Search Query Ranking Using Online User Profile ART1 Classifier and Genetic Algorithm

Mahdi Bazarganigilani

Charles Sturt University, Australia
Mahdi62b@yahoo.com

Abstract

Today, getting an exact result from the huge raw data on the internet is not an easy task. Utilizing search engines is a good way, but still there is a need for a tool for personalization and having the adaptive capability according to the users' interests. In this paper, we propose a new algorithm which uses the positive and negative feedback from the user for filtering the information. We will use ART1 classifier to generate the dynamic profile of the users and utilize the genetic algorithm to make the most suitable query to give a better result according to user's criteria search.

Keywords: *Information Filtering, ART1 classifier, Dynamic User Profile, Genetic Algorithm*

1. Introduction

Finding the relevant data from huge information on the internet is a difficult task. Users use different ways like surfing the web or using the search engines, however, the search engines do not cover all the internet, and therefore, there should be a way for finding the information according to the similarity to the users' interests, retrieving the relevant information from huge unstructured and dynamic data stored according to the user profile, called Information Filtering [1]. Agents are the components which have some level of autonomy and adaptation and intelligent behaviors [2]. In our proposed method we combine these two techniques for filtering the huge information on the web and delivering the relevant information to the user according to his/her profile. Our method uses the feedbacks of the users to make a dynamic profile according to positive or negative relevancies of the pages they traverse.

Finally, we make such a profile to produce the most proper query. In our previous work [3], we used a meta search to determine the fitness of each testing query. It is obviously not an efficient method. In our paper, we use an optimized semantic similarity algorithm to determine the similarity between the query and the user's profile.

2. User Profile Modeling

In our method every user has his/her own profile. Every profile consists in three sections: the first section is the information the user is interested in, the second one contains the information which doesn't satisfy the user, and in the third section we have a long descriptor which reflects the interesting rate of the long term of users' interests and represents the interest of the profile. In this model, every descriptor has its own weight representing the intensity of that descriptor. Since the users' interests may change in time, we use a long term descriptor to represent the long term changing interests. In Figure 1, we have shown the structure of our profile. Every profile has different categories and every category has three

descriptors. Every descriptor has the vector of words representing the occurrence of the words: k_i represent the $i'th$ word in the pattern and w_i represents the occurrence of it in the document to feed to our classifier. w_p and w_n represent the total weight of the positive and negative categories, respectively, according to the learning rate of the algorithm and w_{lt} represents the long term weight of descriptor [4].

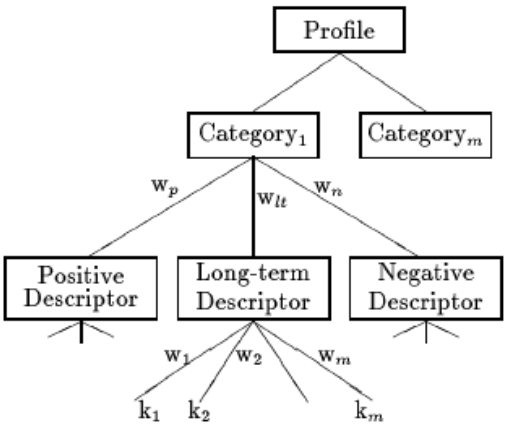

Figure 1. 3-Descriptor User Profile

3. Adaptive Response Theory Neural networks (ART1)

ART1 is a network which is classified into unsupervised learning networks therefore it is called self-organizing network. As shown in the Figure 2, this network consists of two layers, comparison layer (F1) and recognition layer (F2), with the inputs in binary format. Every node in F1 represents a cluster of the technologic property. Every two layers are connected strongly and there are two kinds of connections: one is up-down and the reverse down-up. ART1 also has three other modules: Gain1, Gain2, and Reset, in which it performs the actions for comparison and classification. The algorithm tries to reconcile the input vector with other clusters according to the similarity degree, which is gained according to a parameter named vigilance parameter. If the algorithm can not find any cluster, it creates a new one.

