An Architectural Framework of a Personalized Web Crawler based on User Interests

J. AKILANDESWARI[1]
N.P. GOPALAN[2]

[1]Sona College of Technology
Department of Computer Science and Engineering
Salem - 636005, Tamilnadu, India
akila_rangabashyam@yahoo.com
[2]National Institute of Technology
Department of Computer Applications
Tiruchirappalli - 620015, Tamilnadu, India
gopalan@nitt.edu

Abstract. The World Wide Web (WWW) is overwhelmed with information which can not be assimilated by the normal users without the use of search tools. The traditional search returns thousands of results for a single search query making the search and surfing experience cumbersome. This drawback has triggered the need for implementing personalized search tools. In this paper, a novel architecture is proposed to gather pages that are relevant to a particular user or group of users. The system consists of three modules: input, crawling and feedback. The input module is integrated with topic suggestion component extracting search query terms and representative documents from different sources. The crawling module is realized with intelligent multi-agent system for prioritizing the download of appropriate URLs. The relevance of the documents is computed based on interests of the users. While rendering the results, the user gives feedback and the system is compared to different crawler implementations. The empirical results clearly suggest the advantage of using topic suggestion component and computation of personalized relevance score in terms of harvest ratio and coverage.

Keywords: Personalized Crawler, Multi-agent system, URL Ordering, Multi-level frontier queue, Web mining, Classification

(Received December 31, 2008 / Accepted April 26, 2009)

1 Introduction

Due to the enormous growth of World Wide Web, users usually prefer search engines to locate a Web page of their interest. Search engines try to index as many Web pages as possible. Yahoo! claims that it had indexed over 20 billion Web pages while Google had indexed three times more than its competitor [5]. Most of the existing Web search engines return a list of search results based on a query but disregard the user's specific interests and/or search context. Therefore, the identical query from different users or in different contexts will generate the same set of results displayed in the same way for all users, a so called one-size-fits-all approach. A user may have to go through many irrelevant results or try several queries before finding the desired information. Problems encountered in searching are exaggerated further when the search engine users employ short queries. An alternative to this approach can be a personalized search tailored for the needs of specific users. Its essentiality is felt as they provide the location of information on the WWW as accurately as possible, using the methodologies of data mining and knowledge discovery. The Web has now become intelligent in terms of serving the users with the documents they prefer. Creating such an environment requires new tools and infrastructure components which includes agent technology and various soft computing techniques. The development of an intelligent Web using computational intelligence and mining techniques has been an active research topic [9].

Text-based search engines compute document's relevance using the query text position and frequency of its occurrence in the document. For instance, to locate a Web page about 'data mining', a text based search engine will give a list of Web pages containing that text, even though they might not be relevant to the user. Also, this technique allows the spammers to stuff important keywords in the Web document solely with the purpose of increasing the site's rank. The computation of rank using link structures reduces the problem described above. Most of the empirical studies say that Google outperforms in technology of gathering Web pages and indexing methods because of its ranking algorithm which exploits link structures. Nowadays, people began to use more specialized search tools which will fetch only those URLs that are more important to them. Because of the size of the Web, developing such a personalized search crawler that utilizes the link structures for ranking the documents is a very challenging task.

For designing a crawler that is personalized to a particular user or group of users, the crawler has to focus on certain topic(s). Unlike general purpose Web crawlers, focused crawlers crawl only on particular topical portions of the Web. It is therefore very useful for the circumstances where the interest lies only on certain domain. A focused crawler tries to foresee whether a target URL is pointing to a relevant and high-quality Web page before actually fetching that page. Many researchers have employed Machine Learning strategies to improve crawler's functionalities in many aspects. Classification is one such approach used in determining the relevance of a Web page and in building and maintaining Web directories or portals. In semi-supervised machine learning approaches like reinforcement learning, the crawlers learn progressively by interacting directly with the dynamic environment. Because of the inherent dynamism in WWW, the crawlers are never told about the correct action, instead they are told about how good or bad its action was [2].

