Available online at www.sciencedirect.com

ScienceDirect

Decision Support Systems 44 (2008) 482 – 494

Decision Support Systems

www.elsevier.com/locate/dss

A machine learning approach to web page filtering using content and structure analysis

Michael Chau [a,*], Hsinchun Chen [b]

[a] School of Business, The University of Hong Kong, Pokfulam, Hong Kong
[b] Department of Management Information Systems, The University of Arizona, Tucson, Arizona 85721, USA

Received 15 January 2006; received in revised form 10 February 2007; accepted 13 June 2007
Available online 22 June 2007

Abstract

As the Web continues to grow, it has become increasingly difficult to search for relevant information using traditional search engines. Topic-specific search engines provide an alternative way to support efficient information retrieval on the Web by providing more precise and customized searching in various domains. However, developers of topic-specific search engines need to address two issues: how to locate relevant documents (URLs) on the Web and how to filter out irrelevant documents from a set of documents collected from the Web. This paper reports our research in addressing the second issue. We propose a machine-learning-based approach that combines Web content analysis and Web structure analysis. We represent each Web page by a set of content-based and link-based features, which can be used as the input for various machine learning algorithms. The proposed approach was implemented using both a feedforward/backpropagation neural network and a support vector machine. Two experiments were designed and conducted to compare the proposed Web-feature approach with two existing Web page filtering methods — a keyword-based approach and a lexicon-based approach. The experimental results showed that the proposed approach in general performed better than the benchmark approaches, especially when the number of training documents was small. The proposed approaches can be applied in topic-specific search engine development and other Web applications such as Web content management.
© 2007 Elsevier B.V. All rights reserved.

Keywords: Web page classification; Link analysis; Machine learning; Web mining

1. Introduction

The most popular way to look for information on the Web is to use Web search engines such as Google (www.google.com) and AltaVista (www.altavista.com). Many users begin their Web activities by submitting a query to a search engine. However, as the size of the Web is still growing and the number of indexable pages on the Web has exceeded eight billion, it has become more difficult for search engines to keep an up-to-date and comprehensive search index. Users often find it difficult to search for useful and high-quality information on the Web using general-purpose search engines, especially when searching for specific information on a given topic.

Many vertical search engines, or topic-specific search engines, have been built to facilitate more efficient searching in various domains. These search engines alleviate the information overload problem to some extent by providing more precise results and more customized features [11]. For example, *LawCrawler*

* Corresponding author.
E-mail addresses: mchau@business.hku.hk (M. Chau), hchen@eller.arizona.edu (H. Chen).

0167-9236/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.dss.2007.06.002

(www.lawcrawler.com) allows users to search for legal information and provides links to lawyers and legal information and to relevant government Web sites. *BuildingOnline* (www.buildingonline.com) is a specialized search engine for the building industry, where users can search by manufacturers, architects, associations, contractors, etc. *BioView.com* (www.bioview.com) and *SciSeek* (www.sciseek.com) are two other examples that focus on scientific domains.

Although they provide a promising alternative for users, these vertical search engines are not easy to build. There are two major challenges to building vertical search engines: (1) How to locate relevant documents on the Web? (2) How to filter irrelevant documents from a collection? This study tries to address the second issue and to propose new approaches. The remainder of the paper is structured as follows. Section 2 reviews existing work on vertical search engine development, text classification, and Web content and structure analysis. In Section 3 we discuss some problems with existing Web page filtering approaches and pose our research questions. Section 4 describes in detail our proposed approach. Section 5 describes an experiment designed to evaluate our approach and presents experimental results. In Section 6, we conclude our paper with some discussion and suggestions for future research directions.

2. Research background

2.1. Building vertical search engines

A good vertical search engine should contain as many relevant, high-quality pages and as few irrelevant, low-quality pages as possible. Given the Web's large size and diversity of content, it is not easy to build a comprehensive and relevant collection for a vertical search engine. There are two main problems:

- The search engine needs to locate the URLs that point to relevant Web pages. To improve efficiency, it is necessary for the page collection system to predict which URL is the most likely to point to relevant material and thus should be fetched first.
- After the Web pages have been collected, the search engine system needs to determine the content and quality of the collection in order to avoid irrelevant or low-quality pages.

Search engines usually use spiders (also referred to as Web robots, crawlers, worms, or wanderers) as the software to retrieve pages from the Web by recursively following URL links in pages using standard HTTP protocols [9,13,15]. These spiders use different algorithms to control their search. To address the first problem mentioned above, the following methods have been used to locate Web pages relevant to a particular domain:

- The spiders can be restricted to staying in particular Web domains, because many Web domains have specialized contents [36,50]. For example, most Web pages within the domain www.toyota.com will be relevant to automobiles.
- Some spiders are restricted to collecting only pages at most a fixed number of links away from the starting URLs or starting domains [36,45]. Assuming that nearer pages have higher chances of being relevant, this method prevents spiders from going too "far away" from the starting domains.
- More sophisticated spiders use more advanced graph search algorithms that analyze Web pages and hyperlinks to decide what documents should be downloaded. Cho et al. [16] proposed a best-first search spider that used PageRank as the ranking heuristic; URLs with a higher PageRank scores will be visited first by the spider. The spider developed by McCallum et al. [38] used reinforcement learning to guide their spiders to retrieve research papers from university Web sites. Focused Crawler locates Web pages relevant to a pre-defined set of topics based on example pages provided by the user. In addition, it also analyzes the link structures among the Web pages collected [7]. Context Focused Crawler uses a Naïve Bayesian classifier to guide the search process [19]. A Hopfield Net spider based on spreading activation also has been proposed [8,10]. Page content scores and link analysis scores are combined to determine which URL should be visited next by the spider. The spider was compared with a breadth-first search spider and a best-first search spider using PageRank as the heuristics, and the evaluation results showed that the Hopfield Net spider performed better than the other two.

