Journal of Engineering Sciences

Vol 10, Issue 9, Sept /2019
ISSN NO: 0377-9254

Capturing User Intention for Personalized Website's for Improving Retrieval Effectiveness

[1]S. Somasekhar, [2]H. Ateeq Ahmed, [3]G. Prathibha Priyadarshini
[123]Assistant Professor
DEPARTMENT OF CSE
Dr. K.V. SUBBA REDDY INSTITUTE OF TECHNOLOGY,KURNOOL

ABSTRACT: Current web search engines are built to serve all users, independent of the special needs of any individual user. Personalization of web search is to carry out retrieval for each user incorporating his/her interests. We propose a novel technique to learn user profiles from users' search histories and use them to improve retrieval effectiveness in web search. A user profile and a general profile are learned from the user's search history and a category hierarchy respectively. These two profiles are combined to map a user query into a set of categories, which represent the user's search intention and serve as a context to disambiguate the words in the user's query. Web search is conducted based on both the user query and the set of categories. Several profile learning and category mapping algorithms and a fusion algorithm are provided and evaluated. Experimental results indicate that our technique to personalize web search is both effective and efficient.

Keywords: Personalization, Search Engine, Category Hierarchy, Information Filtering, Retrieval Effectiveness

1. INTRODUCTION

As the amount of information on the Web increases rapidly, it creates many new challenges for Web search. When the same query is submitted by different users, a typical search engine returns the same result, regardless of who submitted the query. This may not be suitable for users with different information needs. For example, for the query "apple", some users may be interested in documents dealing with "apple" as "fruit", while other users may want documents related to Apple computers. One way to disambiguate the words in a query is to associate a small set of categories with the query. For example, if the category "cooking" or the category "fruit" is associated with the query "apple", then the user's intention becomes clear. Current search engines such as *Google* or *Yahoo!* have hierarchies of categories to help users to specify their intentions. The use of hierarchical categories such as the Library of Congress Classification is also common among librarians.

A user may associate one or more categories to his/her query manually. For example, a user may first browse a hierarchy of categories and select one or more categories in the hierarchy before submitting his/her query. By utilizing the selected categories, a search engine is likely to return documents that are more suitable to the user. Unfortunately, a category hierarchy shown to a user is usually very large, and as a result, an ordinary user may have difficulty in finding the proper paths leading to the suitable categories. Furthermore, users are often too impatient to identify the proper categories before submitting his/her queries. An alternative to browsing is to obtain a set of categories for a user query directly by a search engine. However, categories returned from a typical search engine are still independent of a particular user and many of the returned categories do not reflect the intention of the searcher. To solve these problems, we propose a two-step strategy to improve retrieval effectiveness. In the first step, the system automatically deduces, for each user, a small set of categories for each query submitted by the user, based on his/her search history. In the second step, the system uses the set of categories to augment the query to conduct the web search. Specifically, we provide a strategy to (1) model and gather the user's search history, (2) construct a user profile based on the search history and construct a general profile based on the *ODP* (Open Directory Project[1]) category hierarchy, (3) deduce appropriate categories for each user query based on the user's profile and the general profile, (4) conduct web search using these categories, and (5) perform numerous experiments to demonstrate that our strategy of personalized web search is both effective and efficient.

The categories obtained in the first step of our proposed method are likely to be related to the user's interest and, therefore, can provide a proper context for the user query. Consider the situation where a mobile user wants to retrieve documents using his/her PDA. Since the bandwidth is limited and the display is small, it may not be practical to transmit a large

number of documents for the user to choose the relevant ones. Suppose that it is possible to show the retrieved documents on one screen. If these documents are not relevant to the user, there is no easy way for the user to direct the search engine to retrieve relevant documents. With the use of our proposed technique, a small number of categories with respect to the user's query are shown. If none of the categories is desired, the next set of categories is provided. This is continued until the user clicks on the desired categories, usually one, to express his/her intention. As will be demonstrated by our experiments, the user usually finds the categories of interest among the first 3 categories obtained by our system. Since 3 categories can easily fit into one screen, it is likely that effective retrieval can be achieved with minimal interaction with the user. Thus, our proposed technique can be used to personalize web search.