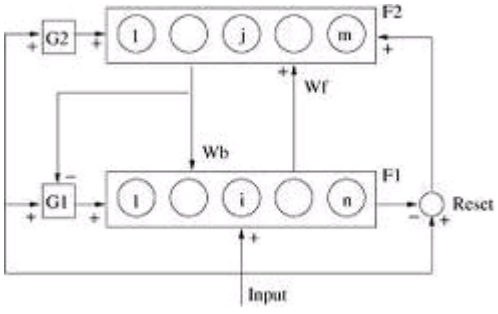

Figure 2. ART1 Neural Network

In this paper we use ART because of the following reasons [5]:

- Binary Input Vector: this network can process input vectors in binary format
- Consistency and Scalability: this network has a good capability to save the previous learning patterns and also to adopt new patterns as a new cluster.
- Unsupervised Learning: in this learning there is no feedback from the environment and the network determines the clusters according the input vectors.
- Fast Learning: in this network a pattern fit in the most similar cluster otherwise a new cluster created.

Another parameter to mention is the Vigilance parameter. In our implementations we set the Vigilance Parameter as high since we need the most accurate classification due to the long input vectors.

4. Information Filtering

Firstly we give an opportunity to the user to search the internet. According to its first query, which denotes to the best result, the first one would be feeded to our pre-process algorithm. The pattern for ART1 classifier consists of 150 words, mostly used in the document. Certainly, the pre-process method eliminates the trivial words and also HTML tags. The pattern is sorted according to the number of occurrences of the keywords. The consequent documents would be listed to the user to select them if he/she is interested to them or not. The input document would be feeded to our pre-process algorithm and the output is keyword according to our pattern, the input vectors $K = \{k_1, k_2, ..., k_m\}$ would be feeded to ART1 classifier. Every input vector would be in interesting subset or its opposition. Accordingly, we compute the weights of descriptors and ART1 gives us the categories. For updating the weights, we consider two methods of learning: Explicit learning and Implicit learning.

5. Explicit and Implicit Learning

Since the profile changes according to the user's interests, we have two types of learnings: Explicit learning which reflects the short – term interests of the user and has the high rate of learning. On the other hand, the implicit learning occurs during the long-term and reflects the long term interest of the user or the reluctances. Our approach updates the weight vectors by considering these two situations, As described before every profile consists of different descriptors, the negative would be in range $[-1,0]$. While the positive one is in the range $[0,1]$ and the long descriptor is between $[-1,1]$. Every positive or negative vector adding to any category according to ART1 classifier will update those values to reflect any changes in short and long term periods. The interest weight of the long descriptor updates as follow [4].

$$w^c_{lt(new)} = \begin{cases} f\left(f^{-1}\left(w^c_{lt(old)}\right) + \alpha\right) & for\ \ positive\ \ feedback \\ f\left(f^{-1}\left(w^c_{lt(old)}\right) - \alpha\right) & for\ \ negative\ \ feedback \end{cases}$$

f is bipolar sigmoid logistic function, and α is the learning rate.

$$f(x) = \frac{2}{1+\exp^{-x}} - 1$$

Furthermore, the positive and negative descriptors updated as follow,

$$w^c_{p(new)} = w^c_{p(old)} + \left(1 - w^c_{p(old)}\right) * \alpha \quad for\ \ positive\ \ feedback$$
$$w^c_{n(new)} = w^c_{n(old)} + \left(-1 - w^c_{p(old)}\right) * \alpha \quad for\ \ negative\ \ feedback$$

While we update the interest values accordingly, we should reduce the opposite interest's weight to have exact opposition among two positive and negative descriptors. For doing this, we compute the similarity of the input vector and positive vectors and accordingly, reduce the negative vector's effect. We apply this technique reversely for negative vectors.