While these methods have different levels of performance in efficiency and effectiveness, in most cases the resulting collection is still noisy and needs further processing. Filtering programs are needed to eliminate irrelevant and low-quality pages from the collection to be used in a vertical search engine. The filtering techniques used can be classified into the following four categories:

- Domain experts manually determine the relevance of each Web page (e.g., Yahoo) [30].
- In the simplest automatic procedure, the relevance of a Web page can be determined by the occurrences of

particular keywords [16]. Web pages are considered relevant if they contain the specified keyword, and are considered irrelevant otherwise.
- TFIDF (term frequency * inverse document frequency) is calculated based on a lexicon created by domain-experts. Web pages are then compared with a set of relevant documents, and those with a similarity score above a certain threshold are considered relevant [2].
- Text classification techniques such as the Naive Bayesian classifier also have been applied to Web page filtering [5,38].

To our surprise, it appears that some vertical search engines do not perform filtering; they assume that most pages found in the starting domains (or at a specified depth) are relevant, e.g., NanoSpot (www.nanospot.com) [11].

2.2. Text classification

Text classification is the study of classifying textual documents into predefined categories. The topic has been extensively studied at SIGIR conferences and evaluated on standard testbeds. There are a number of major approaches. For example, the Naive Bayesian method has been widely used [28,33,38]. It uses the joint probabilities of words and categories to estimate the probability that a given document belongs to each category. Documents with a probability above a certain threshold are considered relevant to that category.

The *k*-nearest neighbor method is another popular approach to text classification. For a given document, the *k* neighbors that are most similar to a given document are first identified [22,37]. The categories of these neighbors are then used to decide the category of the given document. A threshold is also used for each category.

Neural network programs, designed to model the human neural system and learn patterns by modifying the weights among nodes based on learning examples, also have been applied to text classification. Feedforward/backpropagation neural network (FF/BP NN) is usually used [31,39,49]. Term frequencies or TFIDF of the terms are used as the input to the network. Based on learning examples, the network can be trained to predict the category of a document.

Another new technique used in text classification is called support vector machine (SVM), an approach that tries to find a hyperplane that best separates two classes [47,48]. Joachims first applied SVM to a text classification problem [23]. It has been shown that SVM achieved the best performance among different classifiers on the Reuters-21578 data set [20,44,51].

In addition to general text documents, classification of Web pages also has been studied. Web pages are often noisy, but they provide additional information about each document. For example, terms marked with different HTML tags (such as titles or headings) can be assigned a higher weight than regular text [34]. Terms from neighborhood Web pages also have been used in attempt to improve classification performance. However, it turns out to worsen performance because there are often too many neighbor terms and too many cross-linkages between different classes [5,52]. Use of other information about neighborhood Web pages has been proposed. Examples of such information include the predicted category of a page's neighbors [5,40], anchor text pointing to a page [21], or a page's outgoing links to all other documents [25]. It has been shown that using such additional information improves classification results.

2.3. Analysis of Web content and structure

There has been much research on different ways of representing and analyzing the content and structure of the Web. In general, they can be classified into two categories: content-based and link-based. The two approaches are discussed in the following.

2.3.1. Content-based approaches

The actual HTML content of a Web page provides much useful information about the page itself. For example, the body text of a Web page can be analyzed to determine whether the page is relevant to a target domain. Indexing techniques can be used to extract the key concepts that represent a page. Information extracted from a document using various techniques can be useful for text classification [42]. In addition, the relevance of a page can often be determined by looking at the title. Words and phrases that appear in the title or headings in the HTML structure are usually assigned a higher weight. Such weights can be calculated based on the TFIDF scores discussed earlier.

Domain knowledge also can be incorporated into the analysis to improve results. Domain knowledge refers to expert knowledge such as domain-specific lexicon or rules, often obtained from human experts. For example, words in Web pages can be checked against a list of domain-specific terms. A Web page containing words that are found in the list can be considered more relevant.

The URL address of a Web page often contains useful information about the page. For example, from the URL "http://ourworld.compuserve.com/homepages/LungCancer/", we can tell that it comes from the domain *compuserve.com*, and that it is likely to be related to the topic *Lung Cancer*. We also know that this page comes from a *.com* site, which may be considered less authoritative than pages from a *.gov* site. Some metrics also consider URLs with fewer slashes to be more useful than those with more slashes [1].

2.3.2. Web structure analysis

In recent years, Web link structure has been widely used to infer important information about pages. Intuitively, the author of a Web page A places a link to Web page B if he or she believes that B is relevant to A, or of good quality. Usually, the larger the number of in-links, the better a page is considered to be. The rationale is that a page referenced by more people is likely to be more important than a page that is seldom referenced. One can also obtain the anchor text that describes a link. Anchor text is the clickable text of an outgoing link in a Web page. Anchor text may provide a good description of the target page because it represents how other people who have linked to the page actually describe it.

In addition, it is reasonable to give a link from an authoritative source (such as Yahoo) a higher weight than a link from an unimportant personal homepage. Researchers have developed several methods to incorporate this into link analysis. Among these algorithms, PageRank and HITS are the two most widely used.

The PageRank algorithm is computed by weighting each in-link to a page proportionally to the quality of the page containing the in-link [3]. The quality of these referring pages also is determined by PageRank. Thus, the PageRank of a page p is calculated recursively as follows:

$$\mathrm{PageRank}(p) = (1-d) + d \times \sum_{\text{all } q \text{ linking to } p} \left(\frac{\mathrm{PageRank}(q)}{c(q)} \right)$$

where d is a damping factor between 0 and 1,$c(q)$ is the number of out-going links in q.

Intuitively, a Web page has a high PageRank score if the page is linked from many other pages, and the scores will be even higher if these referring pages are also good pages (pages that have high PageRank scores). It is also interesting to note that the PageRank algorithm follows a random walk model. The PageRank score of a page is proportional to the probability that a random surfer clicking on random links will arrive at that page. Applied in the commercial search engine Google, this score has been shown to be very effective for ranking search results [3]. Computation time, however, is a main problem in using PageRank. The PageRank score of each Web Page has to be calculated iteratively, making it computationally expensive.