The contribution of this paper is as follows:

(1) We provide methods to deduce a set of related categories for each user query based on the retrieval history of the user. The set of categories can be deduced using the user's profile only, or using the general profile only or using both profiles. We make the following comparisons. We show that the accuracy of combining both profiles is better than those using a single profile and the accuracy of using the user profile only is better than that using the general profile only, provided that there is sufficient history data.
   (a) The accuracy of combining the user profile and the general profile versus that of using the user profile only.
   (b) The accuracy of combining the user profile and the general profile versus that of using the general profile only.
   (c) The accuracy of using the user profile only versus that of using the general profile only.

(2) We propose two processes, one semi-automatic and another completely automatic modes, to personalize web search based on both the query and its context (the set of related categories). We show that both personalization modes can improve retrieval effectiveness.

2. PROBLEM

The problem is to personalize web search for improving retrieval effectiveness. Our strategy includes two steps. The first step is to map a user query to a set of categories, which represent the user's search intension and serves as a context for the query. The second step is to utilize both the query and its context to retrieve web pages. In order to accomplish the first step, a user profile and a general profile are constructed. We propose a tree model in Section 2.1 to represent a user's search history and describe how a user's search history can be collected without his/her direct involvement. In Section 2.2, a brief description of a user profile is given. A matrix representation of the user history and the user profile is described in Section 2.3. General knowledge from a category hierarchy is extracted for the purpose of constructing the general profile. This is given in Section 2.4. Section 2.5 sketches the deduction of the appropriate categories based on a user query and the two profiles. The last Section sketches the second step.

2.1 User Search History

A search engine may track and record a user's search history in order to learn the user's long-term interests. We consider using the following information items to represent a user's search history: queries, relevant documents and related categories. One **search record** is generated for each user search session. A tree model of search records is shown in Figure 1. In this model, nodes are information items and edges are relationships between nodes. The root of a search record is a query. Each query has one or more related categories. Associated with each category is a set of documents, each of which is both relevant to the query and related to the category. Based on our experiments with users, for almost all queries, each query is related to only one or two categories.

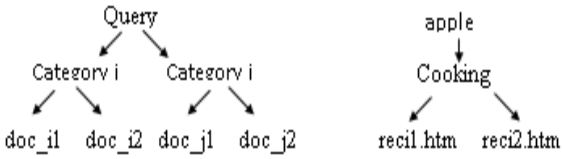

Figure 1: Model and example of a search record

In practice, a search engine may be able to acquire the type of user's search records described above, without direct involvement by the user. Some possible scenarios are as follows:

(1) A document retrieved by a search engine can be assumed to be relevant to the user with respect to a user query if some of the following user behaviors are observed: the user clicks it and there is a reasonable duration before the next click; the user saves/prints it.

(2) A user utilizing some of the popular search engines may first select a category before submitting a query. In this way, a category related to the user query is identified. Furthermore, some search engines such as *Google* have pre-classified some results into categories; some other search

engines such as *Northern Light* cluster all results into categories. When such documents are observed to be relevant (see scenario 1 above), the user query, its related categories and its relevant documents are identified.

Based on (1) and (2), a set of search records representing a user's search history can be obtained. As an example, consider the following session with the *Northern Light* search engine. A user who is interested in cooking submits a query "apple" to the search engine, and it returns the top 10 documents and 12 categories. The user clicks the 8th category "Food & cooking" and the search engine shows all documents that have been clustered into this category. Then, the user clicks two documents about cooking apples. When this search session is finished, a search record as shown in Figure 1 can be generated and saved for the user.