Agent technology is preferred for retrieving information in a large environment such as the Web. Unlike objects, an agent is defined in terms of its autonomous behavior. This technology stresses in autonomy oriented computing for modeling multiple entities and the self organization of them towards a specific goal. Intelligent agents can be developed in an efficient way because of their use of memorized information. Given that a single agent approach may be inefficient and impractical for a large-scale Information Retrieval environment, most of the systems employ multi-agent systems. The multi-agent framework supports cooperative search and have the potential of parallelism, robustness and scalability. In multi-agent systems, communication and organization enables the agents to cooperate and coordinate their actions to accomplish a common goal. There are a number of communication languages like KIF (Knowledge Interchange Format) [14], KQML (Knowledge Query and Manipulation Language) [13], and ACL (Agent Communication Language) [20].

This paper presents an architectural framework for locating relevant Web documents for a specific user or group of users. The personalization is incorporated in the input and crawling modules. The input module consists of a topic suggestion component that extracts search query terms from different sources. The crawler module is realized with two types of agents: retrieval agents and coordinator agent. The coordinator agent is built with multi-level frontier queue to achieve tunneling and the URLs stored there are disseminated to retrieval agents. The retrieval agents download and classify Web pages as relevant or irrelevant using personalized relevance score.

The remainder of this paper is organized as follows: section 2 discusses on the work related to focused Web crawlers. In section 3, the architectural framework of the crawler is described. In section 4, experimentation and evaluation details are discussed. Finally in section 5, conclusions are presented.

2 Related Works

There are several works on crawler designs and strategies [10],[16], [25], [11],[28]. There are research works that exploited structural similarities between the Web and modeled WWW as social networks to develop techniques or methodologies for enhancing the search experience [19]. The Web in many ways simulates a social network: links do not point to pages at random but endorses the page author's idea of what other relevant or interesting pages exist. This information can be exploited to collect more on-topic data by intelligently choosing what links to follow and what pages to discard.

One of the pioneering approaches in ordering the URLs according to the relevance was Fish search [7]. The system was query-driven and text-based which considered only those pages that were matching the query and links that emanated from those pages. An improvement of Fish Search was proposed as Shark search [15]. The algorithm used weighted term frequency (TF) and inverse document frequency (IDF) measure to determine page relevance score. In [17] a technique was proposed to reorder the URLs in the frontier queue according to various heuristics like page rank, *in* link count, *back* link count and combination of these features.

A soft focused crawler is proposed in [10]. This technique used a classifier to obtain a score. The main shortcoming of this technique is that it will not support tunneling i.e. following a path of off-topic pages.

A context-graph based crawler described in [11] and Cora's crawler [22] used tunneling concept. Cora is a domain-specific search engine whose spider is incorporated with reinforcement learning algorithm for progressively capturing the dynamic environment. There are other systems like Web Topic Management System [24] that fetched only those pages that were parent, child or sibling to on-topic pages. This system has a short coming of meeting a dead end in the earlier stage of crawling. Bingo! [29] is a crawling system that eliminated the initial training step and employs continual training of the classifier with high quality pages. Menzcer et al. [23] presented a framework to evaluate focused crawlers and developed a crawler based on evolutionary concepts. An information integration framework ALII is presented in [6] which used active logic. ALII, employs a compact context representation and build a hierarchy model of query and Web pages. The crawler has also made limited backward crawling using general search engine indices.

An intelligent crawling architecture was presented by Aggarwal et al. [1] based on predicates. It applies a self-learning mechanism that can dynamically adapt to the particular structure of the relevant predicate. For building a composite crawler several factors are considered during the crawl to evaluate its effectiveness.