$$w^c_{n(new)} = w^c_{n(old)} * \left(1 - \alpha * sim\left(d^c_n, k_i\right)\right) \quad for\ \ positive\ \ feedback$$
$$w^c_{p(new)} = w^c_{p(old)} * \left(1 - \alpha * sim\left(d^c_p, k_i\right)\right) \quad for\ \ negative\ \ feedback$$

In the above formula, sim could be any similarity function like cosine similarity function [6] and k_i is the input vector. d^c_p, d^c_n are the average vectors of the positive and negative vectors. Moreover, we also consider Implicit learning to reflect the changing interests and long-term tracing of the user's interests. In this case we trace the activities of the users in a sessions, if in any session there is no vector from interested level, the penalty applies for that category. Otherwise if any page visited the explicit learning with a very little learning rate would be applied. The penalty rate is as follow. For tracing the user on a session, we consider a variable like $flag$ [3] which is zero, if it is not visited.

$$w^c_{p(new)} = \begin{cases} \lambda * w^c_{p(old)} & if\ \ flag = 0 \\ w^c_{p(old)} & otherwise \end{cases}$$

$$w^c_{n(new)} = \begin{cases} \lambda * w^c_{n(old)} & if\ \ flag = 0 \\ w^c_{n(old)} & otherwise \end{cases}$$

$$w^c_{lt(new)} = \begin{cases} \lambda * w^c_{lt(old)} & if\ \ flag = 0 \\ w^c_{lt(old)} & otherwise \end{cases}$$

λ is the penalty factor which is near to 1. In this case if after enough numbers of sessions there is no visit from the interested descriptor, the three descriptors would be decreased. Consequently, after a long period, the weights become zero and the category will be eliminated from the profile.

6. Using Genetic Algorithm For Producing Efficient Queries

The next phase in our algorithm is to generate the best query according to the profile. We use the keywords from the categories and utilize the genetic algorithm [7, 8]. Every individual represents a query which should be ranked according to the fitness function. The fitness function computed using the similarity of the query with the catalogue of the user. We use an optimized Latent Semantic Analysis [9] to compute the similarity of the produces query. The genetic algorithm process is as follow [10]:

1. Initializing the population: For each category, we consider some keywords from interested list, while others from uninterested list, in which we negate them. For every profile we make r queries.
2. Evaluation: By using the fitness function as stated above, each query evaluated.
3. Selection: We select the m best queries, according to the fitness functions.
4. Performing the genetic operations:
    a. Cross-Over: the key words simply crossed using one-point cross over,

$$Q_1 = T_1 \quad AND \quad T_2 \quad OR \quad T_3^{'} \quad OR \quad T_4$$

$$Q2 = T_5^{'} \quad AND \quad T_6 \quad OR \quad T_7 \quad OR \quad T_8$$

consider the query, if the cross over applied on point 2 the reproduction is as

$$Q_3 = T_1 \quad AND \quad T_2 \quad OR \quad T_7 \quad OR \quad T_8$$

$$Q_4 = T_5^{'} \quad AND \quad T_6 \quad OR \quad T_3^{'} \quad OR \quad T_4$$

follow,

    b. Mutation: This operation can be applied on terms which another key-word would be selected randomly from category.
5. After a fixed number of iterations the generation would be stabilized and algorithm has been finished.

7. Latent Semantic Analysis

To determine similarity between two documents we use LSA algorithm. Considering a set of distinct terms as D. A term-document matrix, $X_{m,n}$, where m is the number of terms and n is the number of documents in dataset D.The singular value decomposition (SVD) decomposes the term-document matrix, into three matrices , where U and V are left and right singular vectors respectively and S is a diagonal matrix of singular values ordered in decreasing magnitude.

$$X = USV^T$$

SVD can optimally approximate matrix X with a smaller sample of matrices by selecting k largest singular values and setting the rest of the values to zero. Matrix U_k of size $m*k$ consists of matrix V_k of size $n*k$ along with $k*k$ singular value matrix S_k .

$$\overset{\Lambda}{X}_{m,n} = U_k S_k V_k{}^T$$

Matrix $\overset{\Lambda}{X}_{m,n}$ is known to be the matrix of rank k which is closest in the least squares sense to X . Matrix U_k becomes the latent semantic kernel matrix. To compute the content similarity between query q and document d , we use below formula.

$$contSim(d_x, q) = \frac{d_x^T PP^T q}{\left|P^T d_x\right\|P^T q|}$$

Where matrix P is matrix U_k , and P is used as a mapping function to transform the document and query into concept space to determine the semantic association of document contents [11].