2.2 User Profile

User profiles are used to represent users' interests and to infer their intentions for new queries. In this paper, a **user profile** consists of a set of categories and for each category, a set of terms (keywords) with weights. Each category represents a user interest in that category. The weight of a term in a category reflects the significance of the term in representing the user's interest in that category. For example, if the term "apple" has a high weight in the category "cooking", then the occurrence of the word "apple" in a future query of the user has a tendency to indicate that the category "cooking" is of interest. A user's profile will be learned automatically from the user's search history.

2.3 Matrix Representation of User Search History and User Profile

We use matrices to represent user search histories and user profiles. Figure 2 shows an example of the matrix representations of a search history and a profile for a particular user, who is interested in the categories "COOKING" and "SOCCER". This user's search history is represented by two matrices *DT* (Figure 2(a)) and *DC* (Figure 2(b)). *DT* is a document-term matrix, which is constructed from the user queries and the relevant documents. (In the following discussion, we use "documents" to denote both queries and relevant documents in the matrices *DT* and *DC*). *DC* is a document-category matrix, which is constructed from the relationships between the categories and the documents. A user profile is represented by a category-term matrix *M* (Figure 2(c)). In this example, D1, D2, … are documents; lowercase words such as "football", "apple",… are terms; uppercase words such as "SOCCER", "COOKING", … are categories.

We now describe the construction of the matrices *DT* and *DC* based on the user's search records.

Doc\Term	apple	recipe	pudding	football	soccer	fifa
D1	1	0	0	0	0	0
D2	0.58	0.58	0.58	0	0	0
D3	0	0	0	1	0	0

DT

Doc\Category	COOKING	SOCCER
D1	1	0
D2	1	0
D3	0	1

DC

Cate\Term	apple	recipe	pudding	football	soccer	fifa
COOKING	1	0.37	0.37	0	0	0
SOCCER	0	0	0	1	0.37	0.37

M

(c) Category-Term matrix *M* represents a user profile

2.4 A Category Hierarchy

In addition to the matrices *DT* , *DC* and *M* as described above, we also utilize some general knowledge which is applicable to all users. The reason for using the additional information is that the knowledge acquired from a user is often limited and may not be sufficient to determine the user's intention when a new user query is encountered. For example, a new query may contain terms that have never been used by the user before, nor appeared in any of his/her previous retrieved relevant documents. The general knowledge that our system utilizes is extracted from *ODP*. Specifically, we use the first three levels of *ODP*. The categories in the first two levels (15 first level categories and 604 second level categories) are used to represent the set of all categories. The terms appearing in these three levels of categories are used to represent the categories in the first two levels. From the category hierarchy, we learn a general profile, using a process similar to that for learning the user profile. Let the three corresponding matrices related to the general knowledge be denoted by *DTg* , *DCg* and *Mg* (general profile).

To construct document-term matrix *DTg* , we generate two documents for each category in the first two levels. One document consists of all terms in the text descriptions of its subcategories. The other document consists of terms in the category's own text description. For example, in Figure 3, "Artificial intelligence" is a second level category, and has subcategories "Data mining", "Genetic algorithms", etc. Thus, for this category ("Artificial intelligence"),

one document with the terms "data", "mining", "genetic" and "algorithms" and another document with the terms "artificial" and "intelligence" are generated, respectively. Note that both of the above two documents for each category are needed. On one hand, all of the terms in the text descriptions of the category and its subcategories are useful for describing the category. On the other hand, the terms in the text description of the category are likely to be more important than the terms in the text descriptions of the subcategories for describing the category. By using the above two documents, the difference in importance among the terms can be captured. As a result, a pair of rows is created in *DTg* for each category.

