Chapter 3

Proposed systems

3.1 Motivation

Semantic technologies promise a next generation of semantic search engines. General search engines don’t take into consideration the semantic relationships between query terms and other concepts that might be significant to user. Thus, semantic web vision and its core ontologys are used to overcome this defect. The order in which these results are ranked is also substantial. Moreover, user preferences and interests must be taken into consideration so as to provide user a set of personalized results.

Through personalization, one can improve the navigation on a Website by, for example, highlighting content and links of interest, hiding those that are irrelevant, and even providing new links in the site to the users likely web destinations. While personalization can help to identify relevant new information, which can create problems in re-finding when presented in a way that does not account for previous information interactions. This work presents a model of what people remember about search results, and shows that it is possible to invisibly merge new information into previously viewed search result lists where information has been forgotten. Personalizing repeat search results in this way enables people to effectively find both new and old information using the same search result list.
The main characteristic of agent-based technology is that the structure of the software is represented by a group of agents who collaborate in achieving the goal of the task in hand. The combination of Information Retrieval and Multi-agent technology has the following features: Adaptability, Initiative and Collaborative. Among different types of agents, the personal assistant agents are particularly interesting to this research. This type of agents operates at the user interface level and actively assists users by offering information and advice to the users. These agents usually apply a kind of intelligent learning algorithm so that they can intercept the user input, examine it and take actions that are more specific to those particular users needs at that moment. These agents are also called learning or adaptive agents. Agent can initiatively retrieve the corresponding information based on users demand, and even can monitor the changes of information sources and agents also share the information with other agents.

This work introduces a Personalized Information Retrieval system in Semantic Web based on multi-agent, which can accomplish Information Retrieval according to user interest knowledge via multi-agent collaboration for providing personal service to user. In the process of personal Information Retrieval, the precision and quality depend on the various degree that the system manage user interest. Therefore, this work solves problems like construction of user interest model based on vector space, and Updating of user interest model in time when users interest changes.

The proposed system suggests that the Personalization process be taken to a new level where the user does not to be actively involved with the personalization process. All that the user needs to do is to have an active profile file and when the user logs onto a Website, the browser checks for that profile file as it checks for the cookies. The profile file describes the users interest and the levels at which the user wants a particular Personalizable feature. Classic Information Retrieval usually used ranking algorithms based solely on the keywords in the documents. The proposed system proposes a different ranking based on URL. The ranking is query-dependent. The
proposed algorithm assigns a score that measures the quality and relevance of a selected set of pages depending on their URL to a given user query. The basic idea is to build a query-specific two dimensional vector table, called a related vector table, and perform URL analysis. The outcome of this analysis is used for ranking which is slightly different ranking compared to existing one. In current research work the system use hybrid approach to find ranked. In order to address this issue, this work proposed, an architecture for Agent Based Personalized Semantic Web Information Retrieval System (ABPSWIRS).

The Agent Based Personalized Semantic Web Information System should be able to

- perform a search for users regardless of user type.
- search by different criteria for authorized users.
- perform a search taking into account the personal needs of the user.
- provide user relevant queries information.
- keep statistics of requests and, if necessary, provide this information.
- remember the successful search results.
- design and implement the architecture of search retrieval module based on search algorithm.
- test search module of ontology data model for data and metadata exchange repository on a large amount of input data.
- research and analyse various strategies for the search module (analysis and test various strategies to search engine, search optimization).

The extension prototype consists of:

- a search plug-in, which provides the user with a query search bar integrated into the browser.
• a suggestion tool, which shows a list of suggested additional terms while
the query is being formulated by the user.

• a tracking tool, which collects relevant activities performed by the user.

• Relevance feedback tools, which let the users express their own
relevance assessments on each record retrieved by the user, thus
affecting the feedback terms provided by the suggestion tool.

Main dataspaces were defined are

• Semantic, which includes term and relationship representation

• Statistical, which includes the frequency distribution of pairs of terms
stored log files.

• Personal user profile, which includes past users queries and users
behaviour data such as clickthrough, time spent on a page, explicit and
implicit relevance assessments

3.2 ABPSWIRS Architecture

This ABPSWIRS architecture, can helps users to get the relevant WebPages
based on their selection from the domain list, so that users can obtain a set
of related WebPages from the system. A high level view of the overall system
is given in Figure 3.1. The overall scenario utilizes ontology as the basis to
transform query and semantic features in the context repository into semantic
pattern so as to identify the corresponding contents. ABPSWIRS is a crawler-
based search engine that makes use of crawler to collect resources from both
semantic as well as traditional web resources.

The Semantics of the query is analysed by means of the following
procedures:

• The user query is initially analysed grammatically and syntactically by
parsing.
The domain related keywords in the ontology are retrieved to form the refined query. The results obtained are more relevant by adopting the following procedure.

- The refined queries that serve as the input for the search engine is formed based on the semantic analysis of the user query.

- The web links retrieved for all the newly formed refined queries are re-ranked based on the domain specific information.