Kleinberg [26] proposed a measure called the HITS (Hyperlink-Induced Topic Search) algorithm, which is similar to PageRank. In the HITS algorithm, authority pages are defined as high-quality pages related to a particular topic or search query. Hub pages are those that are not necessarily authorities themselves but that provide pointers to other authority pages. A page to which many others point should be a good authority, and a page that points to many others should be a good hub. Based on this intuition, an authority score and a hub score can be calculated for each Web page as follows:

$$\mathrm{AuthorityScore}(p) = \sum_{\text{all } q \text{ linking to } p} (\mathrm{HubScore}(q))$$

$$\mathrm{HubScore}(p) = \sum_{\text{all } r \text{ linking to } p} (\mathrm{AuthorityScore}(r))$$

A page with a high authority score is one pointed to by many hubs, and a page with a high hub score is one that points to many authorities. One example that applies the HITS algorithm is the Clever search engine [6], which has achieved a higher user evaluation than the manually compiled directory of Yahoo.

3. Research questions

Based on the review, we identified several problems with traditional approaches to Web page filtering. Firstly, a manual approach is very labor-intensive and time-consuming. Although such approach can achieve high quality, it is usually not feasible under limited resources. The keyword-based and the lexicon-based approaches can automate the process, but they both have shortcomings. A simple keyword-based approach cannot deal with problem of *polysemy*, i.e., words having more than one semantic meaning. For example, a Web page containing the word *cancer* might well be a medical report about treatment for lung cancer or the horoscope for people born under the zodiac sign of cancer. As a result, this approach can easily fail to eliminate irrelevant pages, thus lowering precision. On the other hand, as people often use different terms to refer to the same concept, e.g., *lung cancer* and *lung neoplasm*, this approach also can easily miss out relevant pages, thus lowering recall. The lexicon-based approach, which used the TFIDF-based similarity score between each document and a given domain lexicon, alleviates the problem by considering all terms in the documents. However, TFIDF calculation can be biased by the collection; if the collection is "noisy", irrelevant terms can possibly get a very high IDF score and thus a high TFIDF score. In addition, both the keyword-based and the lexicon-based approaches do not robustly resist text spamming, a popular practice in which Web page authors manipulate their page content to boost ranking.

Using text classifiers for Web page filtering seem to be the most promising approach, given their good performance in traditional text classification. However, one problem is that most classifiers were evaluated using at least 2/3 of the data for training in the hold-out sampling method. The problem becomes even worse in other evaluation methods such as k-fold cross-validation and leaving-one-out [27,43], in which $(100-k)\%$ of the data and all but one instance, respectively, were used for training. It is not feasible to obtain so large a set of training data in vertical search engine creation because usually only a small number of documents are tagged for classifying a large number of documents. It would be very expensive and time-consuming to tag a large number of documents manually. Also, most existing text classification techniques do not make use of domain knowledge, which is important in vertical search engines.

On the other hand, the hyperlink structure of the Web has been studied and applied with considerable success in Web structure mining research. For example, the PageRank and HITS algorithms have been widely used in Web search result ranking. These techniques can help identify Web pages with high quality and relevance [3,26]. In addition, such techniques can help identify Web pages that are in the same community and thus the same domain [29]. These issues are exactly those that need to be addressed in Web page filtering applications. However, the application of these Web structure mining techniques in Web page filtering has not been much investigated. It would be an interesting research question to study the effectiveness of using Web structure mining in Web page filtering.

In this study, the following research questions are investigated: (1) Can Web structure analysis techniques be used to help create a vertical search engine? (2) Can domain knowledge be used to enhance Web page filtering for a vertical search engine? (3) Can Web page classification be applied to a large collection (e.g., a million documents) with only a small number (a few hundred) of training examples?

4. A Web-feature approach

To address the problems with current approaches in Web page filtering, we propose an approach that incorporates Web content and structure analysis into Web filtering. Instead of representing each document as a bag of words, each Web page is represented by a limited number of content and link features. This reduces the dimensionality (the number of attributes used) of the classifier and thus the number of training examples needed. The characteristics of Web structure also can be incorporated into these "Web features."

Based on our review of existing literature, we determined that in general, the relevance and quality of a Web page can be reflected in the following aspects: (1) the content of the page itself, (2) the content of the page's neighbor documents, and (3) the page's link information. Several features are defined for each aspect.

4.1. Page content

The content of a page is probably the primary factor in determining whether a page is relevant to a given domain. As mentioned earlier, we represented the content of each page by a set of feature scores rather than a vector of words. We adopted an automatic approach that extracted all the terms from a page and compared them with a domain lexicon, similarly to the method used in Baujard et al. [2]. We looked at both the number of relevant terms that appeared in the page title and the TFIDF scores of the terms that appeared in the body of the page. Two feature scores were defined:

1. Title(p) = Number of terms in the title of page p found in the domain lexicon
2. TFIDF(p) = Sum of TFIDF of the terms in page p found in the domain lexicon.

4.2. Page content of neighbors

To incorporate the page content of the neighbors of a page, a score from each neighborhood document can be used instead of including all the terms from neighborhood documents, which appears to be more harmful than helpful [5,52]. In our approach, three types of neighbors were considered: incoming, outgoing, and sibling [5]. For any page p, incoming neighbors (parents) are the set of all pages that have a hyperlink pointing to p. Outgoing neighbors are pages whose hyperlinks are found in p. Sibling pages are those pages that are pointed by any of the parents of p. An example is shown in Fig. 1. In the example, pages a, b, and c are incoming neighbors of p; pages f and g are outgoing neighbors of p, and pages d and e are siblings of p.

Two content scores (title and TFIDF scores) of the neighborhood documents were determined similarly to those created in the previous aspect. Six features were used: the averages of the two scores for all incoming

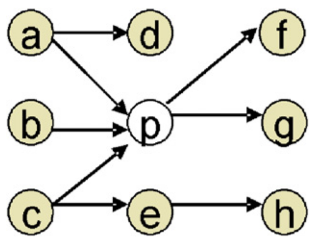

Fig. 1. Examples of incoming, outgoing, and sibling pages.

neighbors, the averages for all outgoing neighbors, and the averages for all siblings.