1: Computers
2: Algorithms
… 2: Artificial intelligence
3: Data mining
3: Genetic algorithms
… 2: Internet

Figure 3: An example of the category hierarchy

2.5 Inference of User Search Intention

In our environment, the first step of personalized search is accomplished by mapping a user query to a set of categories, which reflects the user's intention and serves as a context for the query, based on the user profile and the general profile. The mapping is carried out as follows. First, the similarities between a user query and the categories representing the user's interests are computed. Next, the categories are ranked in descending order of similarities. Finally, the top three categories together with a button indicating the next three categories are shown to the user. If the user clicks on one or more of these top three categories, then the user's intention is explicitly shown to the system. If the user's interest is not among the top three categories, then the button can be clicked to show the next three categories.

2.6 Improving Retrieval Effectiveness Using Categories

Our goal is to improve retrieval effectiveness. To accomplish the second step of personalized search, we propose the following modes of retrieval based on whether and how categories are used. The second and the third modes are personalized search.

(1) If the system does not utilize categories, queries are processed against the whole database of documents. In fact, this is not a mode of personalized search and will be considered as the *baseline* mode in our experiment.
(2) As discussed before, our system determines the three categories which are most likely to match the interests of the user with the given user query. From these three categories, the user can either pick the ones which are most suitable or he/she can decide to see the next 3 categories. The process continues until the desired categories are chosen by the user. The user usually finds the desired categories within the first three categories presented by the system. Let us call this the *semi-automatic* mode.
(3) In the *automatic* mode, the system automatically picks the top category or the top 2 categories or the top 3 categories without consulting the user. Thus, the two-step personalization of web search can be accomplished automatically, without the involvement of users as shown in the second mode.

3. ALGORITHMS TO LEARN THE PROFILE

Learning a user profile (matrix M) from the user's search history (matrices DT and DC) and mapping user queries to categories can be viewed as a specific multi-class text categorization task. In sections 3.1-3.3, we describe four algorithms to learn a user profile: bRocchio, LLSF, pLLSF and kNN. The last three algorithms have been shown to be among the top-performance text categorization methods in [Yang99].

3.1 Two LLSF-based Algorithms

Given the m-by-n document-term matrix DT and the m-by-p document-category matrix DC, the Linear Least Squares Fit (LLSF) method [Yang94] computes a p-by-n category-term matrix M such that $DT * M^T$ approximates DC with the least sum of square errors, where M^T is the transpose of M. A common technique for solving this problem is to employ the Singular Value Decomposition (SVD) [Golub96]. DT is decomposed into the product of three matrices $U*\Sigma*V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix. After such decomposition, it is rather straightforward to compute: $M = DC^T * U * \sum^+ * V^T$, where Σ^+ is the inverse of Σ.

We also evaluate another variant called "pseudo-LLSF" (pLLSF), in which the dimensions of DT are reduced. Matrices Σ, U and V are replaced by Σ_k, U_k and V_k respectively, where Σ_k contains the highest k entries in the diagonal matrix Σ, U_k and V_k are obtained by retaining the first k columns of U and V respectively. Essentially, the original space is replaced by a k dimensional space. After the replacements, M is computed from these modified matrices using the same formula, i.e.,

$M = DC^T * U_k * \sum_k^+ * V_k^T$. The basic idea is that the noise in the original document-term matrix DT is removed by the dimension reduction technique. This technique is also the key of the Latent Semantic Indexing method (LSI) [Deer90], which has been used successfully in various applications in IR [Deer90, Foltz92, Dolin98]. In practice, it is not easy to give a good value of k. Thus, we choose a k such that the ratio of the smallest retained singular value over the largest singular value is greater than a threshold θ, which is set to be 0.25 in this paper.

3.2 Rocchio-based Algorithm

Rocchio is originally a relevance feedback method [Rocc71]. We use a simple version of Rocchio adopted in text categorization:

$$M(i, j) = \frac{1}{N_i} \sum_{k=1}^{m} DT(k, j) * DC(k, i)$$

where M is the matrix representing the user profile, N_i is the number of documents that are related to the i-th category, m is the number of documents in DT, $DT(k, j)$ is the weight of the j-th term in the k-th document, $DC(k, i)$ is a binary value denoting whether the k-th document is related to the i-th category. Clearly, $M(i, j)$ is the average weight of the j-th term in all documents that are related to the i-th category and documents that are not related to the category are not contributing to $M(i, j)$. We call the batch-based Rocchio method bRocchio.

