Optimizing Personalized Retrieval System Based on Web Ranking

Hao-ming Wang[1,3], Ye Guo[2], and Bo-qin Feng[1]

[1] School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, P.R. China
{wanghm, bqfeng}@mail.xjtu.edu.cn
[2] School of Information, Xi'an University of Finance & Economics, Xi'an, Shaanxi 710061, P.R. China
guoyexinxi@126.com
[3] School of I & C, Swiss Federal Institute of Technology(EPFL), 1015 Lausanne, Switzerland

Abstract. This paper drew up a personalized recommender system model combined the text categorization with the pagerank. The document or the page was considered in two sides: the content of the document and the domain it belonged to. The features were extracted in order to form the feature vector, which would be used in computing the difference between the documents or keywords with the user's interests and the given domain. It set up the structure of four block levels in information management of a website. The link information was downloaded in the domain block level, which is the top level of the structure. In the host block level, the links were divided into two parts, the inter-link and the intra-link. All links were setup with different weights. The stationary eigenvector of the link matrix was calculated. The final order of documents was determined by the vector distance and the eigenvector of the link matrix.

1 Introduction

Applying the peer-to-peer architectural paradigm to Web search engines has recently become a subject of intensive research. Whereas proposals have been made for the decomposition of content-based retrieval techniques, such as classical text-based vector space retrieval or latent semantic indexing, it is much less clear of how to decompose the computation for ranking methods based on the link structure of the Web.

As a user, in order to find, collect and maintenance the information, which maybe useful for the specific aims, s/he have to pay more time, money and attention on the retrieval course.

There are many search engines, such as Yahoo, Google, etc. to help the user to search and collect the information from the Internet. The features of the Internet, such as mass, semi-structure, have become drawbacks in using the information widely in Internet.[1]

D. Grigoriev, J. Harrison, and E.A. Hirsch (Eds.): CSR 2006, LNCS 3967, pp. 629–640, 2006.

In order to make the information retrieval system more efficient, the artificial intelligence (AI) technologies have been suggested to use. The IR models can be divided into two items, the one is based on the large scales machine learning, and another is based on the intelligent personalization.

For the user interested in the special domain, for example, the computer science or architectonics, it may be a good way to support the results by using intelligent personalization.

This paper describes the architecture of information retrieval model based on the intelligent personalization. The aim is to provide the information related to the given domain. The model receives the request of user, interprets it, selects and filters the information from Internet and local database according to the profile of the user. The user profile is maintained according to the feedback of the user. The pagerank values of documents are computed before they are stored.

This paper is organized as follows: Section 2 introduces the basic concept of TFIDF and the Pagerank. Section 3 describes the architecture of the new system. Section 4 discusses the process of re-ranking. Section 5 introduces the query process. Section 6 discusses the shortcoming and the room for improvement of the system, and section 7 gives the conclusion.

2 Basic Concept

2.1 TFIDF and Text Categorization

TFIDF (Term Frequency / Inverse Document Frequency) is the most common weighting method used to describe documents in the Vector Space Model (VSM), particularly in IR problems. Regarding text categorization, this weighting function has been particularly related to two important machine learning methods: kNN (k-nearest neighbor) and SVM(Support Vector Machine). The TFIDF function weights each vector component (each of them relating to a word of the vocabulary) of each document on the following basis.

Assuming vector $\tilde{d} = (d^{(1)}, d^{(2)}, ..., d^{|F|})$ represents the document d in a vector space. It is obviously that documents with similar content have similar vector. Each dimension of the vector space represents a word selected by the feature selection.

The values of the vector elements $d^{(i)}(i \in (0, |F|))$ for a document d are calculated as a combination of the statistics $TF(w, d)$ and $DF(w)$ (document frequency).

$TF(w, d)$ is the number of word occurred in document d. $DF(w)$ is the number of documents in which the word w occurred at least once time. The $IDF(w)$ can be calculated as

$$IDF(w) = log\frac{N_{all}}{DF(w)}.$$

Where N_{all} is the total number of documents. Obviously, the $IDF(w)$ were low if w occurred in many documents and it were the highest if w occurred in only one.