A crawling algorithm is presented by Ehrig et al. [12] based on ontology and computed page relevance according to a particular user's need. Entities (words) occurring in the ontology are extracted from the page and counted. Relevance of the page is computed with regard to user selected entities using several measures on ontology graph like direct match, taxonomic and more complex relationships. The experimental results of this system has shown improvements in harvest rate when compared to baseline focused crawler that computed page relevance by a simple binary keyword match. Case based BDI-agent [27] is a domain specific search engine that made use of Case Based Reasoning (CBR) as its learning component. The system has used past results and reused that for answering future queries. A personalized focused crawler is proposed by Kim et al. [18] which has exploited both link structures and fuzzy concept networks. They have modeled the user interests as fuzzy concept networks and retrieved the results in a personalized manner. Altingovde et al. [4] proposed an efficient algorithm to order the URLs in frontier queue based on rule discovery method and their results showed an improvement in coverage.

AutoCrawler (Automatic Topical Crawler), developed by Tsay et al. [30] is an integrated crawling system that consists of a user interest specification module to mediate between users and search engines in identifying target examples and keywords that together specify the topic of their interest and URL ordering. AutoCrawler is designed to have a topic specification module, a classifier learning module, an URL ordering module and an analysis module. The user interests can be specified using two approaches: taxonomy-based and keyword-based. In taxonomy-based approach, users select their interest from topics of a predefined taxonomy. In keyword based approach, interest is specified by keywords that define the targets of the interest. In AutoCrawler user interests are specified using taxonomy based approach and through a set of target examples and a set of keywords that together specify the topic of their interest. Similar to the proposed work, Autocrawler combines term suggestion, query modification and document ranking. But the it lacks in ordering the URLs based on user's need. In [3] a profile representation is described using Internet domain features extracted from URLs. Users are required to specify interest profiles as binary vectors, where each feature corresponds to a set of one or more DNS tree nodes. Given a profile vector, weighted pagerank is computed assigning a weight to each URL based on the match between the URL and the profile. Since the profile is represented as a binary vector, it is very unlikely that all possible terms are included in it.

Ma et al. [21] suggested a mapping framework that automatically maps a set of known user interests on to a open directory project hierarchy. The paper also described the comparison between personalization categorization system (PCAT) and non personalized categorization system (CAT) and found that PCAT is preferable for short queries. The work described is a form of personalization based on an interest-taxonomy-mapping framework and result categorization. It is embedded on a standard search engine such as Google and displayed the categorized search results on the basis of known user interests. In the reference [31], a system is reported which uses a fuzzy intelligent search agent to satisfy the specific requirements of different users. An example of job hunting agent is described to illustrate fuzzy approach. However, no description is given on how the information is collected from the Web and the application of the fuzzy rules.

3 Architectural framework

Figure 1 depicts the overall component structure of the crawler design. It consists of three phases: input, crawling and feedback.

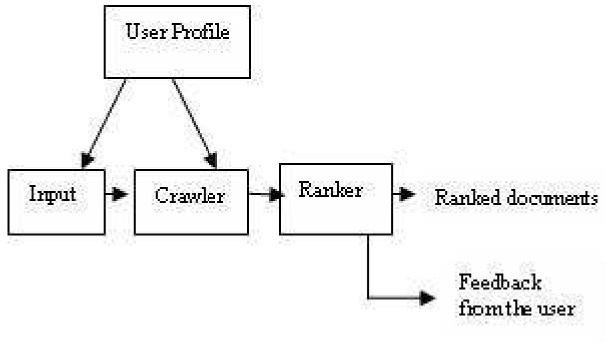

Figure 1: Overall architecture of the personalized crawling system

3.1 Input Phase

The elements of input phase are shown in Figure 2.