In this way the proposed system provides a semantic search that retrieves the appropriate results for the user query. This system consists of different components like User Agent, Semantic Extraction Agent, Semantic Searching Agent, Filtering Agent, Personalized Ranking Agent and Knowledge Base. All agents are monitored entirely to fulfill proprietary system functions, including Information Retrieval and Knowledge Base update.
3.2.1 User Agent

User Agent is the mutual interface between user and system, which provides a friendly platform to users. It can construct user interest model according to Users browsing history record and registration data. User Agent incepts users retrieval request which is transformed to prescriptive format, and transmits the formatted user request to Semantic Extraction Agent to expand the query based on the respective domain and related terms based on ontology. User Agent also takes overall results from Personalized Ranking Agent, and presents personally these results to user. In addition, User Agent presides over creating a profiles user for new user. Users browsing or evaluating behaviour can also be stored as profiles and it is learned by User Agent, so user interest model may be updated and improved in time. The user interface proposed a prototype consists of a query suggestion tool. While the user types his/her query in the search bar, several lists of possible suggestions are populated. The suggestions are listed by context and the user can choose his/her preferred context by hovering a different part of the selector. If the user chooses one of the suggested term/phrase, the search bar text field is filled with the selected as he/she would do with every other search plug-in (usually with the default Google plug-in).

User Agent includes Environment view, Memory Base, Knowledge Base, Learning mechanism and Inference Engine. (Show in Figure 3.2)

- The view of the environment module in the User Agent is the user’s input and output interface.

- Memory Base records the original information entered by the user.

- Knowledge Base defines a user’s personal knowledge, classified information and the user model.

- Learning mechanism is used to summarise the behaviour of users and formats the information.

- Inference engine infers the user’s interest based on the analysis.
Users are required to register their basic information, when they use the system for the first time. The system offers each user with a user profile after gathering user browsing behaviour and stores it in the user interest profile. Afterwards, the system utilizes the user profile and recommends resources to the user. At the meantime, the system updates the profiles via mining the users log.

### 3.2.1.1 Automatic creation of user profiles

This section describes how the user profile is constructed and updated in the system. Any personalized system relies on the use of some form of user profile. The user profile may be created explicitly, by users filling in online or implicitly from information gathered from users as they use the system. It may reside on the client machine or at the server. It may be temporary, created for each user session, or persistent, stored and reused sessions. Finally, the user profiles may be represented as a set of attribute values stored in...
databases, keyword vectors, or ontology.

In proposed system, the user profile is created automatically and implicitly while the users browse. The user profile is essentially a reference ontology in which each concept has a weight indicating the perceived user interest in that concept. Profiles are generated by analysing the surfing behaviour of the user, specifically the content, length, and time spent on each page they visit.

User profiles can be constructed on the user’s machine where the re-ranking of web search results can be done using the profile then presented to the user. The other way is not only construct the user profile on the user’s machine but also do the re-ranking computations on the same machine.

The main advantage of re-ranking top results on the user’s machine is privacy. Profiles are typically constructed using browsing history, visiting time, users feedback on documents on the user’s machine etc which are very sensitive for some users. They may not like that information to be stored on the Internet. Another advantage is that because the system re-rank only top results, the computation overheads are not so high on a user’s machine. ABPSWIRS constructs the user’s profile on the user’s own machine. Various techniques from machine learning and Information Retrieval have been used to construct the user’s profile. It takes both explicit and implicit feedback from the user to construct the profile. The profile is also dynamic in the sense that it reflects changes in interests of the user over time. The construction and update of the profile is fully automated.

For example created ontologies for the user profile, the culture and the religion. Then, created the necessary relations between them and the food and health ontologies. The user profile ontology is represented as an ontological concept that consists of many properties as shown in Figure 3.3. For clarification, we visualize the user profile ontology properties in four categories: one category has the users basic information, such as name and age; one category has the users basic health information, such as the weight
and the blood type; one category has the users medical information, such as the diseases and allergies; and finally, one category has the usage statistics, such as previous searches and user feedback. The arrow represents a relationship between two concepts, which is referred to in RDF terminology as triple.

![Figure 3.3: User Profile Ontology](image)

### 3.2.1.2 Information Sources

Before the user profile construction a system needs to identify the interests of users, the system uses both the implicit feedback and explicit approach to decipher the interests of a user. The information sources currently used are:

1. The assumption is that if a user keeps a document on his/her machine there is a strong possibility that the user is interested in those documents.

2. Pages browsed are of interest to the user but pages that are frequently visited and those on which more time is spent can be taken as useful documents for the user’s interest.

3. The download history can be used as a starting point for deciding favorite directories for download. This would be more useful in case of those users...
who organize files on their machines properly. These are the sources the current system have used in constructing a user’s profile.

3.2.1.3 Training the Classifier

In the process of training classifier, a fixed number of sample documents for each concept are collected and used as examples for the concept. These are the WebPages that is associated with a node in the subject hierarchy, which form the training set. Use a vector space classifier in which each concept is represented by a vector of weighted terms. The terms are extracted from the training set and weighted using a variation of the vector space term weighting formula. As shown in Equation (3.1), the weight of term $i$ in concept $j$ is calculated as

$$ W_{t_{ij}} = t_{f_{ij}} \cdot icf_i \cdot cdf_{ij} $$

where $t_{f_{ij}} =$ the total frequency of term $i$ in each training documents for concept $j$

$icf_i = \log(TC/NC)$ which may be called as the inverse concept frequency

$cdf_{ij} = \log(NTD)/(TTD)$ which may be called as the concept document frequency

$TC =$TotalConcepts

$NC =$ Number of Concepts containing term $i$

$NTD =$ Number of Training Documents for concept $j$ containing term $i$

$TTD =$Total Training Documents for concept $j$

3.2.1.4 Building the User Profile

The representative text collected for each user is periodically classified into the appropriate concept(s) in the reference ontology. For each of the text samples, a document vector is calculated using the same formulae used for the concept vectors. The similarity between the vector for sample$k$ and the vector associated with concept $j$, $c$, was calculated using the similarity measure. The concepts with the highest similarity values were assumed to be those most related to the sample text.
Initially, a user profile starts off with all concepts in the ontology having a weight of zero. As sample texts are classified with respect to the reference ontology, the values reported by the classifier are added to the top five concepts weights, i.e., the user profile. Over time, as more and more text is classified, the weights are accumulated. Concepts into which many representative texts are classified continue to increase in weight, and it is our hypothesis that higher weighted concepts represent concepts of greater user interest.