1. InTitle(*p*) = Average(number of terms in the title of page *q* found in the domain lexicon) for all incoming pages *q* of *p*
2. InTFIDF(*p*) = Average(sum of TFIDF of the terms in page *q* found in the domain lexicon) for all incoming pages *q* of *p*
3. OutTitle(*p*) = Average(number of terms in the title of page *r* found in the domain lexicon) for all outgoing pages *r* of *p*
4. OutTFIDF(*p*) = Average(sum of TFIDF of the terms in page *r* found in the domain lexicon) for all outgoing pages *r* of *p*
5. SiblingTitle(*p*) = Average(number of terms in the title of page *s* found in the domain lexicon) for all sibling pages *s* of *p*
6. SiblingTFIDF(*p*) = Average(sum of TFIDF of the terms in page *s* found in the domain lexicon) for all sibling pages *s* of *p*

4.3. Link analysis

Connectivity (link analysis) was used to represent the quality of a page. Link analysis, such as number of in-links, HITS and PageRank, have been useful in many Web applications such as search result ranking [3,6], but have not been used in text classification. To incorporate link analysis scores in our filtering approach, six scores, namely hub score, authority score, PageRank score, number of in-links, number of out-links, and number of relevant terms in the anchor texts, were used as features.

1. Hub(*p*) = Hub score of page *p* calculated by the HITS algorithm
2. Authority(*p*) = Authority score of page *p* calculated by the HITS algorithm
3. PageRank(*p*) = PageRank score of page *p*
4. Inlinks(*p*) = Number of incoming links pointing to *p*
5. Outlinks(*p*) = Number of outgoing links from *p*
6. Anchor(*p*) = Number of terms in the anchor texts describing page *p* found in the domain lexicon

4.4. FF/BP NN Text classifier

In total, 14 features have been identified and can be used as the input values to a classifier. We used a neural network (NN) [12,35] and a support vector machine (SVM) [47,48] as our classifiers. A feedforward/back-propagation neural network (FF/BP NN) had been adopted because of its robustness and wide usage in classification [31,39,49]. The algorithm used is summarized as follows:

4.4.1. Initializing the network

A neural network was first created with three layers, namely the input layer, the hidden layer, and the output layer. The input layer of the neural network consisted of a threshold unit and 14 nodes that corresponded to the 14 feature scores of each page. The output layer consisted of a single output node which determined the relevance of a page (whether or not a Web page should be included in the vertical search engine). The number of nodes in the hidden layer was set at 16 and the learning rate at 0.10. These parameters had been set based on some initial experimentation using a small subset of our data. The parameters that achieved the best performance were used throughout the experiment.

4.4.2. Training and tuning the network

The training documents were passed to the network for learning (the method of selecting the training set will be discussed in Section 5.5). The training documents were then further divided into two sets: 80% of the documents were used for training and 20% were used for tuning. The 14 features of each training document, as well as a binary score representing whether the document was relevant, were presented to the network. Each feature score was normalized to a value between 0 and 1 using the sigmoidal function. The network then updated the weights of its connection, based on the training documents. After all training documents had passed through the network once, the tuning documents were presented to the network and the mean square error (MSE) of the network was recorded. The whole process was repeated 3000 times (i.e., 3000 epochs) and the network with the lowest MSE was selected.

4.4.3. Testing

Each testing document was presented to the trained network, which tried to predict whether the document was relevant. The predictions were recorded and used to calculate the performance measures.

4.5. SVM Text classifier

In order to allow for a better comparison, a support vector machine also was used because of its outstanding performance in traditional text classification [51]. It performed classification based on the same set of feature scores. Our SVM classifier involved the following steps:

4.5.1. Model selection

A linear kernel function had been chosen for our SVM classifier because it is simple, learns quickly, and has been shown to achieve performance comparable to that of non-linear models like polynomial classifiers and radial basis functions in text classification applications [20,23].

4.5.2. Training

Each training example was represented as a vector of the 14 features selected and presented to the SVM to learn the feature weights.

4.5.3. Testing

Similarly to the neural network algorithm, the SVM tried to predict, on the basis of its classification model, whether each document in the testing set was relevant to the chosen domain. The results were recorded and used for evaluation.

5. Evaluation

5.1. Experiment testbed

In order to evaluate the proposed approach, two experiments that compared the proposed approaches with traditional approaches were conducted. The medical field was chosen as the domain for evaluation because many diverse users (including medical doctors, researchers, librarians and general public) seek important and high-quality information on health topics on the Web. It is also important for them to distinguish between Web pages of good and poor quality [14].

A Web page testbed and a medical lexicon created in previous research were used [8]. The Web page testbed was built by running a random-first search that started with 5 URLs in the medical domain and traversed the Web following random outgoing links. The random-first search was run until 1 million pages had been collected and indexed. The testbed represented a typical collection from simple Web spiders, and consisted of 1,040,388 valid, unique Web pages.

The medical lexicon was created based on the Metathesaurus, part of the Unified Medical Language System (UMLS) developed by the National Library of Medicine. About 600,000 medical phrases were extracted from the Metathesaurus. The lexicon was manually edited by a medical librarian and two filtering programs were developed and applied to refine the lexicon. The resulting lexicon has 300,442 unique terms.

To evaluate the proposed Web page classification approaches, 1000 documents were randomly chosen from the testbed. Each of these documents was processed automatically to calculate its feature scores and keyword vector. All other Web pages in the testbed were accessible for content, neighbor and link analysis during the process, meaning that such metrics as PageRank and HITS scores were calculated over the entire set of documents instead of just the 1000 chosen. Two graduate students with medical training were also recruited to classify each document manually as either “acceptable” or “not acceptable” for a medical search engine.

5.2. Benchmark approaches

The proposed neural network approach (NN-WEB) and support vector machine approach (SVM-WEB) were compared against two benchmark approaches: (1) a lexicon-based approach (LEXICON), and (2) a keyword-based support vector machine approach (SVM-WORD). The lexicon-based approach was chosen because it is fast and has been used in various information retrieval applications. The keyword-based SVM approach was selected because it has been shown to achieve the best performance in traditional text classification problems [51].

The lexicon-based approach was adopted from Baujard et al. [2]. TFIDF score was calculated for those terms found in the medical lexicon. The Jaccard’s similarity score between every document in the training set and the lexicon was calculated. Jaccard’s score is one of the most widely used similarity scores in information retrieval [46]. A threshold that divided these training documents into the two classes (relevant and irrelevant) with the highest accuracy was determined. This threshold was then used for testing.