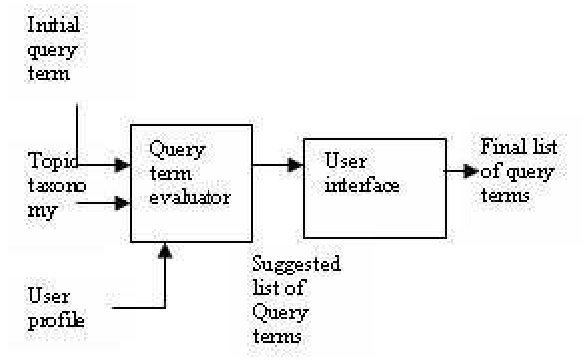

Figure 2: Components of input module

The main functionality of this module is to drive the crawler with appropriate query terms and an example set of URLs. This feature makes the crawler to decide whether a Web page is relevant or not apart from exploiting information available in link structures. This step is considered crucial since the user may not exactly know the keyword describing the topic of search. If these terms are extracted from other sources and suggested, the user may very well be able to guide the crawler to the correct direction. The user has to specify the initial search terms which are stored in the user profile, to query term evaluator. This component integrates topic taxonomy like Yahoo! and user profile, to extract keywords from example set of documents available in the taxonomy. The query term evaluator suggests a list of keywords and URLs of example Web documents to the user. He/she has to select one or more of them which is given as input to the crawler module. While picking the search terms, the user has to provide a weight measure which is the indicator of how important is the presence of that term in the Web document.

3.2 Crawling Phase

The modules represented in Figure 3 are responsible for gathering relevant pages with respect to particular user interest. The following issues are taken care during implementation:

- Decide on next page to download
- Tunneling
- Improving harvest ratio
- Efficient prioritizing of URLs in frontier queue to maximize the number of relevant downloads
- Avoid multiple agents downloading the same page

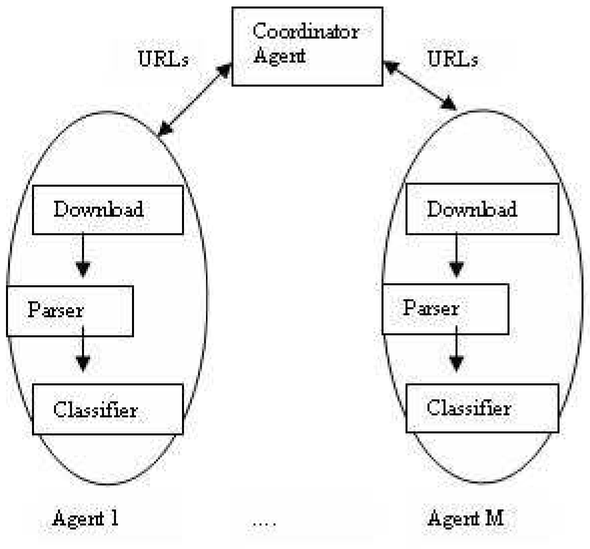

Figure 3: Components of Crawler

The crawler is implemented with multiple agents. Two types of agents are employed: coordinator agent and retrieval agents. The functionality of retrieval agent is to download the Web pages and classify them as relevant or irrelevant. The coordinator agent disseminates the URLs from global frontier queue to different retrieval agents. By using two types of agents, there is a gain of reduced network traffic load and parallelized computation.

The coordinator agent functions as a mediator among retrieval agents and input module. The agent is incorporated with multi-level frontier queue. The URLs with topmost relevance score will be placed in topmost level and the URLs with lower relevance scores are placed in the subsequent levels. This data structure is used to implement tunneling. There are four levels of frontier queue. The URLs received as input are placed along with its relevance score initially in the first level of multi-level frontier queue. Initially, the coordinator agent initiates number of retrieval agents equal to the number of URLs in the first level of queue.

One URL from the queue is placed in each retrieval agent. The agent downloads the Web page, parses it. The parsed results are given to the classifier which is implemented as Naïve-Bayes classifier. The relevance of the document is computed as the combination of page rank [8] and content similarity. The latter component is added to reduce number of spam pages getting into the result. The score is determined as follows:

$$DS(d) = PR_n\ (d) + CS_n(d) \tag{1}$$

where DS is the document score, PR_n is the normalized page rank and CS_n is the normalized content similarity. PR_n and CS_n is normalized to make their sum $\leq$ 1. The weights for normalization are chosen as 0.62 and 0.38 by careful examination of the system with different values. The content similarity between the document d and query vector v is computed as

$$CS = \frac{\sum_{i=1}^{n} Wt_{i,d}\, Wt_{i,v}}{\left(\sqrt{\sum_{i=1}^{n} Wt_{i,d}^2}\right)\left(\sqrt{\sum_{i=1}^{n} Wt_{i,v}^2}\right)} \tag{2}$$

where $Wt_{i,d}$ and $Wt_{i,v}$ are the weights of the terms in the document and query vectors respectively. The weights can be computed in any one of the following forms:

- $W = tf$, where tf is term frequency which specifies the frequency of occurrence of the term in the document
- $W = tf/tf_{\max}$, where $tf_{\max}$ is the maximum term frequency in the document
- $W = \text{IDF} = \log(N/n)$, where IDF is the inverse document frequency which assigns high values for rare words and low values for common words, N is the number of documents in the collection and n is the number of documents containing the query term
- $W = tf\times$ IDF $= tf \times \log(N/n)$, the popular TFIDF measure
- $W = tf\times$ IDF $= tf \times \log((N-n)/n)$, a variation of TFIDF measure

This work has implemented the computation of weights by choosing the popular TFIDF measure.The computed document relevance score is compared with a threshold value which is specified after experimentation and the document is classified as relevant or irrelevant. The links from both relevant and irrelevant documents are extracted and link similarity scores are computed.

$$LS_n = U_n + AT_n \tag{3}$$

where LS_n is the link relevance score, U_n is the relevant measure of URL and AT_n is the similarity computed between anchor text and query vector as in (2). U_n is the conditional probability measured as $P(C \mid R_L)$ where C is the event that current page is relevant satisfying the user's search term with (C) as the associated probability and R_L be the event that document following a link is also relevant. The computed link relevance scores are checked against a given threshold t to determine the probability of adding them into URL frontier. If $LS_n > t$ then percentage of relevance RL_p is computed as:

$$RL_p = \frac{LS_n - t}{t} \times 100 \tag{4}$$

The URLs whose RL_p value in the interval 75%–100%, 50%–75%, 25%–50%, and 0%–25% are added in levels 0,1, 2, and 3 respectively. The Web documents which are determined as relevant are stored in the database along with their document relevance scores. The main purpose of extracting URLs of irrelevant documents is to achieve tunneling i.e. following URLs from off-topic pages so that more coverage is achieved.

3.3 Feedback Phase

The URLs stored in the database are sorted according to their scores and presented to the user by the ranker module. Each result is assigned with a probability of relevance measure by the user. This module is included in the framework solely for the purpose of evaluating the performance of the crawler. During experimentation, users with the knowledge in the particular domain of search are asked to give the feedback in terms of three cases. Those cases are: highly relevant, fairly relevant and irrelevant. The system then assigns a probability score of 0.9, 0.6 and 0.3 to those answers given by the user.

4 Discussion

The proposed system is developed using JADE environment. JADE is one of the multi-agent development environment supporting the implementation of multiple agents. It also integrates communication and coordination mechanisms to allow multiple agents cooperating among themselves to achieve a particular task. Each of the agents dynamically discovers each other and communicates using FIPA ACL [26].

The performance of the proposed system is compared with three baseline crawlers:

1. baseline crawler with relevance scores computed using page rank alone – BC1

2. baseline crawler with relevance scores computed using content similarity alone – BC2

3. baseline crawler without topic suggestion module – BC3

The most important parameter considered for evaluating a personalized crawler is the harvest rate. It is defined as the average relevance of all retrieved pages on a particular topic.

$$\text{Harvest Rate} = \frac{\sum_{i=1}^{N} DS(d)}{N} \quad (5)$$

The performance of the crawlers is plotted in Figure 4

The crawlers are also compared with respect to the relevance of the documents retrieved. This attribute is determined with the help of feedback given by the user. The user has assigned a probability of relevance on all the documents retrieved by different crawlers and average probability is calculated. The evaluation results are tabulated in Table 1 which clearly shows that the proposed crawler design has substantial improvement in retrieving more relevant documents.