### 3.2.1.5 Updating the Profile

Every two weeks the documents can be clustered again and the clusters would represent new interests between the updates. The system keeps track of changing interests by following a simple intuitive idea. Suppose a user has 10 interests identified after clustering. Of these 10 interests he/she may be more interested in some and less interested in others. The interests which are more near to a user would mean that the user browses or downloads more of such content in between updates. So, if the system assigns weight ages to interest vectors on the basis of documents downloaded and browsed, get a fairer representation of a user’s current interest. As far as weight ages are concerned they can be assigned proportional to the number of documents assigned to each cluster on the basis of the similarity metric. Profile can be updated by getting implicit/explicit feedback from the user and also based on the browsing history as follows:

### 3.2.1.6 Collecting feedback

**Explicit feedback**

In the proposed system the information manually provided by the users when they change their degree of interest (DOI) in existing concepts of the application domain, for instance using a numeric value is automatically updated in user profile. The information provided by means of rating specific items of the domain whether they have been previously recommended by the
system or not. In this case, the degree of interest in the concepts associated with each item rated are updated properly depending on if the rating has been positive or negative.

**Implicit feedback**

The time spent by the user viewing the item information can be taken into account as implicit interest to give more or less importance to the particular behaviour and therefore to the statistics associated. The less the time of viewing the less important the impact on the statistics of the concepts associated with the item selected. In contrast to the explicit-feedback collection-methods, the implicit feedback collected is only used to infer positive evidences of user interest, since it has been demonstrated that this kind of feedback is not a good indicator for negative evidences.

**Browsing histories**

The initial investigations of personalized search built user profiles from browsing histories. System can collect the visited URL over a period of time using their browser caches. Each page is classified and the results are added to the profile. Since the classifier returns a list of matching concepts in decreasing order of weight, by the rating a usage of the list and adding to the profile. For these experiments, the top matching 5 concepts for each WebPage is used. The correct concept occurs in the top 5 categories 80 percentage of the time, and the classifier accuracy falls off dramatically on, on progressing further down the list. The system also investigated the influence of other two factors such similarity calculation and time factors in the page concept.

similarity calculation used when building the user profile are the duration of the visit and the page length. Intuitively, if a user spends a long time on the page, their interest value in that page should be increased. However, if the page is long, the influence of the time factor should possibly be decreased since the increased time may be due to the amount of information presented,
not the level of interest. The relation used to combine the time and page length factors to adjust the weight of a concept $c_j$ in a user profile. This happens on the grounds of a previously visited document $d_k$ classified into $c_j$.

\[
update(dk, cj) = \text{timelengthfactor} \ast \text{similarity}(d_k, c_j) \quad (3.2)
\]

where timelengthfactor is calculated in one of two ways:

\[
\text{timelengthfactor} = \frac{\text{time}}{\text{length}} \quad (3.3)
\]

\[
\text{timelengthfactor} = \log\left(\frac{\text{time}}{\text{length}}\right) \quad (3.4)
\]

\* time = the amount of time the user spent visiting the page in seconds  
\* length = the length of the page in bytes

The timelength factor formulas are the time spent browsing the page by the length of the page. The DOI (Degree Of Interest) for a particular topic (DOIweight) is calculated by means of combining a fixed set of weights consisting of real values that are obtained from different information sources and learning approaches.

- Feedback manually provided by the user - The feedback weight (fw) is set when the user manually assigns a DOI for a particular topic through the Web application. The range of possible values is between $[-1, 1]$; where -1 indicates the user does not like at all items related with the topic, and 1 that is very interested in items related with the topic.

- Ratings-based information - The rating weight (rw) is calculated using the average of past ratings of the items related with the topic. The range of possible values also is between $[-1, 1]$; and the meaning is the same that in the previous case.

- Usage-data-based information - The usage weight (iw) is calculated as the probability that the user is interested in the topic based on a weighted sum of the number of its occurrences according to the users statistics in relation to the occurrences distribution for all users (also
called normalized probability). This probability is calculated using a sigmoid function, so if the number of occurrences is greater/lower than the standard deviation of the distribution, then the value is near to 1 or 0 respectively. Depending on the type of user behaviour (query or item selection) a different weight is given to the specific statistic. When the number of users and events in the system is lower than a threshold, the normalized probability is calculated using the number of occurrences distribution of each particular user. The range of values is [0, 1]; where 0 indicates non-interest, and 1 that the user is completely interested. The algorithm **ITEMSCORE** is used to calculate the item score for a particular user works as follows:

Algorithm for finding ITEMSCORE:

---

**Input:**

- item - represents an instance Item object
- user - represents the user profile
- featuresRelevances - set of features type relevances
- ismatchconcept - status variable

---

**Output:**

- conceptScore (represents the predicted score associated with item)

---

**Parameter used:**

- sum - find the total weight

---

**procedure ITEMSCORE(item,uid,feed)**

```java
{
    FOR EACH items concept
    {
        WHILE ismatchconcept==FALSE
        {
            IF has_matching(iconcept) THEN
```
3.2.1.7 Semantic Extraction Agent

Semantic Extraction Agent aims to find the semantic features in the users queries. It will make use of agent technologies and ontology technologies to analyze the association relation in the users queries and document to extract semantic features. This module contains the following components:

- Query Preprocessing: Meaningless words like neuter pronouns, articles,
Semantic Analysis: This component identified semantics elements like Subject, Property, and Object in the Query content and analyzes their semantic relations.