The second benchmark approach was a keyword-based SVM approach adopted from Joachims’s [23]. In the preprocessing stage, each document was first tokenized into single words. Common functional terms that did not bear a significant semantic meaning (e.g., a, of, and is) then were filtered based on a pre-defined stop-word list. In order to reduce the number of unique words and the vector size, we also followed Joachims’s design by applying suffix-stripping (stemming) to the words, using Porter’s stemmer

[41]. After the pre-processing, each document was represented as a keyword vector, which was used as the input to the SVM for training and testing.

5.3. Implementation

All four approaches were implemented in order to test their performances. In the lexicon-based approach, the Arizona Noun Phraser (AZNP) was used to extract noun phrases from each document and these phrases were compared with the domain lexicon. The AZNP is a tool that extracts all valid noun phrases from a document, based on part-of-speech tagging and linguistic rules [53]. For the two approaches that rely on a support vector machine, the SVM-light package was used [24]. All other programs, including the feature score calculation and the neural network algorithm, were implemented in Java.

5.4. Hypotheses

Our experiment sought to compare the two Web-feature approaches with the two benchmark approaches. We posed the following hypotheses:

H1. The keyword-based SVM approach (SVM-WORD) will perform with higher effectiveness than the lexicon-based approach (LEXICON).

We reasoned that a keyword-based approach should be able to make better classification decisions by relying on more keyword information.

H2. The two proposed Web-feature approaches (NN-WEB and SVM-WEB) will perform with comparable effectiveness.

We hypothesized that the two proposed approaches should perform similarly because both neural network and support vector machine have been widely used in text classification applications and should achieve comparable performance.

H3. The two proposed Web-feature approaches (NN-WEB and SVM-WEB) will perform with higher effectiveness than the two benchmark approaches, i.e., the keyword-based SVM approach (SVM-WORD) and the lexicon-based approach (LEXICON).

This hypothesis tests the main thesis of this paper by verifying whether the proposed approaches perform better than the traditional approaches.

H4. The lexicon-based approach and the two Web-feature approaches will require significantly fewer training data to achieve a satisfactory performance than the keyword-based approach (SVM-WORD).

We suggest that the lexicon-based and the Web-feature approaches require fewer training data because they rely on only score(s) that should be similar across Web pages. Only a small number of training samples would be needed for the classifiers to learn the importance of the scores. On the other hand, the traditional keyword-based approach needs the occurrence of certain keywords in order to classify a document. When the number of training documents is small, it is likely that many words in the testing documents have not been seen before and hence provide no information for classification.

5.5. Experiment setup

Each of the four approaches was evaluated using cross-validation, a widely-used evaluation methodology for machine learning and text classification systems [27,43]. A 50-fold cross validation was adopted, in which the 1000 documents in the data set were divided into 50 equal portions, with 20 documents each. Testing was performed for 50 iterations, in each of which 49 portions of the data (980 documents) were used for training and the remaining portion (20 documents) was used for testing. The data were rotated during the process such that each portion was used for testing in exactly one iteration.

We measured the effectiveness of each system using precision, recall, *F*-measure, and accuracy. Precision measures the fraction of the documents correctly classified as relevant, while recall measures the fraction of relevant documents retrieved from the data set. *F*-measure is a single measure that tries to combine precision and recall. Accuracy measures simply the prediction correctness of the classifiers. These measures are commonly used in text classification evaluation and have been adopted as follows:

$$\text{precision} = \frac{\text{number of documents correctly classified as positive by the system}}{\text{number of all documents classified as positive by the system}}$$

$$\text{recall} = \frac{\text{number of documents correctly classified as positive by the system}}{\text{number of positive documents in the testing set}}$$

$$F\text{-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{accuracy} = \frac{\text{number of documents correctly classified by the system}}{\text{number of all documents in the testing set}}$$

There are two popular ways to calculate the averages across the data for these metrics, namely, macro-averaging and micro-averaging [4,18,23,32,51]. In macro-averaging, the performance metrics are calculated for each iteration, and the average of all iterations is obtained. In micro-averaging, the average is calculated across all the individual classification decisions made by a system. Both averages were calculated in our experiment. In addition to effectiveness, we also recorded the time used by each classifier in order to measure their efficiencies.

5.6. Experiment results and discussions

5.6.1. Effectiveness

The experiment results on accuracy, precision, recall, and F-measure are summarized in Table 1. Because F-measures represent a balance between precision and recall, we focus our discussion on accuracy and F-measure. The results demonstrated that the lexicon-based approach in general did not perform as well as the other approaches; it achieved the lowest accuracy and F-measure. NN-WEB achieved the highest accuracy and F-measure.

In order to study whether the differences among the different approaches were statistically significant, two statistical tests were adopted. The first was a micro sign-test that looks at all the classification decisions individually and uses a binomial distribution to determine whether the decisions made by any two approaches of interest are significantly different [17]. The number of observations n is defined to be the number of times that the two systems made different classification decisions. The second test was a macro t test that takes the performance of each iteration as an individual observation in order to determine whether the performances of two approaches are significantly different [51]. Our number of observations was 50, since there were 50 iterations of testing for each approach. The macro t test was applied to both accuracy and F-measure.

The p-values of the micro sign-tests are shown in Table 2. The results show that hypothesis H1 was supported as the keyword-based SVM approach performed better than the lexicon-based approach with a p-value less than 0.00001. H3 was partly supported; both the Web-feature approaches performed significantly better than the lexicon-based approach, and the Web-feature NN approach also performed significantly better than keyword-based SVM approach. H2 was not supported, since the Web-feature NN approach performed better than the Web-feature SVM approach.

Table 2
Micro sign-test results

vs.	SVM-WORD	NN-WEB	SVM-WEB
LEXICON	<0.00001 [b]	<0.00001 [b]	<0.00001 [b]
SVM-WORD		0.0972 [a]	0.3044
NN-WEB			0.0095 [b]

[a] The difference is statistically significant at the 10% level.
[b] The difference is statistically significant at the 1% level.