$$\text{avgprob} = \frac{\sum_{i=1}^{N} \text{prob of Rel}}{N} \quad (6)$$

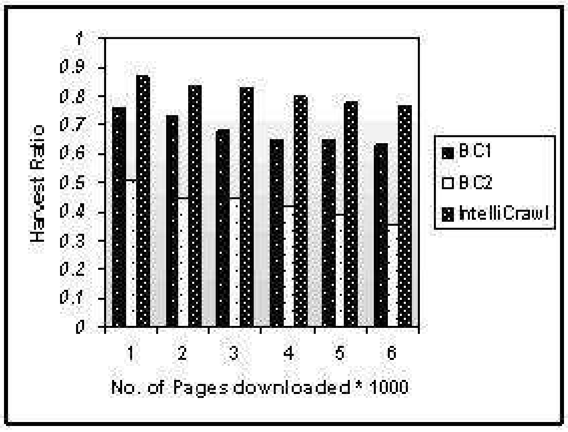

Figure 4: Performance of the crawlers with respect to Harvest Ratio

Table 1: Comparison of average probability of relevance among different crawler implementations

Topics	Tourism	Digital camera	Chinese restaurants in India
BC1	0.71	0.64	0.62
BC2	0.53	0.43	0.45
IntelliCrawl	0.79	0.78	0.70

While comparing the proposed crawler with baseline crawlers, the experimental results showed substantial improvements gained by implementing single level frontier queue as multiple level frontier queues. The coverage is defined as the total number of relevant pages downloaded by the crawler. The experimental results for total of 10000 pages are depicted in Figure 5.

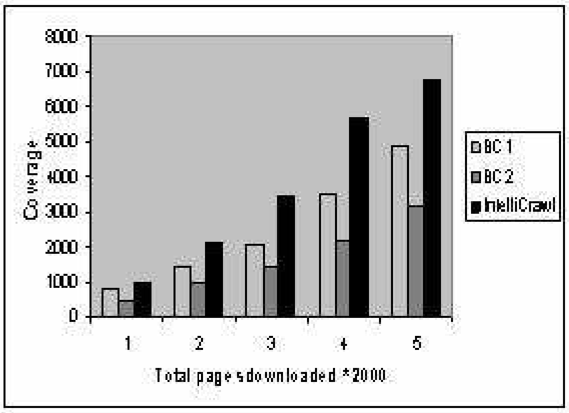

Figure 5: Coverage of crawlers under experimentation

The experimentation is done on different levels of frontier queues. It was found that the relevance scores of the harvested pages increased till the number of levels was 4. Adding one more level to the queue decreases the relevance score as the crawler guided by the low quality pages drifted from the topic of search. The results are represented in Figure 6.

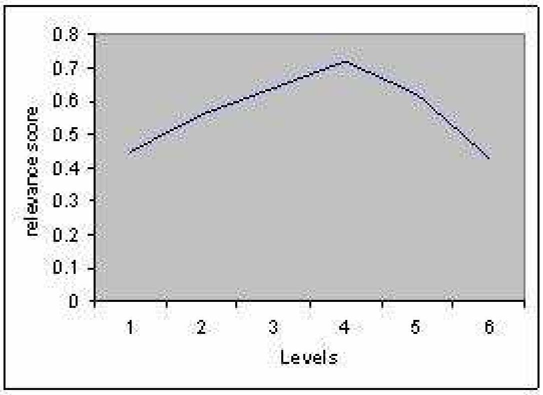

Figure 6: Number of levels vs. average relevance score

Even though the results from the experiment look very promising, it is planned to test its significance with the help of hypothesis testing. The intension of the significance test is to check whether the proposed method yields consistent performance. The annova test is performed on the sample values taken from the experiment and given below:

Let H0 (null hypothesis): There is no significant difference in the calculation of relevance score

H1 : There is significant improvement in the calculation of relevance score

Table 2: Relevance scores computed under different schemes

BC1	BC2	IntelliCrawl
0.73	0.68	0.79
0.74	0.72	0.83
0.76	0.69	0.78
0.71	0.7	0.81

The null hypothesis is successfully rejected and it is concluded that there is significant improvement in the values of the relevance scores computed by intelliCrawl. Another hypothesis testing is applied for substantiating the improvement on performance on using topic suggestion module. One tailed Z-test is used for this purpose. The table of values on relevance scores computed is shown below.