The value $d^{(i)}$ of feature w_i for the document d is then calculated as

$$d^{(i)} = TF(w_i, d) \times IDF(w_i).$$

$d^{(i)}$ is called the weight of word w_i in document d. [9-12]

The TFIDF algorithm learns a class model by combining document vectors into a prototype vector $\tilde{C}$ for every class $C \in$ Ç. Prototype vectors are generated by adding the document vectors of all documents in the class.

$$\tilde{C} = \sum_{d \in C} \tilde{d}.$$

This model can be used to classify a new document d'. d' can be represented by a vector $\tilde{d'}$. And the cosine distance between the prototype vector of each class and $\tilde{d'}$ is calculated. The d' is belonged to the class with which the cosine distance has the highest value.

$$H_{TFIDF}(d') = argmaxcos(\tilde{d'}, \tilde{C}).$$

Where $H_{TFIDF}(d')$ is the category to which the algorithm assigns document d'.

$$H_{TFIDF}(d') = argmax(\frac{\tilde{d'} \cdot \tilde{C}}{\|\tilde{d'}\| \cdot \|\tilde{C}\|}) = argmax(\frac{\sum_{i=1}^{|F|}[d'^{(i)} \cdot C^{(i)}]}{\sqrt{\sum_{i=1}^{|F|}[d'^{(i)}]^2} \cdot \sqrt{\sum_{i=1}^{|F|}[C^{(i)}]^2}}).$$

2.2 Pagerank

For a search engine, after finding all documents using the query terms, or related to the query terms by semantic meaning, the result, which maybe a large number of web pages, should be managed in order to make this list clearly. Many search engines sort this list by some ranking criterion. One popular way to create this ranking is to exploit the additional information inherent in the web due to its hyper linking structure. Thus, link analysis becomes the means to ranking. One successful and well publicized link-based ranking system is PageRank, the ranking system used by the Google search engine.

The Google search engine is based on the popular PageRank algorithm first introduced by Brin and Page in Ref.[6]. The algorithm can be described as:

Let u be the web page. Then let F_u be the set of pages u points to and B_u be the set of pages that point to u. Let N_u be the number of links from u and let c be a factor used for normalization (so that the total rank of all web pages is constant).

$$R(u) = c \sum_{v \in B_u} \frac{R(v)}{N_v}.$$

This algorithm displays that the pagerank of page u comes from the pages that point to it, and u also transfers its pagerank to the pages which it points to.

Considering the pages and the links as a graph $G = P(Page, Link)$, we can describe the graph by using the adjacency matrix. The entries of the matrix, for example p_{ij}, can be defined as:

$$p_{ij} = \begin{cases} 1 & Link \quad i \to j \quad exists \\ 0 & Otherwise. \end{cases}$$

Here $i, j \in (1, n)$ and n is a number of web pages. Because the total probability from one page to others can be considered 1, the rows, which correspond to pages with a non-zero number of out-links $deg(i) > 0$, can be made row-stochastic (row entries non-negative and sum to 1) by setting $p_{ij} = p_{ij}/deg(i)$. That means if the page u has m out-links, the probability of following each of out-links is $1/m$. We assume all the m out-links from page u have the similar probability. Actually, there is difference among them, and one link maybe more important than others. We can assume the probability list,

$$\{w_1, w_2, \cdots, w_m; \sum_{i=1}^{m} w_i = 1\}.$$

which can promise the pagerank more precise.

If we considered the property of the adjacency matrix, we could find the adjacency matrix correspond to a Markov chain.

According to the Chapman-Kolmogorov Equations, for the Markov chains, we can get

$$p_{ij}^{n+m} = \sum_{k=0}^{\infty} p_{ik}^{n} p_{kj}^{m} \quad (n, m \geq 0, \forall i, \forall j)$$

If we let $P^{(n)}$ denote the matrix of $n - step$ transition probabilities p_{ij}^{n}, then we can asserts that

$$P^{(n+m)} = P^{(n)} \cdot P^{(m)};$$

$$P^{(2)} = P^{(1)} \cdot P^{(1)} = P \cdot P = P^2;$$

$$P^{(n)} = P^{(n-1+1)} = P^{(n-1)} \cdot P^{(1)} = P^{n-1} \cdot P = P^n.$$

That is, the $n - step$ transition matrix can be obtained by multiplying the matrix P by itself n times.