Semantic Matching: In the personalized Information Retrieval system, Semantic matching agent takes charge of receiving formatted user request from User Agent, and the user request is expanded (based on ontology) according to user interest. Afterwards, the user request is transmitted to Semantic Searching Agent. It analyses the returned data from Searching Agent, filtrating useless information, and processed results are send to user.

Figure 3.4 shows the Semantic Extraction process steps. In the proposed view of the semantic search, it is assumed that the information available in standard WebPages is indexed using semantic knowledge found in the semantic web. A key step in achieving this lies on linking the semantic space to the unstructured content space by means of the explicit annotation of documents with semantic data. In ABPSWIRS, it is assumed that a knowledge base has been built and associated to the information sources by using one or several domain ontologies that describe concepts appearing in the document text. This system can work with any arbitrary domain ontology with essentially no restriction, except for some minimal requirements, which basically consists of conforming to a set of root ontology classes.

3.2.1.8 Query processing steps

After getting the users query, a spell checker is used to check the spelling of the query and suggest corrections if needed. Next, noise words, such as do, does, an, the, etc., are removed in order to have only the words that could be related to the domain ontology. Then, the system identify the concepts related to education ontology through a populated list of all the ontologys classes and the knowledge bases instances. After that, domain ontology is used to identify the possible relations between these concepts by finding all that
are synonymous with the pre-defined relations. Next, system enrich the query based on the user profile. Then, match the identified concepts and relations to the best query. Finally, a semantic annotation that represents the user's query is produced for retrieval. Figure 3.5 shows the query processing steps.

3.2.1.9 Semantic Analysis

The personalization starts from the query processing time, utilizing the user profile ontology to enrich the query. The user profile ontology has defined relations. The properties of the user profile ontology can be used not only to enrich and expand the query. This leads to more accurate and relevant results by filtering the mass result records based on the user profile.

Algorithm For Query Expansion using domain ontology

Input:
Original query terms set (Qor) where Qor = t1 , t2tn

Output:
Query terms set (Qset) where Qor Qex Qex is the expanded query terms

Qor - Query set

D_o  - domain ontology
R_o  - related ontology
P_o  - possible ontology

QUERYEXPAND(Qor)
{
    Qset={empty}
    Get Qor and add it to Qset.
    // split each word in the query and stored as Qset
    For each term t_i in Qset
    {

If ( \( t_i \) in \( D_o \)) then
{
    If ( \( t_i \) in \( P_o \)) and ( \( t_i \) in \( R_o \)) then
    {
        \( Q_{ex} = P_o + R_o \)
    }
}
Else If ( \( t_i \) in \( D_o \)) then
{
    If ( \( t_i \) in \( D_o \)) and ( \( t_i \) in \( R_o \)) then
    {
        \( Q_{ex} = D_o + R_o \)
    }
}
Else If ( \( t_i \) in \( P_o \)) then
{
    If ( \( t_i \) in \( D_o \)) and ( \( t_i \) in \( P_o \)) then
    {
        \( Q_{ex} = D_o + P_o \)
    }
}
}}

once get the search term from the user the system take each term from the query find its semantic relation from the domain ontology. For e.g., If the user enter the keyword like deadlock, it may be related to process management and related to Memory Management in Operating system ontology of computer science domain. The ontology can be updated dynamically for any number of domain and its related items.
3.2.2 Semantic Searching Agent

This component is responsible for searching and retrieving relevant results. The purposive seeking of information as a consequence of a need to satisfy some goal. Information seeking includes all activities directed towards accessing information to meet an information need. The need can be very specific, like a list of breast cancer treatments, or something broad and less defined, like learning about the disease in general. Finding activities include the specification of a focused query to a search service (e.g., typing breast cancer treatments into a search engine’s query box) as well as less directed browsing.

Finding of previously viewed information is an incredibly important piece of finding, so re-finding is defined here. The finding of new information and the re-finding of previously viewed information are often interleaved as part of a larger information-seeking task. For example, Connie may want to both find information about new breast cancer treatments and re-find the list she found initially in the process of deciding the next step in her treatment.

3.2.2.1 Query Evaluation

Given a query term \( q \) as input, the system returns the set of the \( n \) co-occurring terms with respect to \( q \) with the highest frequency. The component can also provide a list of possible completion words (i.e., all the terms for which \( q \) is a prefix) together with the corresponding suggestions. An example set of suggestion terms for \( q = "information" \) is: [information, information systems, information science, information technology, information retrieval, information system, information literacy, information society, information management, information library]. The queries evaluated were selected in two different manners. In one approach (self-selected queries), participants were asked to choose a query that mimicked a search they had performed earlier that day, based on a user history. In another approach (pre-selected queries), participants were asked to select a query from web.
The second approach is essentially based on the commercial search engines (i.e. Google and Yahoo!). The Semantic Searching Agent takes as input the lists of suggestions from search engines by using the available APIs. In the process, matching algorithms are presented to enable fast matching and searching for content. For each search result, the participant was asked to determine whether they personally found the result highly relevant, relevant, or not relevant to the query. The results were presented based on relevancy order.

