Improved Automatic Web-page Classification by Neighbor Text Percolation*

Shyh-Ming Tai, Cheng-Zen Yang and Ing-Xian Chen
Department of Computer Engineering and Science
Yuan Ze University
Chungli, Taiwan, R.O.C.
{dais,czyang,sean}@syslab.cse.yzu.edu.tw

Abstract

Automatic Web document classification is a very important issue to help users effectively find highly relevant information. However, traditional content-analysis schemes suffer a problem that documents that lack rich descriptive content cannot be effectively classified. In this paper, we present an improved classification scheme by incorporating a link-analysis-based method to enrich the textual content with neighbor documents. To study the improvement, we apply our scheme to a typical content-based scheme proposed by Jenkins and Inman (J&I) and have conducted preliminary experiments. The experimental results show that the classification accuracy is elevated from 76.3% to 85%. In addition, only one page cannot be classified in our approach, contrast to the result of four pages in the J&I approach. From the working experience, we believe that the neighbor text percolation scheme can be applied to other content-based approaches.

Keywords: Web page classification, content-based analysis, link-based analysis, neighbor text percolation, information retrieval.

1. Introduction

As the amount of Web pages emerges from 1990s, Web document classification is a very important issue to help users effectively find highly relevant information. Traditional Web document classification is manually performed for high quality. However, manual classification is tedious and cannot effectively classify the explosively growing amount of web pages. In recent years, algorithms have been proposed to automatically classify the large amount of Web pages [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. They are fallen into three categories: the content-based analysis approach [1, 6, 7, 10, 12], the link-based analysis approach [3, 5, 7] and the hybrid approach [2, 4, 9, 11]. The algorithms based on content analysis are primarily to use some IR techniques such as k-NN classifiers [10] and support vector machine (SVM) classifiers [12] to filter out the *feature* words from a set of training documents, and then compute the similarity

* This research was supported by the National Science Council under grants No. NSC 91-2213-E-155-038.

between the training documents and the test documents. However, researches have pointed out an underlying problem in such a approach that some pages cannot be correctly classified if they have very few feature words or little descriptive text [7, 8].

On the other hand, the algorithms based on link analysis utilize the hyperlink information embedded in Web pages for classification. This approach is based on an important assumption that if there is a hyperlink pointing to page *v* from page *u*, the author of page *u* does believe that *v* is relevant to *u*. However, though page connectivity provides valuable information to classify relevant pages, there are still several potential problems. First, the linkage information cannot directly reflect the semantic meaning. If a page is highly relevant but not linked by many other pages, it is likely excluded from the category to which it should belong [2]. Furthermore, the topic drift problem may arise if some topic-irrelevant but popular pages are pointed to [2, 13]. For example, the homepage of Apache Web server (www.apache.org) may be pointed to by several irrelevant but homogeneous homepages, say educational pages, just because these sites are constructed with Apache Web servers. If the neighborhood pages of the Apache homepage are included in the link analysis of these educational homepages, the Apache homepage may get a high authority score and is thus regarded as a relevant page to the educational homepages. In addition, as reported by Ng et al. [14, 15], link-based algorithms may be unstable if some documents are excluded from the corpus. In the examples shown in their papers, a page ranked in top ten in HITS [7] may get behind hundred of pages after some pages are excluded.

Hybrid algorithms are proposed to improve the classification precision by combining link-based analysis and content-based analysis [2, 4, 9, 11]. They can be further classified into two sub-categories. One is to reinforce the link-based analysis with content-based analysis, and the other is vice versa. For example, the algorithms proposed by Bharat and Henzinger [2] are based on Kleinberg's HITS algorithm with reinforcement of content-based pruning, and can achieve at least 45% improvement of precision. On the contrary, the iterative relaxation algorithm is an example in which the term-based classifier is reinforced by computing the distances between the test page and the pages followed from co-citation links [4]. It can reduce the error rate from original 68% of the text classifier to 21%.

However, the link-based algorithms with reinforcement of content-based analysis may suffer two problems. The first problem arises when hubs are mixed and content-based pruning may discard some valuable links [11]. The second problem arises when the link-based analysis is unstable [14, 15].