3.3 kNN

The k-Nearest Neighbor (kNN) does not compute a user profile. Instead, it computes the similarity between a user query and each category directly from DT and DC.

3.4 Adaptive Learning

The algorithms introduced above are all based on batch learning, in which the user profile is learned from the user's previous search records. Batch learning can be inefficient when the amount of accumulated search records is large. An adaptive method can be more efficient, as the user profile is modified by the new search records. LLSF-based algorithms are not suitable for adaptive learning as re-computation of the user profile M is expensive. kNN requires storing DT and DC, which is space inefficient. Furthermore, the computation of similarities using kNN can be inefficient for large amount of search records. Rocchio is efficient in both computation and storage, and is adaptive. The following formula is used:

$$M(i, j)^t = \frac{N_i^{t-1}}{N_i^t} M(i, j)^{t-1} + \frac{1}{N_i^t} \sum_k DT(k, j) * DC(k, i)$$

where M^t is the modified user profile at time t; N_i^t is the number of documents (documents are related to the i-th category) that have been accumulated from time zero to time t; the second term on right hand side of the equation is the sum of the weights of the j-th term in the documents that are related to the i-th category and obtained between time $t-1$ and time t divided by N_i^t. For example, suppose at time $t-1$, the value of $M(i, j)^{t-1}$ is 0.5, N_i^{t-1} is 10; between time $t-1$ and t, we collect a number of new documents, among which there are 5 documents that are related to the i-th category; and the sum of the weights of the j-th term in these 5 document is 1. Then, N_i^t is 10+5=15 and $M(i, j)^t = \frac{10}{15} * 0.5 + \frac{1}{15} * 1 = 0.4$. We call this adaptive–based Rocchio method aRocchio.

4. IMPROVING RETRIEVAL EFFECTIVENESS

Our system maps each user query to a set of categories, and returns the top three categories. In this section, we provide methods to improve retrieval effectiveness using categories as a context of the user query. Three modes of retrieval have been briefly introduced in Section 2.6. In the three modes of process, the user query is submitted to the search engine (in this case Google Web Directory) multiple times. In the first mode, it is submitted to the search engine without specifying any category. Let the list of documents retrieved be *DOC-WO-C* (documents retrieved without specifying categories). Let its cardinality be *MO*. In the second and third modes, the query is submitted by specifying a set of categories which is obtained either semi-automatically or completely automatically. Let the list of documents retrieved by specifying the top i category be *DOC-W-Ci*. Let its cardinality be *MWi*. The reasons for the multiple submissions of the query are as follows:

- *MO* is usually larger than *MWi*. As a consequence, a fair comparison between retrieval using the specified categories and that of not specifying any category is not possible. Our solution is as follows. We will merge the retrieved lists of documents *DOC-WO-C* and *DOC-W-Ci* in such a way that the resulting set has exactly the same cardinality as *DOC-WO-C*.

- The list of retrieved documents *DOC-WO-C* usually contains reasonable number of relevant documents, although there are quite a few irrelevant documents in it. This information should be utilized in determining the final set of retrieved documents. Furthermore, the automatically determined categories have certain probabilities of being wrong. Thus, it will not be wise to utilize that information solely by itself.

4.1 Algorithm

Our algorithm to merge multiple lists of retrieved documents, *DOC-WO-C* and *DOC-W-Ci*, is by modifying a voting merging scheme [MoAs02]. The original merging scheme is as follows: Each retrieved list has the same number of documents, say *N*. The i-th ranked document in a list gets (*N* - i +1) votes. Thus, the first document in a list gets *N* votes. If a document appears in multiple lists, it gets the sum of the votes of that document appearing in the lists. In other words, if a document appears in multiple lists, it usually gets more votes than a document appearing in a single list. Documents in the merged list are ranked in descending order of votes. It has been shown in [MoAs02] that this way of merging is effective.