3.2.2.2 Components

Semantic Searching Agent mainly includes Search Strategy, Search Optimization and Crawler.

- Search Strategy includes depth-first search strategy

- Crawler it is a program that crawl pages based on the hyperlinks between webs to collect information. During crawling, crawlers (robots) traverse the web to collect web resources.

- Search Optimization includes the way of accessing the page should be subject to management of websites and the frequency of visiting, collecting important WebPages which have high page weight and have changed, and ensuring that pages will not be repeated crawled.

As crawlers go through a stage of URL extraction. HTML crawlers extract links from HTML pages in order to find additional sources to crawl. This mechanism usually doesn’t work for semantic resources, as there exists no direct concept of a hyperlink. That’s why our system uses multi crawlers that can traverse both traditional as well as semantic web.

Crawlers (seen in Figure 3.6) are initiated within the collector. The collector receives a URL to be crawled passes it to URL Extractor which in turn extracts URLs from page. The filter then checks if URLs are already visited, stores them into database, associates related URLs, and queues them for further examination. Our crawling algorithm is the standard depth-first
search algorithm as it has been known for its high quality results. Moreover, in order to be able to scale, crawling is performed in parallel.
Algorithm for semantic search

Input:
Semantic Query

Output:
Set of related URLs

Parameters:
Qor - semantic query

Semantic search(Qor)
{
  Q{set}=Queryexpand{Qor}
  // expand user query based on ontology
  For Each term in Qor {
    If the term is found then
      {
        Retrieve the relevant WebPage from in knowledge base/web
db()— URL
        // Store the query and WebPages to db
      }
    Else
      // new ontology term
      {
        Update ontology with new terms
      }
  }
}

A new user poses a query that goes to a search engine, results are retrieved
and relevant documents are marked by down the users clicks to the pages or asking the user through feedback. Now, an old user poses an old query topic i.e query topic used earlier or similar; context from the user profile is retrieved using algorithm (QUERYEXPAND) and query is expanded with context. Subsequently, the user profile is updated with weight of terms. Henceforth, the expanded query goes to the search engine documents are retrieved and re-ranked using algorithm and results are presented to user.

3.2.3 Filtering Agent

Web Query Suggestion and Ontology Term Suggestion services are responsible for retrieving the appropriate information from all the described database at run-time, so as to provide the end-user with personalized or/and context aware query and term suggestions, as a response to an input query. Therefore, the output results generated by these two services serve the same purpose, i.e. to aid the user in query reformulation. However, there is in fact no interaction between the computation and final production of the two types of outputs. For this reason, they have been implemented as two distinct, separate services, each producing its own output.

3.2.3.1 Filtering from profile

The Profile Context was a hybrid between the web Context and the User Context and was obtained by re-ranking the suggestions computed by the web Context on the basis of the users relevance feedback assessments provided. The algorithm was as follows:

- Extract the k top-ranked suggestions using the web Context.
- Search a database selecting the records for which a relevance feedback was submitted. The records are processed by extracting the co-occurrences of semantic terms for the given keyword, for each pair of terms the frequency and the relevance assessments which could be: +1 for positive feedback, -1 for negative feedback, 0 for no feedback.
Lastly, the frequencies of the k top-ranked suggestions are recomputed using the relevance assessments as correction factor.

The Web Context is built by using the Google API effort in making a Web service available for thesaurus definitions. If a keyword can be found through the API, the suggestions can be retrieved and presented to the user.

3.2.3.2 Filtering from Web Context

Given a query q, sends the query to Google which return the surrogates of the top k results. In the current experimental environment assume k = 20.

3.2.3.3 Components of Filtering Agent

Components included in this module are: Knowledge Base, Model Base, and Semantic Matching. (Shown in Figure 3.7)

- **Knowledge base**: It includes the user’s personalized information transmitted by the User Agent. When matching, Semantic Matching will make use of personalize information (users gender, users age, users occupation, users search history and so on) to match and search more accurate and useful information for users.

- **Model Base**: It includes a variety of information retrieval model and matching algorithm, for example: Bayesian Probability Model, Support Vector Machine (SVM), Neural Network Algorithm and so on. It also includes new algorithms based on semantic pattern which can solve some traditional problems.

- **Semantic Matching**: According to the Model Base, this component will chose apposite model and algorithm to matches semantics in users queries and semantics in the documents, and in accordance with the relevant, the results will be submitted to the User Agent.

In proposed system Google APIs are used for query implementation. GoogleAPI is a wrapper for Google that implicitly collects information from users. When queries are submitted by users, API file forwards the query to
the Google search engine. It intercepts the search engine results, logs along with the query and the userID, re-ranks them, and then displays to the user. When users click on a result, the system logs the selected document along with the user ID before redirecting the browser to the appropriate WebPage. The log file is split between users and, for each user, further divided into training and testing sets.