On the other hand, although the content-based algorithms with reinforcement of link-based analysis inherit good performance of precision from content-based analysis, it is not trivial to use neighbor text decided by the link analysis to facilitate classification. As reported in [4], simply adding neighbor text into content analysis can even increase the classification error rate due to too many noisy feature words. Though the co-citation information highly facilitates the reduction of classification errors, it hurts the coverage [4]. Therefore, these research results suggest that neighbor text should be carefully selected.

In this paper, we present an enhanced approach to the content-based analysis by percolating highly relevant feature words from neighbor pages. The percolation is performed with a PageRank-like [16] filtering function to control the quality of the

included neighbor text. To evaluate our percolation approach, we implement it on the basis of the algorithm proposed by Jenkins and Inman (J&I) [8]. The reason of using J&I's algorithm is that it is a typical content-based algorithm and can be easily implemented. Indeed, our approach should be easily applied to other content-based algorithms.

We have conducted similar experiments as described in [8] for comparison. A classification hierarchy for architecture from Yahoo! is chosen. The experiment results show that the classification accuracy is elevated from 76.3% to 85% in our algorithm, and the improvement is about 11.5% to the original J&I approach. In addition, only one page cannot be classified in our approach, but the original J&I approach cannot classify four pages.

The rest of this paper is organized as follows. Section 2 briefly reviews the original J&I approach. In Section 3, the percolation enhancement is presented and processing flow is described. Section 4 reports on experimental results. Section 5 concludes the paper with some comments on future work.

2. Overview of the J&I Approach

The J&I approach utilizes vocabulary vectors generated from training documents to form the classification hierarchy. In their approach, only words in <TITLE>, <H1>, and <H2> tags are considered in the content analysis. These words are used to comprise vectors for classification. There are two kinds of vectors: the document vocabulary vectors (DVV) and the class vocabulary vectors (CVV). Each DVV consists of a series of DocEntry objects to record the appearance frequency and the tag attribute of each considered word in the document. Figure 1 depicts the data structure of a DocEntry object.

Word (string)	Frequency (int)	Title (boolean)	H1 (boolean)	H2 (boolean)

Figure 1: The DocEntry data structure for

Each time a new training document is parsed, the correspondent DocEntry objects are decided. Then the resulting DVV is used to update the correspondent VocabularyEntry in the CVV. Each CVV represents a class of feature vocabulary, and will be used to generate a classification hierarchy. Figure 2 depicts the data structure of a VocabularyEntry object in a CVV. When all training documents are processed, the procedure of creating the final vocabulary for a class then starts. The words that occurred in more than 10% of the training documents are extracted and scored. The 50% words of the top scored are used to construct the hierarchy. Scores are calculated as follows:

Word (string)	Overall Frequency (integer)	Document Frequency (integer)	<TITLE> (boolean)	<H1> (boolean)	<H2> (boolean)	Score (integer)

Figure 2: The VocabularyEntry data structure in the class vocabulary.

$$score = f(d + t + H + h), \quad (1)$$

where f = overall frequency, d = document frequency, t = title frequency, H = <H1> frequency, and h = <H2> frequency.

These automatically generated vocabularies are then organized into a classification hierarchy. The hierarchy is generated by using a numerical prefix on the class name. The automatically generated hierarchy is then used to classify further test documents. For more details about hierarchy creation and automatic classification, readers can refer [8].

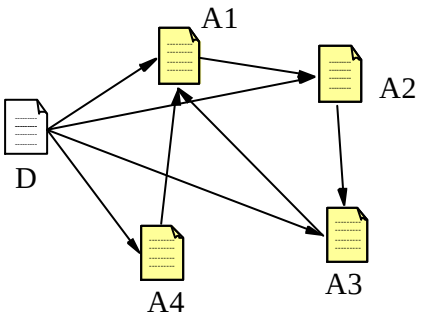
3. Neighbor Text Percolation

However, in J&I's experiment, 4 out of total 28 URLs cannot be classified [8]. The reason they concluded is that these documents contain very little descriptive text. For this sparse content problem, a direct solution is to gather auxiliary