8. Optimized LSA

In this section, we delineate the optimization of our LSA algorithm. Such method could have overhead to the recital of the complete system. However, it effects in more very fruitful construction of the LSA matrixes. This is valued in large datasets and stores. The computation time for building SVD of a matrix $m*n$ is $O(mn^2 + m^2 n + n^3)$ [12].

Hofmann [13] recommended the probabilistic latent semantic indexing (pLSI) which is utilised for written material modelling. In pLSI, a generative type for the documents and their remark occurrences is ushered in as follows. For a document-word pair (d,w). In pLSI, the latent variable z interpreted as the topics for words identifies the connection prospect of a bestowed document-word pair. Let us presume the number of written material N and remarks M are both fastened, hence z have a Multinomial dissemination with dimension N ×M. When we fix this latent variable to be K-dimensional disseminated, the latent variable truly acknowledges a clustering process in the connection space of written material and words. In [13] Hofmann z correspond to projection space for words.

Bose [13] introduced a novel process to diminish the complexity of makeup document/term matrix by adhesive binding and merging the documents. In this way, we don't loss any terms since we exercise all the term for makeup the kernel matrix .There are couple ways to decide the document to fit in the bins. Random selection process decides the document steadily and left them illogically in this bins. While, support based approach decide the bulk noteworthy records steadily for the bins. It escapes the bias caused by illogical selection of the documents .If the training dataset contains $D = \{D_1, D_2, D_3, \ldots, D_m\}$ .Let document D_i has

the maximum of n unique terms $(t_{i1}, t_{i2}, \ldots, t_{in})$ .While later stemming and removing stop-words the frequencies are as pursue $(f_{i1}, f_{i2}, \ldots, f_{in})$ .the total frequency of term T_j is as follow $F_j = \sum_{i=1}^{m} f_{i,j}$. We afterward compute the importance of each document. This process leads smaller amount of documents to fit in every bin. This greatly diminishes the term-document matrix size. We dispense the documents steadily according to their importance and merge all the records in each bin. In lead to compute the document importance, we estimate couple factors, Support of term and the weight. Support of term defines the relative importance of the term in the whole corpus. Weight defines the importance of the term in the document. In this manner, we do not have any bias toward the documents with frequent terms [14].

The weigh of term t_{ij} is computed as follow.

$$W_j = \frac{f_{i,j}}{\sum_{j=1}^{n} f_{i,j}}$$ .t

The size of dataset, F , is total frequencies of terms in dataset .

$$F = \sum_{j=1}^{n} \sum_{i=1}^{m} f_{i,j}$$

Let $S = \{S_1, S_2, ..., S_n\}$ is the support of every term in corpus. The support shows the relative importance of each document among total corpus. It is computed as follow [14].

$$S_j = \frac{F_j}{F} = \frac{\sum_{i=1}^{m} f_{i,j}}{\sum_{j=1}^{n} \sum_{i=1}^{m} f_{i,j}}$$

$$DI_i = \sum_{j=1}^{n} W_j * S_j$$

Let B be the accumulation of q bins, $B = \{B_1, B_2, \ldots, B_q\}$,containing the same number of written documents from dataset in the bins. Let BD be a accumulation of q number of documents $BD = \{BD_1, BD_2, \ldots, BD_q\}$ where BD_i is the joining documents after the dissemination algorithms in bin_i .After augmenting the frequency of merged terms in documents ,term with small number of frequencies (less than 5) or very high figures (above

5000) deliberated outliers. Such redundant term eliminated from term document matrix .In random selection, we select equal number of documents for each bin at random. While for support based, we select the equal number of document according to their r importance. In this way, we have similar important documents in each bin. Such method avoids any bias caused by random selection and boosts the accuracy of kernel. We use Bose's algorithm for selection of documents for bins as follow [14].