The p-values of the macro t-tests on accuracy and F-measure are shown in Tables 3 and 4, respectively. The results obtained were similar to those of the micro sign-test. In general, H1 was supported as the keyword-based approach performed significantly better than lexicon-based approach. H3 also was partly supported, since the Web-feature NN approach performed better than both benchmark approaches, but the Web-feature SVM approach only performed better than the lexicon-based approach but no better than the keyword-based approach. H2 was not supported; the Web-feature NN approach performed better than the Web-feature SVM approach. One possible reason is that because of the limitation in resources and time, we were only able to use the linear model in the SVM. It is possible that the performance of both SVM approaches might improve if a non-linear model could be adopted, which could be more time-consuming, however.

5.6.2. Efficiency

We also recorded the time needed for each system to perform the 50-fold cross validation (including both training and testing time). The data are shown in Table 5. As can be seen, the keyword-based SVM approach required the longest time. The reason is that

Table 1
Experiment results

	Accuracy (%)	Precision (macro/micro) (%)	Recall (macro/micro) (%)	F-measure (macro/micro)
LEXICON	80.80	63.40/63.95	60.52/62.50	0.6005/0.6322
SVM-WORD	87.80	87.97/94.94	55.08/56.82	0.6646/0.7109
NN-WEB	89.40	81.38/82.38	76.19/76.14	0.7614/0.7913
SVM-WEB	87.30	85.35/86.24	61.99/61.74	0.7049/0.7196

Table 3
Macro *t*-test results on accuracy

vs.	SVM-WORD	NN-WEB	SVM-WEB
LEXICON	<0.00001 [b]	<0.00001 [b]	<0.0001 [b]
SVM-WORD		0.1627	0.6091
NN-WEB			0.0216 [a]

[a] The difference is statistically significant at the 5% level.
[b] The difference is statistically significant at the 1% level.

Table 5
Time

	Time (min)
LEXICON	7.45
SVM-WORD	382.55
NN-WEB	103.45
SVM-WEB	37.60

each document was represented as a large vector of keywords, which created a high dimensionality for the classifier. In our experiment, there were more than 6000 unique words after stop-word removal and stemming. The classifier had to learn the relationships between all these attributes and the class attribute, thus requiring more time. The lexicon-based approach used the least time, as it needed only to calculate the TFIDF and similarity scores for each document and determine the threshold, both of which did not require complex processing. Comparing the two Web-feature approaches, the NN classifier required a longer time than the SVM classifier because the neural network had to be trained in multiple epochs, i.e., in each iteration the training data set had to be presented to the network thousands of times in order to improve the network's performance.

5.6.3. *Effect of the number of training examples*

In order to analyze the effect of the number of training examples on the performance, we ran the experiments on the systems while varying the number of training data used. We started with 20 documents in the first run, and increased the number of training documents by 20 in each subsequent run. There were thus 49 runs in total (from 20 to 980 training documents). In each run, a 50-fold cross validation similar to the one described above was used, and 20 documents were used for testing with rotation. The macro-averaged *F*-measure for each iteration was recorded and the results are shown in Fig. 1.

From the graph shown in Fig. 2, we can see that the performances of the lexicon-based approach and the two Web-feature approaches became relatively stable after approximately 300, 140, and 260 training documents were used respectively. For the keyword-based approach, however, performance was unstable until about 700 training documents had been used. This supported our hypothesis H4 that fewer documents were needed for the lexicon-based approach or the Web-feature approaches to achieve a satisfactory performance. As discussed earlier, this finding is especially important for building vertical search engines as a large number of training documents often is not unavailable.

Table 4
Macro *t*-test results on *F*-measure

vs.	SVM-WORD	NN-WEB	SVM-WEB
LEXICON	0.0827 [a]	<0.00001 [b]	0.0041 [b]
SVM-WORD		0.0024 [b]	0.2446
NN-WEB			0.0033 [b]

[a] The difference is statistically significant at the 10% level.
[b] The difference is statistically significant at the 1% level.

6. Conclusion and future directions

In this paper, we have described a Web-feature approach to Web page classification that combines Web content analysis and Web structure analysis. We compared our approaches with traditional text classification methods and found the experimental results to be encouraging. We believe that the proposed approaches are useful for various Web applications, especially for vertical search engine development.

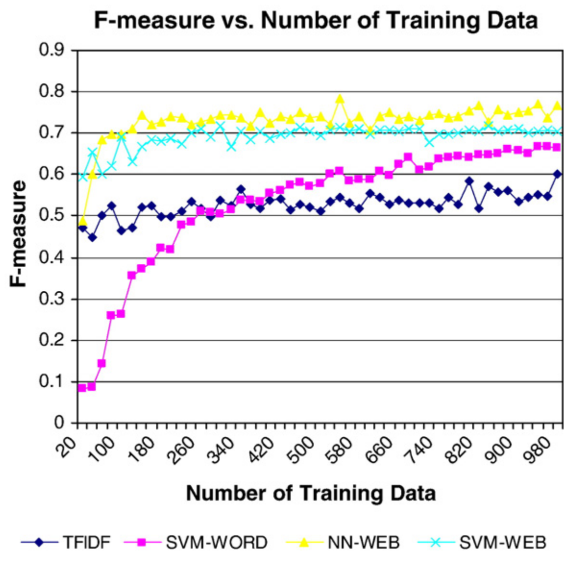

Fig. 2. *F*-measure vs. the number of training data.

While the Web-feature approaches are promising, it is interesting to examine which of the 14 features used are more important than the others in determining the relevance of a page. We plan to apply factor analysis techniques to the data set to investigate the features in detail. Another direction of our future work will be to study whether a combined keyword-based and Web-feature approach will perform better than using either the keywords or the Web features alone. We believe that a combined approach may potentially acquire the strengths of both approaches and perform better by allowing the classifier to rely on the feature scores when the number of training documents is small but to rely more on unique keyword attributes in the vector when the number of training documents reaches a certain level. Finally, we are also investigating how the proposed classification method can be used in other applications, such as knowledge management and Web content management.

Acknowledgements

This project has been supported in part by the following grants:

- NSF Digital Library Initiative-2 (PI: H. Chen), “High-performance Digital Library Systems: From Information Retrieval to Knowledge Management,” IIS-9817473, April 1999–March 2002;
- NIH/NLM Grant (PI: H. Chen), “UMLS Enhanced Dynamic Agents to Manage Medical Knowledge,” 1 R01 LM06919-1A1, February 2001–January 2004;
- HKU Seed Funding for Basic Research (PI: M. Chau), “Using Content and Link Analysis in Developing Domain-specific Web Search Engines: A Machine Learning Approach,” February 2004–July 2005.