The case discussed above is ideal. For a real adjacency matrix P, in fact, there are many special pages without any out-link from them, which are called *dangling page*. Any other pages can reach the dangling page in $n(n \geq 1)$ steps, but it is impossible to get out. The dangling page is called absorbing state. In the adjacency matrix, the row, corresponding to the dangling page is all zeros. Thus, the matrix P is not a row-stochastic. It should be deal with in order to meet the requirement of the row-stochastic.

One way to overcome this difficulty is to slightly change the transition matrix P. We can replace the rows, all of the zeros, with $v = (1/n)e^T$, where e^T is

the row vector of all 1s and n is the number of pages of P contains. The P will be changed to $P' = P + d \cdot v^T$. Where

$$d = \begin{cases} 1 & if \quad deg(i) = 0 \\ 0 & Otherwise. \end{cases}$$

is the dangling page indictor. If there were a page without any out-link from it, we could assume it can link to every other pages in P with the same probability. After that there is not row with all 0s in matrix P'. P' is row-stochastic. [26]

Because P' corresponds to the stochastic transition matrix over the graph G, Pagerank can be viewed as the stationary probability distribution over pages induced by a random walk on the web. The pagerank can be defined as a limiting solution of the iterative process:

$$x_j^{(k+1)} = \sum_i P'_{ij} x_i^{(k)} = \sum_{i \to j} x_i^{(k)} / deg(i).$$

Because of the existing of zero entries in the matrix P', it cannot be insure the existence of the stationary vector. The problem comes from that the P' may be reducible.

In order to solve the problem, P' should be modified by adding the connection between every pair of pages.

$$Q = P'' = cP' + (1-c)ev^T, \quad e = (1, 1, \cdots, 1)^T.$$

Where c is called dangling factor, and $c \in (0, 1)$. In most of the references, the c is set [0.85,1).

After that, it can be consider that all of the pages are connected (Strong connection). From one of the pages, the random surfer can reach every other page in the web. The Q is irreducible. For $Q_{ii}^{(k)} > 0, (i, k \in [1, n])$, the Q is aperiodic.

Above all, the matrix Q is row-stochastic, irreducible and aperiodic. The Perron-Frobenius theorem guarantees the equation $x^{(k+1)} = Q^T x^{(k)}$ (for the eigensystem $Q^T x = x$) converges to the principal eigenvector with eigenvalue 1, and there is a real, positive, and the biggest eigenvector.

3 Architecture of the System

The goal of this system is to help the users to find the information on Internet easily and quickly. It requests the system should process the information resource independently, gather the information the users are interested in, filter the repeating one, wipe off the useless one, and store them to the local database. This system should be feasible, friendly, adaptable, and transplantable.

The system would not take the place of Yahoo or Google, it will be an entrance of personalized search engine. With the interaction between the user and the system, it will gather the personalized information of the user. Fig.1 shows

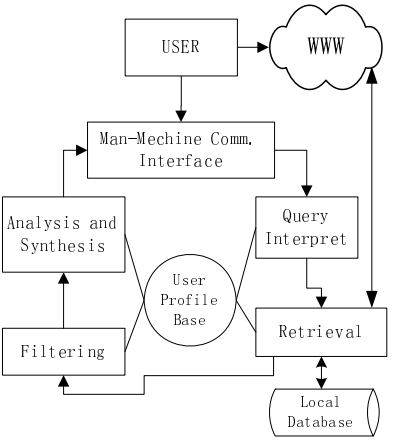

Fig. 1. The architecture of the system

the architecture of the system. The architecture is composed of five components, local database, and a user profile base. The architecture of the system is described in details as follows:[4]

(1) Man-Machine Communication component (MM): this component serves as the communication interface between the user and the system. The user inputs the keywords or other kinds of the requests to the system and receives the results from the interface.

(2) Query Interpret component (QI): this component enhances the user's query based on the user profile. Due to the difference of the knowledge and the domain, not all of the users can express his/her request exactly. As an information retrieval system, it should decide what kinds of sources should to be queried, how to modify the query expression the user has submitted in order to utilize the underlying search engines more, and how to order the feedback results.

(3) Retrieval component (RE): this component sends the requests and gets the information from Internet and the local database. There are two expectations in information retrieval, the *precision* and the *recall*. The precision shows the degree of the feedback results case to the user's needs. The recall shows the percentage of the feedback results to the total records, which are related to the user's needs. The results from the Internet maybe more general and the results from the local database maybe more accurate. In this system, the output list will be combined by the two kinds of results.