information from the neighborhood of these unclassified pages, and then use it to help the classification. However, directly using neighbor text may suffer the "noisy word" problem due to the dissimilar term distribution of the neighbor documents [4]. These noisy words need to be filtered out to improve the classification precision and the error rate. To effectively distill relevant features from the neighbor text, a PageRank-based function is used in our approach to first compute the rank *PR(i)* of each neighbor document, defined recursively according to the equation [16]

$$PR(i) = (1-d) + d \sum_{j \to i} [PR(j)/N(j)] \quad (2)$$

The parameter *d* is the damping factor and *N(j)* denotes the number of links going out of page *j*. Here we adopt a PageRank-based function rather than the HITS algorithm because the HITS algorithm suffers the problem that it may rank a low quality page with a high authority score [17]. Furthermore, it also suffers the nepotism problem [11]. However, the PageRank algorithm relieves these problems by weighting each link to the quality of the page containing the link [17]. Figure 3 depicts a ranking computation example. In



PR	PR(A1)	PR(A2)	PR(A3)	PR(A4)
Initial	0.15	0.15	0.15	0.15
Iteration 1	0.405	0.2775	0.2775	0.15
Iteration 2	0.5134	0.4943	0.3858	0.15
…	…	…	…	…

Figure 3: A PageRank-like algorithm is used to compute the rank of each neighbor document.

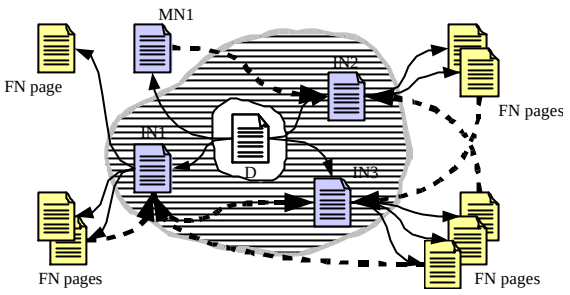


Figure 4: A conservative percolation example.

this example, the initial PR value of each document is 0.15, and the damping factor d is 0.85. Document A1 has a top PR rank after several iterations. If the PR rank of a document is higher than a predefined percolation threshold T_{per}, the feature words of the document are thereafter considered in the following content analysis. In this example, assume that only A1's PR rank is higher than T_{per}. The PR ranks of these documents are stable after several computation iterations. The feature words of document A1 is thereafter involved in the following augmented content analysis.

From the rank results, the neighbor documents can be classified into the following three categories: immediate neighbors, medium neighbors, and far neighbors. An *immediate neighbor* is defined as a neighbor whose link distance to the test document is less than or equal to a threshold D_{im}, and the PR rank is larger than or equal to the percolation threshold T_{per}. For example, in Figure 3, A1 is an immediate neighbor document of the test document D when D_{im} = 1. A *medium neighbor* is a neighbor whose link distance to the test document is still less than or equal to the distance threshold D_{im}, but has a PR rank lower than T_{per}. In our example, A2, A3 and A4 are D's medium neighbors. All other neighbors are called *far neighbors.*

Clearly, what we are interested are those immediate neighbors. They look likely to be the relevant pages and can provide auxiliary information of high quality. In fact, two

thresholds, T_{per} and D_{im}, control the quality. The percolation threshold T_{per} is used to control the link authority. Assigning T_{per} a higher value means that only the pages of high link authority will be considered in the following content analysis. Another distance threshold D_{im} is used to control the influence of topic drift. Since link analysis suffers the topic drift problem if far neighbor pages are crawled [4, 13], assigning D_{im} a higher value will filter out these far neighbors. Therefore, an optimistic classifier can be constructed by having a lower T_{per} and a larger D_{im}. A conservative classifier can be constructed vice versa. Figure 4 shows an example in which a conservative classifier is used. The immediate neighbors of the test document D are identified if their PR ranks are higher than the threshold T_{per} and their distances to the test document are less than D_{im}.

After the immediate neighbors are identified, their content is merged into the content of the test document for the further content analysis. If *C(p)* denotes the content of the page *p* and *C"* denotes the merged content, the merging procedure is performed to the test page *i* as described in the following

$$C'' = C(i) \cup \bigcup_{i \to j, PR(j) \geq T_{per}, d \leq D_{im}} C(j) \qquad (3)$$

where *i* ® *j* means there is a path from *i* to *j* and *d* is the path distance.