// K is term-document matrix
// P is Latent Semantic Kernel
//Calculate the document importance
1. for each $D_i \in D$.
    a. $DI_i = \sum_{j=1}^{n} W_j * S_j$
2. end for
3. Sort DI in ascending order.
4. //Bin documents
5. for $i = 0$ to q
    a. $B_i = 0$
    b. for $j = 0$ to m/q
        i. $B_i = B_i \cup D_{j*q+i}$
    c. end for
6. end for
7. //Merge the documents
8. for each $B_q \in B$
    a. $BD_q = empty$
    b. for each $D_i \in B_q$
        i. $BD_q = BD_q \cup D_i$
    c. end for
9. end for
10. //Initialize the term-document matrix
11. for each $x = 0$ to T_j
    a. for each $y = 0$ to D_i
        i. $K[x][y] = 0$
    b. end for
12. end for
13. for each $BD_q \in BD$
    a. for each $T_j \in BD_q$
        i. $K[T_j][BD_q] = K[T_j][BD_q] + S_j$
    b. End for
14. end for
15. perform SVD on K
16. $P = U_k$

It is inferred that pLSI is not a well-defined type, since it delights each document as an index and hence is not generalizable to new documents. Another obstacle of pLSI is that longer documents get higher weights in the type, which in addition suggests that the documents are not alone sampled. Moreover, pLSI uses a probabilistic method such as Na ïve bayes algorithm to compute the probability of each topic. This implementation is obviously less efficient comparing to method proposed by Bose.

9. TF-IDF Vector Representation

One of key issues in content similarity, is to properly score the best words for classification algorithms [14].In this algorithm, we use $tf * idf$ to update our kernel resulted from previous section. In this algorithm, the term specific weights in the document vectors are products of local and global parameters. The model is known as term frequency-inverse document frequency model. The weight vector for document d is $V_d = [w_{1,d}, w_{2,d}, ..., w_{N,d}]^T$,where $w_{t,d} = tf_{t,d} . \log \frac{|D|}{|\{d \in D | t \in d\}|}$ and $tf_{t,d}$ is term frequency of term t in document d.

$\log \frac{|D|}{|\{d \in D | t \in d\}|}$ is inverse document frequency , $|D|$ is the total number of documents in the document set and $|\{d \in D | t \in d\}|$ is the number of documents containing the term t.

10. Fitness Function

To compute the fitness of each query produces by genetic algorithm, we add the similarity gained by Optimized LSA for each positive document in the catalogue. Moreover, we subtract them with negative feedbacks and divide it by the number of catalogue feedbacks.

$$Fitness(q) = \frac{\sum_{c \in C_p} OpLSA(q,c) - \sum_{c \in C_N} OpLSA(q,c)}{n}$$

where $OpLSA(q,c)$ computer the semantic similarity of the query q and document c. C_p represents the positive feedbacks in positive feedback of the profile and C_N denote the negative feedback in the user ‘s profile.

11. Evaluation Results

In this section, we explain the properties of our algorithm, Firstly; we want to know about the learning rate of the algorithm. We use the following steps to do our experiments [4].

1- Consider 50 WebPages related to same categories from the internet.

2-Generate a random profile with equal probability with positive and negative feedback.

3- Rank all the pages and select the last ranked page D.

4- Give the page a positive feed back.

5- Use the new profile to rank all the pages again and compute the changing percentage of the selected page D.

6- Repeat the above steps for 20 different learning rates.

