. Rajagopalan, Automatic resource list compilation by analyzing hyperlink structure and associated text, in *Proceedings of the Seventh International World Wide Web*, 1998, pp. 65–74.
8. C. Cortes and V. Vapnik, Support vector networks, *Machine learning* **20**(3) (1995) 273–297.
9. W. B. Croft, D. Metzler and T. Strohman, *Search Engines: Informaton Retrieval in Practice* (Addison-Wesley, 2009).
10. R. Cota, A. Ferreira, C. Nascimento, M. Goncalves and A. Laender, An unsupervised heuristic-based hierarchical method for name disambiguation in bibliographic citations, *Journal of the American Society for Information Science and Technology* **61**(9) (2010) 1853–1870.
11. H. Dong and F. K. Hussain, SOF: A semi-supervised ontology-learning-based focused crawler, *Concurrency and Computation-Practice & Experience* **25**(12) (2013) 1755–1770.
12. R. Feldman and J. Sanger, *The Text Mining Handbook Advanced Approaches in Analyzing Unstructured Data* (Cambridge University Press, 2006).
13. B. Fung, K. Wang and M. Ester, Hierarchical document clustering using frequent itemsets, *SIAM International Conference on Data Mining*, 2003, pp. 59–70.
14. X. Y. Gao, L. P. B. Vuong and M. J. Zhang, Detecting data records in semi-structured Web sites based on text token clustering, *Integrated Computer-Aided Engineering* **15**(4) (2008) 297–311.
15. M. Hu and B. Liu, Mining and summarizing customer review, in *Proc. of ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2004, pp. 168–177.
16. A. Jain, M. Murty and P. Flyn, Data clustering: A review, *ACM Computing Surveys* **31** (2) (1999).
17. T. Jiang and A. H. Tan, Learning image-text associations, *IEEE Transactions on Knowledge and Data Engineering* **21**(2) (2009) 161–177.
18. S. Kim and E. Hovy, Determining the sentiment of opinions, in *Proc. of Intl. Conf. on Computational Linguistics*, 2004.
19. X. L. Li and B. Liu, Learning to classify text using positive and unlabeled data, in *Proc. of 18th Intl. Joint Conf. on Artificial Intelligence*, 2003, pp. 587–594.
20. Y. N. Li, Y. P. Wang and J. T. Du, E-FFC: An enhanced form-focused crawler for domain-specific deep web databases, *Journal of Intelligent Information* **40**(1) (2013) 159–184.
21. B. Liu, W. Hsu and Y. Ma, Integrating classification and associate rule mining, in *Proc. of Knowledge Discovery and Data Mining*, 1998, pp. 80–86.
22. H. Liu and E. Milios, Probabilistic models for focused web crawling, *Computational Intelligence* **28**(3) (2012) 289–328.
23. W. Liu and T. Wang, Online active multi-field learning for efficient email spam filtering, *Knowledge and Information Systems* **33**(1) (2012) 117–136.
24. J. Ma, W. Xu, Y. H. Sun, E. Turban, S. Y. Wang and O. Liu, An ontology-based text mining method to cluster proposals for research project selection, *IEEE Transactions on Systems Man and Cybernetics* **42**(3) (2012) 784–790.
25. A. McCallum and K. Nigam, A comparison of event models for Naïve Bayes text classification, in *Proc. of the AAAI-98 Workshop on Learning for Text Categorization*, 1998.
26. A. Mouton and P. F. Marteau, Exploiting routing information encoded into backlinks to improve topical crawling, in *International Conference of Soft Computing and Pattern Recognition*, 2009, pp. 659–664.

27. G. Pant, Deriving link-context from html tag tree, in *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2003, pp. 49–55.
28. G. Pant and P. Srinivasan, Link contexts in classifier-guided topical crawlers, *IEEE Transactions on Knowledge and Data Engineering* **18**(1) (2006) 107–122.
29. T. Peng, W. L. Zuo and F. L. He, SVM based adaptive learning method for text classification from positive and unlabeled documents, *Knowledge and Information Systems* **16**(3) (2008) 281–301.
30. J. A. Torkestani, An adaptive focused Web crawling algorithm based on learning automata, *Applied Intelligence* **37**(4) (2012) 586–601.
31. G. Pant, K. Tsioutsiouliklis, J. Johnson and C. Giles, Panorama: Extending digital libraries with topical crawlers, in *Proc. Fourth ACM/IEEE-CS Joint Conf. Digital Libraries*, 2004, pp. 142–150.
32. J. R. Quinlan, Bagging, boosting and C4.5, in *Proc. of National Conf. on Artificial Intelligence*, 1996, pp. 725–730.
33. M. Radovanovic, M. Ivanovic and Z. Budimac, Text categorization and sorting of web search results, *Computing and Imformatics* **28** (2010) 861–893.
34. J. X. Shen, An ontology-based adaptive topical crawling algorithm, in *4th International Conference on Wireless Communications, Networking and Mobile Computing*, (2008), pp. 12210–12213.
35. M. Steinbach, G. Karypis and V. Kumar, A comparison of document clustering techniques, KDD Workshop on Text Mining, 2000.
36. C. Tang, C. X. Ling, X. F. Zhou, N. J. Cercone and X. Li, Link-contexts for ranking, *Advanced Data Mining and Applications*, Lecture Notes in Artificial Intelligence, Vol. 5139, 2008, pp. 636–643.
37. Y. H. Tian, Q. Yang, T. J. Huang, C. X. Ling and W. Gao, Learning contextual dependency network models for link-based classification, *IEEE Transactions on Knowledge and Data Engineering* **18**(11) (2006) 1482–1496.
38. S. G. Totad, R. B. Geeta and P. V. G. D. P. Reddy, Batch incremental processing for FP-tree construction using FP-Growth algorithm, *Knowledge and Information Systems* **33**(2) (2012) 475–490.
39. A. Trivedi, P. Rai, H. Daume and S. L Duvall, Leveraging social bookmarks from partially tagged corpus for improved web page clustering, *ACM Transactions on Intelligent Systems and Technology* **3**(4) (2012) Article 67.
40. I. W. Tsang, J. T. Kwok and P. M. Cheung, Fast SVM training on very large data sets, *Journal of Machine Learning Research* **6** (2005) 363–392.
41. C. P. Wei, R. H. L. Chiang and C. C. Wu, Accommodating individual preferences in the categorization of documents: A personalized clustering approach, *Journal of Management Information Systems* **23**(2) (2006) 173–201.
42. H. Yu, J. Han and K. Chang, PEBL: Positive example based learning for web page classification using SVM, in *Proc. of the Knowledge Discovery and Data Mining*, 2002, pp. 239–248.
43. J. Zhang, Q. Wei and G. Q. Chen, An efficient incremental method for generating equivalence groups of search results in information retrieval and queries, *Knowledge-based Systems* **32** (2012) 91–100.