Table 3: Relevance scores of the crawler designs with and without topic suggestion module

S.No.	BC3 without topic suggestion module	*intelliCrawl* with topic suggestion module
1	0.68	0.76
2	0.69	0.79
3	0.60	0.72
4	0.49	0.63
5	0.59	0.70

Let H0 (null hypothesis): No significant effect of topic suggestion module on the result.

H1 : There is significant effect of topic suggestion module on the result.

The null hypothesis is successfully rejected and alternative hypothesis is concluded.

5 Conclusion

Several issues of designing a personalized crawler are discussed in this paper. A new architecture is proposed to locate relevant information with respect to particular user. A novel strategy is proposed to compute personalized relevance score. The input module integrated in the system improves the effectiveness of the retrieval task. Experimental results suggest that the system proposed performs well in terms of harvest rate, and coverage. This work can be extended to apply dynamic data fusion algorithm in the coordinator module allowing the retrieval agents to download duplicate pages in order to reduce the time taken on coordinating them by a central agent.

References

[1] Aggarwal, C. C., Al-Garawi, F., and Yu, P. S. Intelligent crawling on the world wide web with arbitrary predicates. *In Proceedings of the 10th International Conference on WWW, ACM Press*, pages 96–105, 2001.

[2] Akilandeswari, J. and Gopalan, P. A web mining system using reinforcement learning for scalable web search with distributed fault-tolerant multi-agent. *WSEAS Transactions on Computers*, 4:1633–1640, 2005.

[3] Aktas, M. S., Nacar, M. A., and Menzcer, F. Using hyperlinks features to personalizeweb search. *Advances in Web Mining and Web Usage Analysis, LNCS*, pages 104–115, 2006.

[4] Altingövde, I. S. and Özgür Ulusoy. Exploiting inter-class rules for focussed crawling. *IEEE Intelligent Systems*, 19:66–73, 2004.

[5] Bar-Yossef, Z. and Gurevich, M. Random sampling from a search engine's index. *In Proceedings of 15th International conference on WWW*, pages 367–376, 2006.

[6] Barfouroushi, A., Anderson, M., Nezhad, H. M., and Perlis, D. Information retrieval on the world wide web and active logic: A survey and problem definition. *Technical Report*, pages 1–45, 2002.

[7] Bra, P. D., Houben, G.-J., Kornatzky, Y., and Post, R. Information retrieval in distributed hypertexts. *In Proceedings of 4th International Conference on Intelligent Multimedia Information Retrieval Systems and Management (RIAO 94), Center of High International Studies of Documentary Information Retrieval (CID)*, pages 481–491, 1994.

[8] Brin, S. and Page, L. The anatomy of large scale hypertextual web search engine. *In Proceedings of 7th World Wide Web Conference, Elsevier Science*, pages 107–117, 1998.

[9] Cercone, N., Hou, L., Keselj, V., An, A., Neruedomkul, K., and Xu, X. From computational intelligence to web intelligence. *IEEE Computers*, 35(11):72–76, 2002.

[10] Chakrabarti, S., den Berg, M. V., and Dom, B. Focused crawling: A new approach to topic-specific web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[11] Diligenti, M., Coetzee, F., Lawrence, S., Giles, C., and Gori, M. Focused crawling using context graphs. *In Proceedings of 26th International Conference on VLDB*, pages 527–534, 2000.