Our modification of the above scheme is a weighted voting merging algorithm:

(1) Let *MM* be the number of documents in the longest list. In our case, the longest list is DOC-WO-C and *MM=MO*.

(2) Each list, say the j-th list, has a weight wj associated with it. The number of votes assigned to the i-th ranked document in the j-th list is *Wj* * (*MM* - i + 1). The weight *Wj* is dependent on the rank of the category, the similarity of the category with respect to the query and the number of documents in the list.

(3) If the j-th list of retrieved document is obtained from a specified category, C, then *Wj* is given by

Wj = *rank-C* * square-root(*sim-C*) * *num-C*, where

rank-C = *1*, if the rank of C with respect to the query is 1;

0.5, if the rank is 2;

0.25, if the rank is 3.

sim-C is the cosine similarity of the category with respect to the query

num-C is the number of retrieved documents in the list (either *MWi* or *MO*).

rank-C is 1 and *sim-C* is 1 in the semi-automatic mode in which the category is selected by the user. If the list of documents is obtained by not specifying any category, then *rank-C* is 0.5; *sim-C* is 0.1, which is approximately the average similarity of the top rank categories for the queries.

The following scenarios explain the motivation that weights are assigned as introduced above:

- Suppose the top rank category has a similarity much higher than 0.1, then assuming that each list has the same number of documents, the weight associated with *DOC-W-C1* is much higher than that associated with *DOC-WO-C*. This is consistent to the notion that a category, if it obtains high similarity, receives high confidence. This implies that documents in *DOC-W-C1* gets higher votes than those in *DOC-WO-C*. Conversely, if a top-ranked category receives low similarity, then the documents in *DOC-W-C1* get low votes. Consider the extreme situation where none of the query terms appears in either the user profile or the general profile. In that case, the similarities between the query and all the categories will be all zeros. This means that the weight *Wj* = 0. As a consequence, only the list of documents *DOC-WO-C* is retrieved.
- Documents retrieved using higher ranked categories get more votes than those retrieved using lower ranked categories. This explains the relative values assigned to rank-C as well as sim-C.
- If a query is submitted to a wrong category, then the number of documents retrieved is usually very few, which is an indication that the confidence of using the category is low.

5. Experiments

5.1 Data Sets

Statistics	User 1	User 2	User 3	User 4	User 5	User 6	User 7
# of relevant categories	10	8	8	8	10	8	8
# of search records (queries)	37	58	81	26	30	29	29
avg # of related search records to one category	3.7	6.3	7.6	3.25	3.0	3.63	3.2

In our experiments, seven data sets collected from seven different users in two phases. In the first phase, each user submitted a number of queries to a search engine which, in this case, is Google. For each query, the user identified the set of related categories and a list of relevant documents, as well as provided a statement, which describes the semantics of the query (similar to the "Description" part of a "Topic" in TREC[Voor02]**).** The query (not containing the statement), the set of related categories and the list of relevant documents comprise a search record. Figure 4 shows an example taken from our data sets. "bulls" is the query; "online information concerned with the basketball team Chicago Bulls" is the statement; "Sport->Basketball" is the related category; and the four files are the relevant documents. Each data set consists of a set of search records. In the second

phase, each query is submitted to the Google Web Directory in the way. For each submission, at most top 10 documents are returned. The relevance of each returned documents is judged as either relevant or irrelevant.

Figure 4: An example of a search record

Table 1 gives the statistics of the data sets. For example, user 1 has 10 interest categories, and 37 search records with 37 queries and 236 relevant documents. As mentioned in Section 2.4, we generate a set of documents in the construction of the general profile, using the text descriptions of the categories in the first 3 levels of *ODP*. There are 619 categories in the first two levels of the hierarchy.