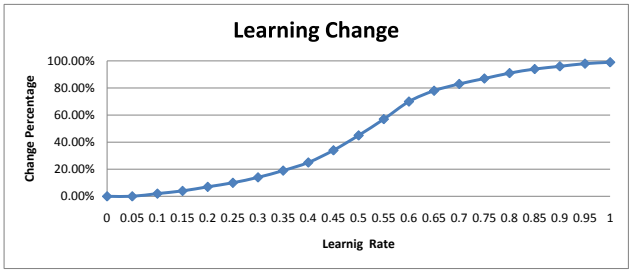

Figure 3. The Effect of Learning Rate on Changing the Page Ranks

We repeat our experiments 10 times for different datasets and got their average to have a better accuracy. As seen from the Figure 3, the values less than 0.2 has little effect on page rank's changes. Therefore, we select 0.2 for our implicit learning algorithm, while, the values above 0.7 has strong effect on page rank's changes [3].

Another experiment we can show is the vigilance parameter of our ART1 Network. To see its effect on the number of classifications on our data sample, we made use them in text classification. The results show the amount bellow 90 % resulted in poor categorizations, mostly, because the network put them in one category. We used 100 documents from five different categories from some news agency. We applied different values, 90 %, 95 %, and 99 %. The results are shown in the following table.

Table 1. Sensitivity of Number of Categories to Vigilant Parameter

	$\rho = 0.90$	$\rho = 0.95$	$\rho = 0.99$
Technology	6	7	14
Finance	8	11	14
Health	1	12	17
Entertainment	2	7	13
Science	2	8	13

The results show for better accuracy, we should use 99% for vigilance parameter.

To construct our kernel, we used Wikipedia datasets, available from the INEX 2006 Document Mining Challenge [15]. We used about 3000 document in 60 categories. To determine best method for semantic similarity, we used different approaches described

in this paper. We used a simple $tf * idf$ algorithm .the second and third methods were to use $tf * idf$ representation along with random selection and support based selection merging algorithm.

For evaluating the accuracy of classifications, we use precision and recall measurement metrics stated in [17].

Table 2. Evaluating Parameters

	Assigned to C_i	**Not assigned to C_i**
Belonging to C_i	tp	fn
Not Belonging to C_i	fp	tn

C_i represents the clusters. The Precision shows the accuracy of the algorithm while the recall represents the integrity of suggestion algorithm.

$$\Pr ecision = \frac{tp}{tp + fp}$$

$$\mathrm{Re}call = \frac{tp}{tp + fn}$$

There maybe instances which the classifier does not categorize them. Therefore, it reduces the Recall. We also use another parameter $F - Score$ can be computed as follow.

$$F = \frac{2 * \Pr ecision * \mathrm{Re}call}{\Pr ecision + \mathrm{Re}call}$$

To test the accuracy of the classification, we used 200 documents from 20 categories and determined the similarity of each using Bi-Section KMeans clustering method [18]. We clustered the pair-wise document similarity matrix produced from our clustering approach and gained the accuracy of each method.

Table 3. Results of Different Aproches

Methods	**Merging Algorithm**	**F (%)**
Latent Semantic Kernel	Support Based	84.56 %
Latent Semantic Kernel	Random	79..37 %
tf*idf	-	74.41%

In the next section, we show the results of our genetic algorithm, after feeding back 100 pages in profile for searching the word “Germany”.

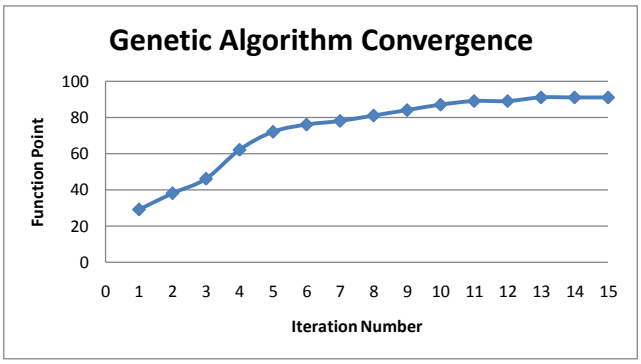

Figure 4. GA Convergence

As we see after 15 iterations the Function Point 91 fixed in our experiments and gives the best query.