(4) Filtering component (FI): this component filters the raw data from the RE based on the user profile. The feedback results from the Internet are the raw data according to the keywords and query expression. They cannot meet the user's needs satisfactorily. This component filters the results according to the domains or the subjects the user interested. The related data of user's feature and the domain have been stored in the User Profile Base.

(5) Analysis & Synthesis component (AS): this component uses the filtered information to enhance decision making, uses data mining techniques and the user profile to analyze and synthesize the information retrieved.

(6) User Profile Base (UPB): a knowledge base for the users. There are two kinds of forms.

- The one is the holistic user profiles for all the users of a system; this conceptual profile base can either be distributed across the system or stored in a central location, the holistic user profile consists of a personal profile, a functional area profile, a current project profile, an organizational environment profile, and a client-type profile.
- Another is the storing feature or the visiting model of the specifically kind of the user. The initial data are set by manual. When the user visits the Internet, the historical pages are recorded and downloaded. The system extracts the information from the pages, extracts the class keywords, and constructs the vector of every page. The distance between the page vector and the domain vector is calculated. According to the distance, the user's personalize model is built. This model will be refined based on the feedbacks the user interested.

(7) Local Database (LD): it stores the data, which have been downloaded from the Internet according to the historical pages. Most of the knowledge on Internet are non-structure or semi-structure. They are different from the data stored in the local relational database. Most of the non-structure and semi-structure data are organized as the natural language model. Those data should be converted before they are stored to the relational database, after that, they can be shared and utilized effectively.

4 Re-ranking the Pages

In this section, we introduce two kinds of calculation used in this system. The first is the similarity computing of the vector, and the second is the pagerank re-computing.

4.1 Extracting Keywords

In our system, we should select the keywords from the given domain. In the traditional algorithm of text categorization or recommender system, all the terms (words) are considered, and the importance of each term is decided by the numbers of it appeared in the documents. But actually, some terms in a given domain maybe more important than the others.

The Ref.[9][15] introduces a new weighting method based on statistical estimation of the importance of a word for a specific categorization problem. This method also has the benefit to make feature selection implicit, since useless features for the categorization problem considered get a very small weight. Extensive experiments reported in the paper shows that this new weighting method

improves significantly the classification accuracy as measured on many categorization tasks.

In our system, the method can be described as selecting the $top-n$ keywords from the paper set, which have been categorized in manual, by calculating the weight of every word.

We donated the keywords list with the vector:

$$\widetilde{D_i} = \{(k_j, w_j), j \in (1, m)\}, i \in (1, n)$$

Where n is the numbers of domains, m is the numbers of keywords in a given domain, (k_j, w_j) is the keyword and its weight in the given domain, and $\widetilde{D_i}$ is the vector of the domain D_i. The detail can be found in the Ref.[16].

Similarly, we denote the document d with the vector $\widetilde{d}$:

$$\widetilde{d_i} = \{(k_j, w_j), j \in (1, m)\}, i \in (1, n)$$

Where n is the numbers of documents, m is the numbers of keywords in a given domain, (k_j, w_j) is the keyword and its weight in the given document, and $\widetilde{d_i}$ is the vector of the document d_i.

In the next section, we will replace the domain D, the document d with the corresponding vector $\widetilde{D}$ and $\widetilde{d}$.

4.2 Downloading Links

According the log file of the server, we can get the visiting queue. In this section, we will download the links between the pages in domain level. In Fig.2, we divide the information management into 4 block levels. They are, 1^{st}: pages; 2^{nd}: Directories. Such as $http://liawww.epfl.ch/Research$; 3^{rd}: Host. Such as $http://liawww.epfl.ch$; 4^{th}: Domain. Such as $http://www.epfl.ch$.

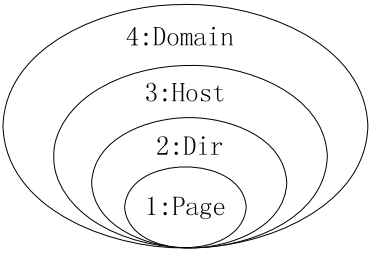

Fig. 2. Levels of Information Management

The Ref.[14] points out that in order to get the best performance, the block level should be set at host. We download the links in domain level in order to calculate the pagerank in host level. We should set up a crawl to download the links. The Ref.[17] showed how to set up and manage the crawls.