API allows users to submit queries to exiting search engines to display results using GR; to store queries issued along with the first 10 results displayed. Each entry of the web log file had the following format: (timestamp, user ID, query, URL, GR). Each entry of the user profile file had the following format: (user ID, query, timestamp, user feedback, TF, TW, access date, URL). Table 3.1 and Table 3.2 shows basic structure of weblog and user profile.
<table>
<thead>
<tr>
<th>UserID</th>
<th>Query</th>
<th>Timestamp</th>
<th>URank</th>
<th>TF</th>
<th>TW</th>
<th>Access Date</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ud1</td>
<td>q1</td>
<td>87</td>
<td>10</td>
<td>788</td>
<td>45</td>
<td>D1</td>
<td>URL1</td>
</tr>
<tr>
<td>Ud2</td>
<td>q1</td>
<td>187</td>
<td>8</td>
<td>800</td>
<td>20</td>
<td>D1</td>
<td>URL2</td>
</tr>
<tr>
<td>Ud1</td>
<td>q2</td>
<td>83</td>
<td>5</td>
<td>300</td>
<td>30</td>
<td>D3</td>
<td>URL3</td>
</tr>
</tbody>
</table>

Table 3.2: User Profile

Systems that present old and new information concurrently do not require their users to actively declare whether they are finding or re-finding, nor to interact only with new or old information. Such systems can help people quickly find information of interest by drawing the users attention to it. Systems designed to support interaction with dynamic information often do so by highlighting new information that has become available.

### 3.2.4 Personalized Ranking Agent

Personalized Ranking Agent is the decision making center of the proposed system based on multi-agent. After retrieving the results from search engine, similarity between the users updated profile P and the feature vector of the $i_{th}$ web page in search results is computed using cosine similarity formula. Based on the values obtained, the re-ranking is done using algorithm and by reorganizing the information satisfying the needs.

The proposed approach aims to effectively personalize search results according to each users information need by accurately identifying the user context, updating user profile. Using Re-ranking algorithm find the new score based on user interest. Personalization of web search involves adjusting search results for each user based on his or her unique interests. Since the purpose of current work is to achieve a personalized ordering of search engine results, the system can assign a page based score on the User Profile and the results returned by a search engine.

The clustering algorithm recursively partitions the words into smaller
clusters, which represent more related words. The assumption here is that words occurring close to each other (within a window size) are related to each other. Clustering algorithm recursively divides clusters into child clusters until it meets the stopping conditions. Before clustering extract words from WebPages that are interested to the user are filter them through a stop list, and stem them.

In order to provide personalized, it is necessary to score each page depending on personal interests. Therefore, the goal is to assign higher scores to WebPages that a user finds more interesting. This section explains how to score a retrieved WebPage using a users profile. First, explain the basic characteristics for each matching term. Second, based on the characteristics, new system propose functions to score a term. These functions determine how interesting a term is to a user. Third, based on the score and the number of the matching terms, the system calculate an overall score for the page. Last, since the search engine provides a score/ranking for a WebPage, new system incorporate this ranking into the final score of the WebPage.

3.2.4.1 Characteristics of a Term

Given a WebPage and a user profile, first identify matching terms (words/phrases) that reside both in the WebPage and in the user profile. The number of matching terms is defined m, which is less than the number of total distinct terms in the WebPage, n, and the number of total distinct terms in the user profile. Each matching term, $t_i$, is analyzed according to four characteristics: the depth level of a profile node where a term belongs to $D_{t_i}$, the length of a term such as how many words are in the term $L_{t_i}$, the frequency of a term $F_{t_i}$, and the emphasis of a term $E_{t_i}$. D and L can be calculated while building a user model from the WebPages. Different WebPage has different values for $F_{t_i}$ and $E_{t_i}$ characteristics.
3.2.4.2 Level/depth of a profile Node

\((Dt_i)\) A User profile represents general interests in large clusters of terms near the root of the User interest hierarchy, while more specific interests are represented by smaller clusters of terms near the leaves. The root node contains all distinct terms and the leaf nodes contain small groups of terms that represent more specific interests.

3.2.4.3 Length of a Term

\((Lt_i)\) Longer terms (phrases) are more specific than shorter ones. If a WebPage contains a long search term typed in by a user, the WebPage is more likely what the user was looking for.

3.2.4.4 Frequency of a Term

\((Ft_i)\) More frequent terms are more significant/important than less frequent terms. Frequent terms are often used for document clustering or Information Retrieval. A document that contains a search term many times will be more related to a user’s interest than a document that has the term only once.

3.2.4.5 Emphasis of a Term

\((Dt_i)\) Some terms have different formatting (HTML tags) such as title, bold, or italic. These specially-formatted terms have more emphasis in the page than those that are formatted normally. A document that emphasizes a search term as a bold format will be more related to the search term than a document that has the term in a normal format without emphasis. If a term is emphasized by the use of two or more types of special formatting, a priority is assign in the order of title, bold, and italic.

3.2.4.6 Scoring a Page

The personalized page scoring function for a WebPage \(S\) adds all the scores of the terms in the WebPage and can be formulated as:

\[
Sp_j = \sum_{i=1}^{m} St_i
\]

Where \(m\) is the total number of matching terms in a WebPage and \(St_i\) is the score for
each distinct term. The time complexity of scoring a page is $O(n)$, where $n$ is the $t_i$ number of unique terms in a WebPage.

3.2.4.7 Incorporating Public Page Score

Personal page scoring is not sufficient for some search engines. The success of using public scoring in popular search engines, such as Google’s PageRank, indicates the importance of using a public page popularity measure to determine what page a user is interested in. Many existing methods determine the public popularity of a page by determining the number of pages that link to it.

Many collaborative filtering approaches also use the popularity of a WebPage for recommendation in order to incorporate the public scoring into our page scoring function so both the popularity of a page and individual interests are taken into account. The new system uses the rank order returned by Google as public score.