Then the correspondent CVVs and DVVs of the merged content are updated or created, and they are integrated together for representing the original test document. These CVVs and DVVs are used in the classification procedure. The calculation of all corresponding frequencies follows the original J&I approach.

4. Experimental Results

To understand the performance enhancement of neighbor text percolation, we have conducted similar experiments as in [8] to compare our approach with the original J&I approach. As their approach, we use Yahoo! pages under the category `Arts/Design_Arts/Architecture` to generate the vocabularies. The classification hierarchy for `Architecture` has 14 leaf nodes.

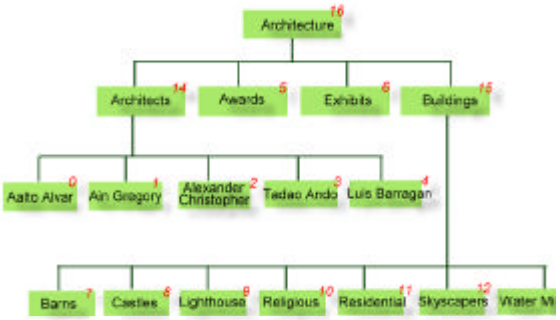


Figure 5: The classification hierarchy for `Architecture`.

Figure 5 depicts the classification hierarchy. The vocabularies of each leaf node are generated from approximately 20 training documents if possible. Two documents are selected randomly from each leaf node category and excluded from the training documents to generate the test corpus. There are 28 test documents in total. However, because some original test pages used in J&I's experiments are not accessible or moved, some test pages are newly added and the test corpus is somewhat different. We also notice that some pages originally misclassified in J&I's experiments can be correctly classified in our implementation of the J&I approach. Since we do not keep the history logs of these pages, we cannot conclude the reasons. Though the test corpus is changed and the classification results of some pages are different, the overall performance of the J&I approach in our experiments is still analogous to what was reported in their original paper [8].

In our experiments, we adopted a somewhat conservative policy that D_{im} was one to prevent topic drift. The PR rank of each neighbor document is 0.15, and the

Table 1: The classification performance of two approaches.

	Our Approach	J&I Approach
Classification score	68/80=85%	61/80=76.3%
Totally correctly classified	22 pages	19 pages
Partially correctly classified	5 pages	5 pages
Cannot be classified	1 page	4 pages

damping factor is 0.85. The threshold T_{per} is initially 0.85, but it may be decreased later to let at least one neighbor document be selected.

The scoring rules are the same as the rules used in [8]. Each classification is scored according to how many correct paths are followed. In our experiments, the total number of paths needed to be reach the correct leaf nodes happens to be the same in [8], which is 80. As listed in Table 1, our classifier scores 68 points out of a possible 80, but the J&I classifier scores only 61 points. The accuracy percentage gains an 11.5% improvement. The detail scores can be found in [18].

The number of pages that cannot be classified is also reduced. In our approach, 22 documents are classified into correct leaf categories, 5 documents are classified into partially correct categories, and only one document cannot be classified. In J&I' s approach, 19 documents are correctly classified into corresponding leaf categories, 5 documents are classified into correct upper categories but then into wrong leaf categories, and 4 documents cannot be classified.

These experimental results are positive to our auxiliary neighbor text percolation approach. However, there exist several problems needed further investigation. First, we have found that the classification scores of two documents are worse in our approach. This is because the neighbor text percolation still introduces noisy neighbor text. Furthermore, our approach cannot gain any improvement if the test page has no links. In addition, if a neighbor is dynamically generated, the results of our approach may be highly perturbed.

5. Conclusions

Automatic Web document classification plays an important role in helping users to effectively find highly related documents. The classification approaches can be primarily divided into three categories: the content-based analysis approach, the link-based analysis approach, and the hybrid approach. Solely using content-based analysis or link-based analysis has been shown that the classification precision may be hurt due to their native limitations [2, 7, 8, 13].