We would like to thank the National Library of Medicine for making UMLS freely available to researchers, and the medical experts who participated in the user studies.

References

[1] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, Searching the web, ACM Transactions on Internet Technology 1 (1) (2001) 2–43.

[2] O. Baujard, V. Baujard, S. Aurel, C. Boyer, R.D. Appel, Trends in medical information retrieval on the Internet, Computers in Biology and Medicine 28 (1998) 589–601.

[3] S. Brin, L. Page, The anatomy of a large-scale hypertextual web search engine, Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, Apr 1998.

[4] K.M.A. Chai, H.L. Chieu, H.T. Ng, Bayesian online classifiers for text classification and filtering, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, Aug 2002, pp. 97–104.

[5] S. Chakrabarti, B. Dom, P. Indyk, Enhanced hypertext categorization using hyperlink, Proceedings of ACM SIGMOD International Conference on Management of Data, Seattle, Washington, USA, Jun 1998.

[6] S. Chakrabarti, B.E. Dom, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, J. Kleinberg, Mining the web’s link structure, IEEE Computer 32 (8) (1999) 60–67.

[7] S. Chakrabarti, M. van den Berg, B. Dom, Focused crawling: a new approach to topic-specific web resource discovery, Proceedings of the 8th International World Wide Web Conference, Toronto, Canada, May 1999.

[8] M. Chau, H. Chen, Comparison of three vertical search spiders, IEEE Computer 36 (5) (2003a) 56–62.

[9] M. Chau, H. Chen, Personalized and focused Web spiders, in: N. Zhong, J. Liu, Y. Yao (Eds.), Web intelligence, Springer–Verlag, 2003b, pp. 197–217.

[10] M. Chau, H. Chen, Incorporating Web analysis into neural networks: an example in hopfield net searching, IEEE Transactions on Systems, Man, and Cybernetics (Part C) 37 (3) (2007) 352–358 (May).

[11] M. Chau, Z. Huang, J. Qin, Y. Zhou, H. Chen, Building a scientific knowledge web portal: the nanoport experience, Decision Support Systems 42 (2) (2006) 1216–1238.

[12] H. Chen, Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms, Journal of the American Society for Information Science 46 (3) (1995) 194–216.

[13] H. Chen, Y. Chung, M. Ramsey, C.C. Yang, An intelligent personal spider (Agent) for dynamic Internet/Intranet searching, Decision Support Systems 23 (1998) 41–58.

[14] H. Chen, A. Lally, B. Zhu, M. Chau, HelpfulMed: intelligent searching for medical information over the internet, Journal of the American Society for Information Science and Technology 54 (7) (2003) 683–694.

[15] F.C. Cheong, Internet Agents: Spiders, Wanderers, Brokers, and Bots, New Riders Publishing, Indianapolis, Indiana, USA, 1996.

[16] J. Cho, H. Garcia-Molina, L. Page, Efficient crawling through URL ordering, Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, Apr 1998.

[17] W.W. Cohen, Text categorization and relational learning, Proceedings of the 12th International Conference on Machine Learning (ICML’95), Morgan Kaufmann, 1995.

[18] W.W. Cohen, Y. Singer, Context-sensitive learning methods for text categorization, ACM Transactions on Information Systems 17 (2) (1999) 141–173.

[19] M. Diligenti, F. Coetzee, S. Lawrence, C.L. Giles, M. Gori, Focused crawling using context graphs, Proceedings of the 26th International Conference on Very Large Databases, VLDB 2000, Cairo, Egypt, 2000, pp. 527–534.

[20] S.T. Dumais, J. Platt, D. Heckerman, M. Sahami, Inductive learning algorithms and representations for text categorization, Proceedings of ACM Conference on Information and Knowledge Management, Bethesda, Maryland, Nov. 1998, pp. 148–155.

[21] J. Furnkranz, Exploiting structural information for text categorization on the WWW, Proceedings of the 3rd Symposium on Intelligent Data Analysis (IDA’99), Amsterdam, Netherlands, 1999, pp. 487–497.

[22] M. Iwayama, T. Tokunaga, Cluster-based text categorization: a comparison of category search strategies, Proceedings of the