[12] Ehrig, M. and Maedche, A. Ontology-focused crawling of web documents. *In Proceedings of the ACM symposium on Applied Computing*, pages 1174–1178, 2003.

[13] Finin, T., Fritzson, R., McKay, D., and McEntire, R. Kqml as an agent communication language. *n Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM'94), ACM Press*, pages 456–463, 1994.

[14] Genesereth, M. R. and Fikes, R. E. Knowledge interchange format, version 3.0. *Technical Report 92-1, Stanford University, Computer Science Department*, 1992.

[15] Hersovici, M., Jacovi, M., Maarek, Y. S., Pelleg, D., Shtalhaim, M., and Ur, S. The shark-search algorithm – an application: Tailored web site mapping. *Computer Networks and ISDN Systems*, 30(1–7):317–326, 1998.

[16] Junghoo Cho, H. G. M. Parallel crawlers. *In Proceedings of 11th International Conference on WWW*, pages 124–135, 2002.

[17] Junghoo Cho, H. G.-M. and Page, L. Efficient crawling through url ordering. *Computer Networks and ISDN Systems*, 30(1–7):161–172, 1998.

[18] Kim, K.-J. and Cho, S.-B. Personalized mining of web documents using link structures and fuzzy concept networks. *Applied Soft Computing*, pages 398–410, 2007.

[19] Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. The web and social networks. *IEEE Computers*, 35(11):32–36, 2002.

[20] Labrou, Y., Finin, T., and Peng, Y. The current landscape of agent communication languages. *Intelligent Systems and their Applications*, 14:45–52, 1999.

[21] Ma, Z., Pant, G., Olivia, and Sheng, R. L. Interest based personalized search. *ACM Transactions on Information Systems*, 25(1):1–38, 2007.

[22] McCallum, A., Nigam, K., Rennie, J., and Seymore, K. Building domain-specific search engines with machine learning techniques. *In Proceedings of AAAI Spring Symposium on Intelligent Agents in Cyberspace, AAAI Press*, pages 28—39, 1998.

[23] Menczer, F., Pant, G., and Srinivasan, P. Topical web crawlers: Evaluating adaptive algorithms. *ACM Transactions on Internet Technology*, 4:378–419, 2004.

[24] Mukherjea, S. Wtms: A system for collecting and analyzing topic-specific web information. *The International Journal of Computer and Telecommunications Networking*, 33(1–6):457–471, 2000.

[25] Najork, M. and Wiener, J. L. Breadth-first crawling yields high-quality pages. *In Proceedings of*

10th International Conference on WWW, pages 114–118, 2001.

[26] Nikraz, M., Caire, G., and Bahri, P. A. A methodology for the analysis and design of multi-agent systems using jade. *International Journal of Computer Systems Science and Engineering, Special issue on Software Engineering for Multi-Agent Systems*, 2006.

[27] Olivia, C., Change, C., Enguix, F., and Ghose, A. Case-based bdi agents: An effective approach for intelligent search on the web. *In Proceedings of AAAI-99, Spring Symposium on Intelligent Agents in Cyberspace, Stanford University, USA*, 1999.

[28] Raghavan, S. and Molina, H. G. Crawling the hidden web. *In Proceedings of 27th International conference on VLDB*, pages 129–138, 2001.

[29] Sizov, S., Graupmann, J., and Theobald, M. From focused crawling to expert information: An application framework for web exploration and portal generation. *In Proceedings of 29th International Conference on Very Large Databases (VLDB 2003), Morgan Kaufmann*, pages 1105–1108, 2000.

[30] Tsay, J., Shih, C.-Y., and Bo-LiangWu. Autocrawler: An integrated system for automatic topical crawler. *In Proceedings of 4th Annual ACIS International Conference on Computer and Information Science*, pages 462–467, 2005.

[31] Wu, J., Qiu, M., Huang, H.-C., and Yang, L. T. Intelligent search agent for internet computing with fuzzy approach. *In Proceedings of International Conference on Computational Science and Engineering*, pages 181–188, 2008.