$GOOGLE_{SP_j}$ is the score of a WebPage $p$ based on the page rank returned by Google for a search term. Users tend to find the answers they are looking for with Google, so it is decide to use the Google’s rank as the public page score. The use of Google’s page rank makes our experimental comparison with Google clearer, because any improvement in the ordering is due to the contribution of our personal page score.

For a given WebPage, the personal and public page score (PPS) equation can be written as:

$$PPS_{SP_j} = c \times R(SP_j) + (1 - c) \times R(SE_{SP_j})$$

where function $R(SE_{SP_j})$ return the rank of a WebPage $P_j$ with the public page score of $SE_{SP_j}$ and $R(SP_j)$ is the rank of a WebPage $P_j$, with a personal page score $SP_j$. If the function $R$ returns the rank in an ascending order, more interesting WebPages will have lower PPS values. Therefore, the function $R$ reverses the rank. The personal page score and the public page score are weighted by the value of the Constant $c$. In this paper, both functions are weighed equally: $c = 0.8$. The performance of proposed may depend on how much we weigh the personal page score over the public page score. The experiment with various $c$ will be future work.

<table>
<thead>
<tr>
<th>algorithm</th>
<th>for finding the term count</th>
</tr>
</thead>
</table>

Input:
Document $T$ and search terms

Output:
total number of search terms
Parameter used:
s - sum of weight
D - document used

termsum(D,$T_i$)
sum=0
j=0

For each term in D
{
if \((T_j = T_i)\) then
{
sum=sum+1
j=j+1
}
}

Algorithm for calculation of relevance score (re-ranking) for the WebPages

Input:
WebPage (P) Query term t1, t1, t3, t1n from the expanded query
Output:
relevance score (ranking) for all the WebPage (WP)

Urlset=ur1, ur2, urn for the given query

Parameter used
TF - term frequency
TF1 - term frequency for correct match n-total number of terms in the query
Qex-semantic query
url-url set
WP-WebPage
tc-view time

Re-ranking(urlset ,Qex)
{
  if Qex match urlset then
    \[ TF1 = TF1 + \text{termsum}(D, T) \]
  For each term Ti in the semantic query
  {
    if Ti match urlset then
      \[ TF = TF + \text{termsum}(D, T) \]
    wcount=TF/N
    Wc=wc+TF1+wcount
    Fs=Get feedback score{0,1,-1}
    ts=viewtime(WP)
    Save fs+wc+ts in DB table
  }
}
3.2.5 Knowledge Base

Knowledge Base is used for storing every user interest model, user record, and rules or parameters that serve for ensuring system well-balanced circulation. When user browses output documents, the system memorizes users behaviour (browsing, saving etc.) to Knowledge Base in real time. The system may give an evaluation page for asking user to do satisfactory degree. Evaluating all documents, which satisfactory degree $SatisDe(D)$ is more than a default minimal value, (threshold value) is extracted for constructing new user interest domain vector.

New user interest domain vector is used to replace old user interest domain vector, which is cited seldom. The storage capacity of user interest model is commonly limited to finite space capability, for example: $NMAXTIME$. When the number of user interest domain vector exceeds the capability limit (greater than or equal to particular time interval); some user interest domain vectors, which are cited seldom (scaling by domain interest degree function $Fi(fw, rw, iw)$), may be deleted from user interest model and moved to dump table. So the number of user interest domain vectors is limited to definite scope, and the system can track user interest in time.

3.3 Performance Analysis

In this section experiments are carried out to evaluate the performance of proposed system will be discussed from a quantitative point of view by running some experiments to evaluate the precision of the results. The basic idea of the experiment is to compare the search result from keyword based search engine with proposed one on the same category and the same keywords. The criteria of new system is to include suitability (the ratio of the amount of useful information to the total amount of information) age (the period of the document post) and semantic matching (the accuracy of
matching). After several time of similar information search, proposed system will get better results than the current search engine expectedly by updating user profile based on the user feedback autonomously. A test set collection is which consists of set of documents, queries and a list of relevance documents are used to evaluate the proposed system. These are used to compare the results of new system by performing relevance based evaluation method.

The proposed system is implemented in C#.Net as Web-based system using Visual Studio 2008,.NET Framework 3.5, and SQL Server 2005. The number of stored documents is more than 3 lakhs documents. These Web documents are from various science domain such as computer science, physics, Mathematics domain. The improvement is measured by performing different experiments using the relevance based evaluation method. It uses the metrics: precision, recall, f-measure, average precession (AP) and mean average precision (MAP), to measure the performance of proposed system.

### 3.3.1 Measures used

#### 3.3.1.1 Precision

Measure of how much relevant information the system has extracted (coverage of system). Precision is the ability to retrieve top-ranked documents that are mostly relevant.

\[
\text{Precision} = \frac{\text{number of relevant links given by the system}}{\text{Total number of relevant links}} \quad (3.5)
\]

#### 3.3.1.2 Recall

Recall is the ability of the search to find all of the relevant items in the corpus. The Recall calculated here is the relative recall in which performance
Table 3.3: Parameters used in the analysis used

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Number of URLs</td>
</tr>
<tr>
<td>N</td>
<td>Number of terms</td>
</tr>
<tr>
<td>U</td>
<td>Different Users</td>
</tr>
<tr>
<td>K</td>
<td>Semantic Query word</td>
</tr>
<tr>
<td>Fs</td>
<td>user Feedback</td>
</tr>
<tr>
<td>ts</td>
<td>time spend on view a page</td>
</tr>
<tr>
<td>WP</td>
<td>Web page</td>
</tr>
<tr>
<td>Wcount</td>
<td>word count in each WP</td>
</tr>
<tr>
<td>P</td>
<td>Precision</td>
</tr>
<tr>
<td>R</td>
<td>Recall</td>
</tr>
<tr>
<td>AP</td>
<td>Average precision</td>
</tr>
<tr>
<td>TF</td>
<td>term frequency</td>
</tr>
</tbody>
</table>

is compared with the Google search engine.