4.3 Setting Up the Link Matrix

In section 2.2, we introduce the calculating of pagerank, which is used by Google.

In the algorithm, the entry $p_{ij} = 1$ when the link $i \rightarrow j \quad exists$. The Ref.[5][7-8][13-14]drew the methods to improve the effects of calculating.

In our system, we consider the host as a block level. The out-links can be divided two parts: the intra-links and the inter-links. The former one mean that the pages linked to and from are belonged to the same host. Otherwise they are belonged to the latter one.

The weight of two kinds out-link are different. We assume the total weight of all out-link pages is 1. All the intra-links share the 3/4 weight, and all inter-links share the 1/4 weight. The link matrix P will be changed to:

$$p_{ij} = \begin{cases} \frac{3}{4*dig(intra)} & i,j \in H_m; \\ \frac{1}{4*dig(inter)} & i \in H_m, j \in H_n, m \neq n; \\ 0 & Otherwise. \end{cases}$$

The P should be dealt with just fellow the steps discussed above to $P''(or \quad Q)$ in order to promise $Q^{(k)}$ converge and have a real, positive and the biggest eigenvector.

4.4 Re-ranking

In the local database, there are: (1)the pages downloaded from the website according to the log file, (2) the vectors of the pages and the domains, (3)the relating link information of the website. There are two lists, (1)the order of the pages, which related to the difference between domain's and page's vector , and (2)the importance (or probability) of the pages in the given domain.

In this section, we combine the list(1) and (2) in order to adjust the order of the pages stored in local database.

$$score(d) = \alpha * sim(\widetilde{d}, \widetilde{D}) + (1 - \alpha) * PR(d).$$

Where d is the page stored in local database, $\alpha \in (0, 1)$ is the adjust factor, $PR(d)$ is the Pagerank of d, and $sim(\widetilde{d}, \widetilde{D})$ is the difference between the page d and the domain $\widetilde{D}$.

The result of the computation is a new order list of the pages, which will be stored in local database, too.

5 Query Process

In this section, we introduce the query process.

(1) Extracting the domain vectors

We use the Open Directory Cluster as the reference. We assume there are m domains. For each domain, we select n papers related to the domain in order to obtain the vector $\widetilde{D_i}(i \in (1, m))$ of given domain. Obviously, the work should be done once only.

(2) Login

This system will be an entrance of personalized search engine. If the user wanted to apply the service, he should register and submit the basic information of himself. The system will create a new user profile in order to record the habit of the new user. If the use wanted not to register, he could visit the Internet freely.

(3) Submitting the keywords

Just as the description in Fig.1, the MM component receives the keywords k and modifies them according to the user's profile. It will do nothing when the user entered the system in the first time. The QI component will submit the keywords to the search engine directly. But in the other times, the QI component will compute the two kinds of difference:

- between k and the domain vector $\widetilde{D_i}$. The m nearest neighbors have the most probability of the keywords belonged to. Assuming the domain list is $D_{ca} = \{\widetilde{D_{k1}}, \widetilde{D_{k2}}, \cdots \widetilde{D_{kl}}, (kl \leq m)\}$.
- between k and the document vectors $\widetilde{d_i}$, $\widetilde{d_i} \in \widetilde{D_i}(\widetilde{D_i} \in D_{ca})$.

The RE component will list all the $top - n$ pages of every domain $\widetilde{D_i}(\widetilde{D_i} \in D_{ca})$ stored in local database. Meanwhile, the RE will send the keywords to the search engine.

All of the results (pages, documents) will be order by the FI component in order to output. The user can select the pages he was interested in, and the log file will record it simultaneity.

(4) Maintaining the Use Profile and Local database

After the user logout, the system will download the link information of the domain (Show in Fig.2) and the pages, which the user has visited. All of the computation will be done by the AS component, and the local database and user profile will be maintained.

6 Shortcoming and Room for Improvement

There are two sides need to be improved, the feature extraction and the pagerank computation.

In our system, we use the traditional algorithm to extract the features of the document and the domain. The feature is the word appeared in the document. Just as we know, in a given document. There are many words have a more importance or have a bigger weight than others do. In this algorithm, we do not consider this condition.

We simulate the pagerank algorithm just as the Google did. Different from the real computation, we centralize the pagerank among the domain, the 4^{th} block level. We hardly consider the influence from and to other domains. It makes the warp from the real one. This model may not respond to the real importance from the link view.