18th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'95), 1995, pp. 273–281.
[23] T. Joachims, Text categorization with support vector machines: learning with many relevant features, Proceedings of the European Conference on Machine Learning, Berlin, 1998, pp. 137–142.
[24] T. Joachims, Making large-Scale SVM Learning Practical, in: B. Schölkopf, C. Burges, A. Smola (Eds.), Advances in Kernel Methods— Support Vector Learning, MIT-Press, 1999.
[25] T. Joachims, N. Chistianini, J. Shawe-Taylor, Composite kernels for hypertext categorization, Proceedings of the 18th International Conference on Machine Learning (ICML'01), 2001.
[26] J. Kleinberg, Authoritative sources in a hyperlinked environment, Proceedings of the ACM-SIAM Symposium on Discrete Algorithms, 1998.
[27] R. Kohavi, A study of cross-validation and bootstrap for accuracy estimation and model selection, Proceedings of the 14th International Joint Conference on Artificial Intelligence, San Francisco, CA, Morgan Kaufmann, 1995, pp. 1137–1143.
[28] D. Koller, M. Sahami, Hierarchically classifying documents using very few words, Proceedings of the 14th International Conference on Machine Learning (ICML'97), 1997, pp. 170–178.
[29] R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, Trawling the Web for emerging cyber-communities, Proceedings of the Eighth WWW Conference, Toronto, Canada, May 1999.
[30] Y. Labrou, T. Finin, Yahoo! as an ontology: using yahoo! categories to describe documents, Proceedings of the Eighth International Conference on Information and Knowledge Management, Kansas City, Missouri, United States, 1999, pp. 180–187.
[31] S.L.Y. Lam, D.L. Lee, Feature reduction for neural network based text categorization, Proceedings of the International Conference on Database Systems for Advanced Applications (DASFAA '99), Hsinchu, Taiwan, Apr 1999, 1999.
[32] W. Lam, M. Ruiz, P. Srinivasan, Automatic text categorization and its application to text retrieval, IEEE Transactions on Knowledge and Data Engineering 11 (6) (1999) 865–879.
[33] D.D. Lewis, M. Ringuette, Comparison of two learning algorithms for text categorization, Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94), 1994.
[34] S.H. Lin, M.C. Chen, J.M. Ho, Y.M. Huang, ACIRD: intelligent Internet document organization and retrieval, IEEE Transactions on Knowledge and Data Engineering 14 (3) (2002) 599–614.
[35] R.P. Lippmann, An introduction to computing with neural networks, IEEE Acoustics Speech and Signal Processing Magazine 4 (2) (1987) 4–22.
[36] U. Manber, M. Smith, B. Gopal, WebGlimpse: combining browsing and searching, Proceedings of the USENIX 1997 Annual Technical Conference, Anaheim, California, Jan 1997.
[37] B. Masand, G. Linoff, D. Waltz, Classifying news stories using memory based reasoning, Proceeedings of the 15th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'92), 1992, pp. 59–64.
[38] A. McCallum, K. Nigam, J. Rennie, K. Seymore, A machine learning approach to building domain-specific search engines, Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99), 1999, pp. 662–667.
[39] H.T. Ng, W.B. Goh, K.L. Low, Feature selection, perceptron learning, and a usability case study for text categorization, Proceedings of the 20th Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'97), 1997, pp. 67–73.
[40] H.J. Oh, S.H. Myaeng, M.H. Lee, A practical hypertext categorization method using links and incrementally available class information, Proceedings of the 23rd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'00), 2000, pp. 264–271.
[41] M.F. Porter, An algorithm for suffix stripping, Program 14 (3) (1980) 130–137.
[42] E. Riloff, W. Lehnert, Information extraction as a basis for high-precision text classification, ACM Transactions on Information Systems 12 (3) (1994) 296–333.
[43] M. Stone, Cross-validation choices and assessment of statistical predictions, Journal of the Royal Statistical Society 36 (1974) 111–147.
[44] A. Sun, E.-P. Lim, W.-K. Ng, Performance measurement framework for hierarchical text classification, Journal of the American Society for Information Science and Technology 54 (11) (2003) 1014–1028.
[45] R.G. Sumner, K. Yang, B.J. Dempsey, An interactive www search engine for user-defined collections, Proceedings of the 3rd ACM Conference on Digital Libraries, Pittsburgh, Pennsylvania, USA, Jun 1998, pp. 307–308.
[46] C.J. van Rijsbergen, Information Retrieval, Second Edition, Butterworths, London, 1979.
[47] V. Vapnik, The Nature of Statistical Learning Theory, Springer, 1995.
[48] V. Vapnik, Statistical Learning Theory, Wiley, Chichester, GB, 1998.
[49] E. Wiener, J.O. Pedersen, A.S. Weigend, A neural network approach to topic spotting, Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95), 1995.
[50] I.H. Witten, D. Bainbridge, S.J. Boddie, Greenstone: open-source DL software, Communications of the ACM 44 (5) (2001) 47.
[51] Y. Yang, X. Liu, A re-examination of text categorization methods, Proceedings of the 22nd Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'99), 1999, pp. 42–49.
[52] Y. Yang, S. Slattery, R. Ghani, A study of approaches to hypertext categorization, Journal of Intelligent Information Systems 18 (2) (March 2002).
[53] K. Tolle, H. Chen, Comparing noun phrasing techniques for use with medical digital library tools, Journal of the American Society for Information Science 51 (4) (2000) 352–370.

Michael Chau is an Assistant Professor and the BBA(IS)/BEng(CS) Coordinator in the School of Business at the University of Hong Kong. He received his PhD degree in management information systems from the University of Arizona and a bachelor degree in computer science and information systems from the University of Hong Kong. His current research interests include information retrieval, Web mining, data mining, knowledge management, and security informatics. He has published more than 60 research articles in leading journals and conferences, including *IEEE Computer*, *Journal of the America Society for Information Science and Technology*, *Decision Support Systems*, *ACM Transactions on Information Systems*, and *Communications of the ACM*. More information can be found at http://www.business.hku.hk/~mchau/.

Hsinchun Chen is a McClelland Professor of Management Information Systems at the University of Arizona and Andersen Consulting Professor of the Year (1999). He received the B.S. degree from the National Chiao-Tung University in Taiwan, the MBA degree from SUNY Buffalo, and the PhD degree in Information Systems from the New York University. Dr. Chen is a Fellow of IEEE and AAAS. He received the IEEE Computer Society 2006 Technical Achievement Award. He is author/editor of 13 books, 17 book chapters, and more than 130 SCI journal articles covering intelligence analysis, biomedical informatics, data/text/web mining, digital library, knowledge management, and Web computing. Dr. Chen was ranked #8 in publication productivity in Information Systems (CAIS 2005) and #1 in Digital Library research (IP&M 2005) in two recent bibliometric studies. He serves on ten editorial boards including: *ACM Transactions on Information Systems, IEEE Transactions on Systems, Man, and Cybernetics, Journal of the American Society for Information Science and Technology,* and *Decision Support Systems*. Dr. Chen has served as a Scientific Counselor/Advisor of the National Library of Medicine (USA), Academia Sinica (Taiwan), and National Library of China (China). He has been an advisor for major NSF, DOJ, NLM, DOD, DHS, and other international research programs in digital library, digital government, medical informatics, and national security research. Dr. Chen is founding director of Artificial Intelligence Lab and Hoffman E-Commerce Lab. He is conference co-chair of ACM/IEEE Joint Conference on Digital Libraries (JCDL) 2004 and has served as the conference/program co-chair for the past eight International Conferences of Asian Digital Libraries (ICADL), the premiere digital library meeting in Asia that he helped develop. Dr. Chen is also (founding) conference co-chair of the IEEE International Conferences on Intelligence and Security Informatics (ISI) 2003–2007. Dr. Chen has also received numerous awards in information technology and knowledge management education and research including: AT&T Foundation Award, SAP Award, the Andersen Consulting Professor of the Year Award, the University of Arizona Technology Innovation Award, and the National Chaio-Tung University Distinguished Alumnus Award. Further information can be found at http://ai.arizona.edu/hchen/.