To evaluate our approach, we use the 10-fold cross-validation strategy [Mitch97]. For each data set, we randomly divide the search records into 10 subsets, each having approximately the same number of search records. We repeat experiments 10 times, each time using a different subset as the test set and the remaining 9 subsets as the training set. As described in Section 2.3, we construct two matrices from the search records in the training set and we call them DT_{train} and DC_{train}. Similarly, two matrices DT_{test} and DC_{test} are constructed from the test set. After the user profile M is learned from DT_{train} and DC_{train}, the set of categories is ranked with respect to each query in DT_{test} and the result is checked against DC_{test} to compute the accuracy. The average accuracy across all 10 runs is computed. This is a measurement of performance of mapping queries to categories. In addition, each query in DT_{test} with the set of ranked categories is used to conduct the three modes of retrieval. A standard effectiveness measure will be used.

5. CONCLUSION

We described a strategy for personalization of web search: (1) a user's search history can be collected without direct user involvement; (2) the user's profile can be constructed automatically from the user's search history; (3) the user's profile is augmented by a general profile which is extracted automatically from a common category hierarchy; (4) the categories that are likely to be of interest to the user are deduced based on his/her query and the two profiles; and (5) these categories are used as a context of the query to improve retrieval effectiveness of web search.

For the construction of the profiles, four batch learning algorithms (pLLSF, LLSF, kNN and bRocchio) and an adaptive algorithm (aRocchio) are evaluated. Experimental results indicate that the accuracy of using both profiles is consistently better than those using the user profile alone and using the general profile alone. The simple adaptive algorithm aRocchio is also shown to be effective and efficient. For the web search, the modified voting merging algorithm is used to merge retrieval results. Both semi-automatic and automatic modes of utilizing categories determined by our system are shown to improve retrieval effectiveness by 25.6% and around 12% respectively.

REFERENCES

[1] James Allan. Incremental relevance feedback for information filtering. Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, p.270-278, New York, 1996, ACM Press.

[2] Marko Balabanovic and Yoav Shoham. Learning information retrieval agents: Experiments with automated Web browsing. In On-line Working Notes of the AAAI Spring Symposium Series on Information Gathering from Distributed, Heterogeneous Environments, 1995, p.13-18, Stanford University.

[3] Kurt Bollacker, Steve Lawrence, and C. Lee Giles. A system for automatic personalized tracking of scientific literature on the web. In Proceedings of the 4th ACM Conference on Digital Libraries, p.105-113, New York, 1999. ACM Press.

[4] Jay Budzik and Kristian Hammond. Watson: Anticipating and contextualizing information needs. In Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science, 1999, Information Today, Inc.

[Ceti00] Ugur Cetintemel, Michael J. Franklin, and C. Lee Giles. Self-Adaptive User Profiles for Large-Scale Data Delivery. Proceedings of 16th the International Conference on Data Engineering, 2000, p.622-633, IEEE Computer Society.

[5] Liren Chen and Katia Sycara. WebMate: A Personal Agent for Browsing and Searching. Proceedings of the 2nd International Conference on Autonomous Agents and Multi Agent Systems, 1998, p.132-139.

[6]Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman. Indexing by latent semantic analysis. Journal of the American Society for Information Science (JASIS), 41(6), p.391-407, 1990.

[7] Ron Dolin, Divyakant Agrawal, Amr El Abbadi and J. Pearlman. Using Automated Classification for Summarizating and Selecting Heterogeneous Information Sources. D-Lib Magazine, 1998.
[8] Peter W. Foltz and Susan T. Dumais. Personalized information delivery: An analysis of information filtering methods. Communications of the ACM, 35(12), p.51-60, 1992.
[9] William B. Frakes and Ricardo Baeza-Yates. Information Retrieval: Data Structures and Algorithms. Prentice Hall, 1992.
[10] Susan Gauch, Guijun Wang, and Mario Gomez. ProFusion: Intelligent Fusion from Multiple, Distributed Search Engines. Journal of Universal Computer Science, 2(9), 1996