For the pagerank computing, two questions are put forward in Ref.[5]. One is that a page may correlative with the given subject, but might not contain

the keywords of query. It makes the page failed to be select by the query. The other question is that some websites might contain a lot of hyperlinks or had a high pagerank, but it might be less correlative with the keywords. The Google claimed that the second problem had been solved. But we have not found the report about how to do that in recent papers.

For the SVM method, it is effective when the pages have one kind of schema. There are many kinds of mass, business information in Internet, and they are arranged in another way different from those pages of science and technology. For those pages, another method should be developed, which includes the information analysis, storing and issuing, too.

7 Conclusion

This paper discussed the disadvantages, which could not provide the personalized service, of the current search engines. It drew up a new personalization model, which combine the text categorization with pagerank computation.

The keywords of the given domain and the documents were extracted in order to form the vectors, which will be used in text categorization. The links of the domain, which in the top of the block level of a website, were downloaded. The model changed the weight of the link in order to distinguish the intra-links and the inter-links of a host. The pagerank was computed, and it was combined with vector difference to form the order list of the page in the given domain.

In the last paragraph, it pointed out that there were some problems should be solved in order to make the retrieval results more veraciously.

Acknowledgements

This work was supported by project 2004F06 of Nature Science Research of Shaanxi Province, P.R. China.

References

1. WANG Ji-Cheng, et al. State of the Art of Information Retrieval on the Web. Journal of Computer Research & Development. Vol.382001(2)pp.187-193.
2. James Kempf. Evolving the Internet Addressing Architecture. Proceedings of the 2004 International Symposium on Applications and the Internet (SAINT'04).
3. D.Rafiei, A.Mendelzon. What is this page known for Computing web page reputations. In 9th International World Wide Web Conference, Amsterdam, Netherlands, May 2000.
4. Neal G. Shaw, et al. A comprehensive agent-based architecture for intelligent information retrieval in a distributed heterogeneous environment. Decision Support Systems 32 (2002) pp.401-415.
5. R.Lempel, S.Moran. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In 9th International World Wide Web Conference, Amsterdam, Netherlands, May 2000.

6. L. Page, S. Brin, R. Motwani, T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Stanford Digital Libraries Working Paper, 1998.
7. T. Haveliwala. Efficient computation of PageRank. Technical report, Computer Science Department, Stanford University, 1999.
8. Wenpu Xing, Ali Ghorbani. "Weighted PageRank Algorithm," cnsr, pp. 305-314, Second Annual Conference on Communication Networks and Services Research (CNSR'04), 2004.
9. Pascal Soucy,Guy W. Mineau. Beyond TFIDF Weighting for Text Categorization in the Vector Space Model. In proceeding of IJCAI-05. pp.1136-1141.
10. Gongde Guo, Hui Wang, et. al . An kNN Model-Based Approach and Its Application in Text Categorization. In Proceeding of CICLing 2004: 559-570.
11. Ralf Steinberger, Bruno Pouliquen, Johan Hagman. Cross-Lingual Document Similarity Calculation Using the Multilingual Thesaurus EUROVOC. In Proceedings of Computational Linguistics and Intelligent Text Processing (CICLing 2002). pp. 415-424.
12. LI-PING JING, HOU-KUAN HUANG. Improved feature selection approach tfidf in text mining - Machine. In Proceedings of the First International Conference of Machine learning and Cybernetics, 2002. pp.944-946.
13. Monica Bianchini, Marco Gori, Franco Scarselli. Inside Pagerank. ACM Transactions on Internet Technology. Vol.(5), No.1,2005(2). pp. 92-128.
14. Xue-Mei Jiang, Gui-Rong Xue, Wen-Guan Song, Hua-Jun Zeng, Zheng Chen, Wei-Ying Ma. Exploiting PageRank at Different Block Level. In Proceedings of WISE 2004. pp. 241-252.
15. B. Arslan, F. Ricci, N. Mirzadeh, A. Venturini. A dynamic approach to feature weighting. In Proceedings of Data Mining 2002 Conference.
16. Thorsten Joachims. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In Proceedings of the Fourteenth International Conference on Machine Learning table of contents(ICML). 1997. pp.143 - 151.
17. http://www.searchtools.com/robots/robot-code.html