\[
\text{recall} = \frac{\text{number of relevant links given by the system}}{\text{Total number of links retrieved}}
\]  (3.6)

**3.3.1.3 F-Measure**

F-Measure is the Harmonic mean of recall and precision.

\[
F = \frac{2PR}{P + R} = \frac{2}{\frac{1}{R} + \frac{1}{P}}
\]  (3.7)

The various parameters used for analysis of this system is listed out the Table 3.3. A set of queries was prepared manually for comparative performance measurement. The set of sample queries is given in Table 3.4 and Table 3.5. It shows the different levels of performance for different queries. The proposed semantic Information Retrieval method that improves the searching characteristic.

Figure 3.8 shows comparative study of the results of the both systems that retrieves the documents based on similarity between the query and the collected documents. This experiment shows the average precision that is
<table>
<thead>
<tr>
<th>keyword</th>
<th>keyword</th>
<th>semantic</th>
<th>keyword</th>
<th>semantic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>0.76</td>
<td>0.56</td>
<td>0.61</td>
<td>0.41</td>
</tr>
<tr>
<td>Constructor</td>
<td>1.00</td>
<td>0.81</td>
<td>0.69</td>
<td>0.52</td>
</tr>
<tr>
<td>Polymorphism</td>
<td>1.17</td>
<td>1.00</td>
<td>0.89</td>
<td>0.67</td>
</tr>
<tr>
<td>memory compaction</td>
<td>1.1</td>
<td>0.79</td>
<td>0.58</td>
<td>0.53</td>
</tr>
<tr>
<td>Encapsulation</td>
<td>1.3</td>
<td>0.79</td>
<td>0.69</td>
<td>0.53</td>
</tr>
<tr>
<td>disk space management</td>
<td>1.20</td>
<td>.79</td>
<td>0.76</td>
<td>0.53</td>
</tr>
<tr>
<td>abstract classes</td>
<td>0.98</td>
<td>0.84</td>
<td>0.89</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 3.4: Different level of query
Figure 3.8: AP measure precision of Personalized vs Un-Personalized (for single user)

Figure 3.9: AP measure precision of Personalized vs Un-Personalized (for multiple user with multiple keyword)

based on retrieving results for different query of single user and single query of multiple users. Graph shows that the system gives high precision during retrieving documents.

Figure 3.9 shows the Average Precision measure precision of Personalized vs un personalized (for multiple user with multiple keyword). The retrieval efficiency is a major challenge when the size of the database increases. This
shows the importance of semantic similarity during determining the documents that are relevant to the user query. The second sets of experiments, which are user centred, are focused on the overall performance of the search engine and the evaluation of real interactions with users. Figure 3.10 discusses the performance efficiency of both system when the system uses to retrieve the result. This graph shows that agent based personalized search is better than other method because it use user profile and study user behaviour to determine ranking for each time. It can also be observed that the contextualization technique consistently results in better performance with respect to simple personalization, as can be seen in the average precision and recall depict by Figure 3.9 which shows the average PR results over the different user cases.

The next experiment aims at determining the importance of personalization by using generated dynamic user model during using the system. The user model is used to re-rank the retrieved documents to match the user interest. Personalization time: Time to retrieve any information depends on the type of search engine, size of data set, relevancy between query and document.
user history and re-ranking algorithm used.

The **Personalization Performance** can be expressed:  
\[
\text{Personalization performance} = \sum_{i=1}^{n} = \text{Grank} + \text{UserRank}
\]
and For each page find  
\[
\text{UseRank} = \sum_{i=1}^{n} = \text{UR} + \text{VT} + \text{FC}
\]
Where \(\text{UR}\) - user rating \(\text{VT}\)-page view time and \(\text{FC}\) -frequency count and \(n\) represents threshold value.

![Efficiency Measure](image)

**Figure 3.11: Efficiency Measure**

The usage efficiency of the systems when the system uses to retrieve the result is discussed in Figure 3.11. It is observed that 80% users, have found improved precision with the proposed approach in comparison to the standard search engine (Google) results, while 20% users have achieved equal precision with both approaches. It has been observed that users who posed Queries in unpopular context than popular context got better performance. In addition, when the system can extract the exact context of users need, the Precision and recall is found better than other search engine results.

### 3.3.2 Discussion

The detailed description of proposed ABPSWIR architecture is given in this chapter. Experiment 1- compare the result of retrieval documents between agents based system and existing one based on single user. This experiments
shows the average precision that is based on retrieving results for different query of single user. The precision is improved 75% over keyword based systems. Experiment 2- compare the result of retrieval documents between agents based system and existing one based on multiple user with sample number of query. This experiment shows the average precision that is based on retrieving results. The precision is improved 80% over keyword based systems. Experiment 3- Measuring performance efficiency of the both systems over the different user and single user. The experiment shows that agent based personalized search is better. The time to retrieve semantic results is very low than keyword based one. Experiment 4- Measuring usage efficiency of the proposed systems. It observed that 80% users have found improved precision with the new systems in comparison with the keyword based system. In sum, the overall precision is improved 25%-50%.

The next chapter discusses the various applications of the proposed systems and their performance.