12. Conclusion

In this paper we presented an algorithm for classifying the search results, by the help of ART1 network. This is a fast and real forwarded approach, comparing to other similar methods. Since using ART1 doesn't need a supervised method, it is suitable for on-line tracings and classifications. Moreover, we used an optimized LSA to get the similarity of the queries produced with the profile of the users. We concluded the promising way is using $tf * idf$ representation along with support based binning algorithm which enhanced LSA.

References

[1] N. J. Belkin and W. B. Croft, "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", In Communications of the ACM, December 1992, vol. 35, no. 12, **(1992)**, pp. 29-38.
[2] Agent Working Group, "Agent Technology Green Paper". OMG Document agent/00-09-01 Version 1.0, **(2000)**.
[3] M. Bazarganigilani, "Online Information Filtering using User Profile ART1 Classifier and Genetic Algorithm", 2nd IEEE International Conference on Software Technology and Engineering, ICSTE 2010, San Juan, Puerto Rico, USA **(2010)**.
[4] D. H. Widyantoro, "Learning User Profile In Personalized News Agent", Master Thesis, Department of Computer Science, Texas A&M University, **(2006)**.
[5] Y. -J. Chen, Y. -M. Chen, H. -C. Chu, C. -B. Wang, D. -C. Tsaih and H. -M. Yang, "Integrated Clustering Approach to Developing Technology for Functional Feature and Engineering Specification-based Reference Design Retrieval",Concurrent Engineering, vol. 13, **(2005)**, pp. 257.
[6] R. Baeza-Yates and B. Ribeiro-Neto, "Modern Information Retrieval", ACM Press, Addison-Wesley, Reading, MA, **(1999)**.
[7] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, **(1989)**.
[8] J. H. Holland, "Adaptation in Natural and Artificial Systems", The University of Michigan Press, **(1975)**.
[9] T. Landauer, P. W. Foltz and D. Laham, "Introduction to Latent Semantic Analysis", (PDF) Discourse Processes, vol. 25, **(1998)**, pp. 259–284.
[10] M. Kalantar, "Adaptive Web Information Filtering System Using Genetic Algorithms", Master Thesis, Ferdowsi University, **(2003)**.
[11] T. Tran, R. Nayak and P. D. Bruza, "Combining structure and content similarities for XML document clustering", In: 7th Australasian Data Mining Conference, Glenelg, South Australia, **(2008)** November 27-28.

[12] D. Widdows and K. Ferraro, "Semantic vectors: A scalable open source package and online technology management application", In: Proceedings of the sixth international conference on Language Resources and Evaluation (LREC 2008), Marrakesh, Morroco, **(2008)**.
[13] T. Hofmann, "Probabilistic Latent Semantic Indexing", In Proceedings of the 22nd Annual ACM SIGIR Conference, Berkeley, California, **(1999)** August, pp. 50–57.
[14] A. Bose, "Effective web service discovery using a combination of a semantic model and a data mining technique", master thesis, Faculty of Information technology, Brisbane, Qeesland , Australia, **(2008)**.
[15] G. Salton, A. Wong and C. S. Yang, "A Vector Space Model for Automatic Indexing", Communications of the ACM, vol. 18, no. 11, **(1975)**, pp. 613–620.
[16] L. Denoyer, P. Gallinari and A. -M. Vercoustre, "Report on the xml mining track at inex 2005 and inex 2006", in `INEX 2006', Dagstuhl Castle, Germany, **(2006)**, pp. 432-443.
[17] C. W. Cleverdon and J. Mills, "The testing of index language devices", Aslib Proceeding, vol. 15, no. 4, **(1963)**, pp. 106–130.
[18] A. Hotho, A. Maedche and S. Staab, "Ontology-based text clustering", In Proceedings of the IJCAI-2001 Workshop Text, **(2001)**.