Some hybrid algorithms are proposed to improve the classification precision by combining link-based analysis schemes and content-based analysis schemes [2, 4, 9, 11]. They can be further classified into two sub-categories. One is to reinforce link-based analysis with content-based analysis, and the other is vice versa. However, the link-based algorithms with reinforcement of content-based analysis may suffer the mixed hubs problem [11] and the unstable performance problem [14, 15].

The content-based algorithms with reinforcement of link-based analysis inherit high precision performance from content-based analysis. However, simply adding neighbor text into content analysis can even increase the classification error rate due to too many noisy feature words. It is not trivial to utilize neighbor text by link analysis to facilitate classification.

In this paper, we present an enhanced automatic classification algorithm and its preliminary experimental results. Our approach is based on the content-based analysis scheme of the J&I approach with an

extension of a PageRank-like function for neighbor text percolation. With the percolation function, highly relevant neighbor text is filtered out and pages of sparse self-descriptive content can be highly alleviated by the percolated neighbor text.

To study the performance enhancement of neighbor text percolation, we have conducted similar experiments as in [8] to compare our approach with the original J&I approach. From the experiments, the results show that our approach indeed improves the original J&I approach. However, there are still many parameters that need to be adjusted manually in our approach. In addition, the scale of the experiments is still limited. The performance of our percolation scheme needs to be further evaluated with large-scaled test corpora. A more accurate percolation scheme is our future work.

References

1. C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal (1997), "Web Search Using Automatic Classification," 6th International World Wide Web Conference.
2. K. Bharat and M. R. Henzinger (1998), "Improved Algorithms for Topic Distillation in Hyperlinked Environments," The 21st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR' 98), ACM, pp104-111.
3. S. Chakrabarti, B. E. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins (1998), "Experiments in Topic Distillation," The ACM-SIGIR Workshop on Hypertext Information Retrieval for Web, ACM.
4. S. Chakrabarti, B. E. Dom, and P. Indyk (1998), "Enhanced Hypertext Categorization Using Hyperlinks," the ACM SIGMOD International Conference on Management of Data, ACM, pp307-318.
5. J. Dean and M. R. Henzinger (1999), "Finding Related Pages in the World Wide Web," the 8th International World Wide Web Conference, pp389-401.
6. T. Guan and K.-F. Wong (1999), "KPS - a Web Information Mining Algorithm," the 8th International World Wide Web Conference, pp1495-1507.
7. J. M. Kleinberg (1999), "Authoritative Sources in a Hyperlinked Environment. Journal of the ACM, 46(5), pp604-632.
8. C. Jenkins and D. Inman (2000), "Adaptive Automatic Classification on the Web," 11th International Workshop on Database and Expert Systems Applications, pp504-511.
9. Y.-H. Kuo and M. H. Wong (2000), "Web Document Classification based on Hyperlinks and Document Semantics," PRICAI 2000 Workshop on Text and Web Mining, pp44-51.
10. O.-W. Kwon and J.-H. Lee (2000), "Web Page Classification based on k-Nearest Neighbor Approach," the 5th International Workshop on Information Retrieval with Asian Languages, ACM, pp9-15.
11. S. Chakrabarti, M. Joshi, and V. Tawde (2001), "Enhanced Topic Distillation using Text, Markup Tags, and Hyperlinks," the ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp208-216.
12. S. Dumais and H. Chen (2000), "Hierarchical Classification of Web Content," the ACM SIGIR Conference on Research and Development in Information Retrieval, pp256-263.
13. F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz (2001), "Evaluating Topic-Driven Web Crawlers," the ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp241-249.

14. A. Y. Ng, A. X. Zheng, and M. I. Jordan (2001), “Link Analysis, Eigenvectors and Stability,” the 7th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, pp903-910.
15. A. Y. Ng, A. X. Zheng, and M. I. Jordan (2001), “Stable Algorithms for Link Analysis” the ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp258-266.
16. S. Brin and L. Page (1998), “The Anatomy of Large-Scale Hypertextual Web Search Engine,” 7th International World Wide Web Conference.
17. M. R. Henzinger (2001), “Hyperlink Analysis for the Web,” IEEE Internet Computing, 5(1), pp45-50.
18. S.-M. Dai (2002), “Link-based Automatic Web Page Classification,” Master Thesis, Dept. of Computer Engineering and Science, Yuan Ze University.