User Preference Modeling - A Survey Report

Hamza Hydri Syed, Periklis Andritsos

August 2007

Technical Report # DIT-07-060

User Preference Modeling - A Survey Report

Hamza H. Syed, Periklis Andritsos
Department of Information and Communication Technology
University of Trento, Italy
{syed, periklis} @dit.unitn.it

1 Introduction

Traditionally Information Retrieval systems and Search Engines have been following a *"One-View of World"* approach when it comes to handling user queries. They do not differentiate between their users. The documents retrieved for a given keyword query would be similar for any user at any given instant of time. The ranking schemes used were based on the popularity of the web-pages, derived from the hits/clicks of the page.(*aka* PageRank [7]). A recent survey[1] claims that there are over 100 million websites hosted currently on the Internet. With this tremendous growth of the documents lying scattered all over, there is a causal need for a change in the Information Retrieval and Web Searching strategies. We observe that there is a prominent shift in the approach towards retrieving the needed information, the type of data being searched for, the size of databases and document collections being used. More importantly, the number of end-users relying on search engines for their information needs has grown rapidly in size and variability. Most of the prominent search engines (Ask, Google, Yahoo Search, Windows Live) have begun providing an option of personalised search through logging in, entering parameters of *advanced search* and providing explicit preferences on their web search portals.

In the recent years there has been a steady growth towards personalization of search systems and implementing context to improve the effectiveness of a search engine. Database and IR systems have realised the importance of understanding the end-users and modeling their search behavior, to improve the effectiveness of their systems. Personalization of search systems can be implemented in a variety of ways and its related research has focussed in several approaches like - implicitly monitoring user behaviour, mining of user interests from her web activites, building and using user profiles to improve the relevancy of search results either by pre-processing the queried terms or post processing the retrieved results. For pre-processing of the queried terms, several techniques are proposed to re-write the query to reflect the user's preferences, while for post-processing of results, several methods have been suggested to re-rank or display the results based on some relevancy criteria which the system learns from the user over a period of time.

In this report we present an overview of the work being done on modeling of User Preferences. In the following sections, we try to summarize the work done in the domain of Personalised Search across the communities of Databases, Information Retrieval and Web Search along the following orthogonal criteria:

- **User Profiles**
  - *Capturing and Storage:* We shall discuss how the system constructs and stores the user profile. We shall also discuss the methods used to capture the user's interests.

[1]http://news.netcraft.com/archives/2007/07/09/july_2007_web_server_survey.html

  - *Architecture:* We shall look into the architecture / representation of these profiles.

- **Ranking / Filtering scheme** Here we would discuss any ranking function or filtering algorithm used by the system.

- **Evaluation Metrics** If there is an evaluation of these systems, then we shall look into the metrics used and discuss the results of these evaluations.

1.1 Filtering Information

In [3], Belkin and Croft claim that Information Retrieval and Filtering are the same in reality, and discuss that by capturing user interests the system can present interesting information to specific users. In one of their conclusions, the authors state that, any research involving filtering of information, needs to study in detail the various dimensions of user's information interests - what they might be, how to identify them, how to represent and modify them.

The **Stanford Information Filtering Tool (SIFT)** [41] provides a large-scale information dissemination service. The user subscribes by submitting one or more profiles that describe her interests along with other control parameters. The system then filters the articles for the user by comparing to these profiles.

Profile - Creation, Storage and Architecture: The user subscribes to the system by submitting one or more profiles. The profiles include a query and parameters like - the frequency of notification, amount of information to receive and time duration of profile. The profiles are differentiated using the user's e-mail address and a profile identifier. The user can interactively modify the profile by adjusting the threshold levels of the desired precision and recall values of the retrieved results. The user uses either a Boolean or a Vector Space Model for specifying her interests and can manually specify a *relevance threshold*, which is the minimum similarity score that a document should have in order to be delivered.

Ranking / Filtering scheme: The similarity between the documents and the desired interests is calculated using the *cosine similarity* measure. For a given document its `SCORE` is accumulated for all the words of user's interest and when it crosses the user defined `THRESHOLD` value, the document is retrieved.

Evaluation: The system performance is evaluated based on the change of the processing time with increasing number of documents and number of profiles. It is observed that the profile building time is very small, while the filtering time is linearly related to the number of documents. The notify time increases more rapidly than the other processing times and is proportional to the number of matchings in a matched document. The processing time performs similarly with the increasing number of profiles i.e. the profile index build time is independent of the number of profiles, while the filter time increases linearly and the notify time increases polynomially with number of profiles. On comparing with alternative document index methods the filtering methods are observed to perform better.

A major problem in many information filtering systems is the dependence on the user for the creation of a user profile, **NewsWeeder** [25] addresses this problem by letting the user rate her interest level for each article she reads, and then learning a user profile based on these ratings. It uses both *content-based-filtering* (determining relevance from the article text) and *collaborative-filtering* (using ratings from earlier readers to predict the rating for the later readers). It collects these ratings as an active feedback of the user's interests to construct a '*bag-of-words*' model of the user's interests. Using machine learning techniques for the user's profile and the past ratings, this model predicts the score for a news article.

Profile - Creation, Storage and Architecture: The user ranks the news articles into one of the 5 categories (*essential / interesting / borderline / boring / gong*) and this active feedback is used as training examples for machine learning routines which are invoked to create a user profile each night. The profile is a vector of tokens, which are weighted using TF-IDF schemes. It is a flat structure with no hierarchy and the profile is stored at a central location.

Ranking / Filtering scheme: The *cosine similarity* measure and Minimum Description Length (MDL) are used to compare the similarity of new articles with the ranking categories in the user profile. The machine learning component then assigns a predicted rating to the article.

Evaluation: The effect of removing the N-most frequently occurring words on the precision values of the retrieved results is studied. It is observed that the precision initially increases gradually and attains a stable value as we increase the number of stop words being filtered. A comparison is done on the performance of the Minimum Description Length (MDL) approach and the TF-IDF approach under the influence of increasing number of training samples. The precision performance of MDL is better than the TF-IDF approach.

PSUN - The Profiling System for Usenet News (PSUN) is an information filter for filtering articles from the Usenet News, which is a large collection of discussion groups covering wide range of topics related to computer science, social issues, etc. This filter is modeled as an interface agent which acts on behalf of the user, encapsulating her interests, accepting feedback from the user on the relevance of the filtered documents and adjusting the user goals based on this feedback [37].

Profile - Creation, Storage and Architecture: The user profile is constructed by initially presenting a set of documents the user finds interesting. The recurring words and phrases in these documents are captured and these terms are structurally represented as n-grams, which are word sequences of length n. Each of these n-grams have a weight associated with them, which shows its importance as compared to other n-grams in the user's profile. This weight is an appraisal of the co-occurance of these words in a document. These n-grams are stored as a network of mutually attracting or repelling words, where the degree of attraction is determined from the frequency of occurance of the words.

Ranking/Filtering: The textual content of the incoming articles is processed and then searched for the relevant phrases. When the phrases are found, then the supervisors (autonomous agents) modify the overall rating of the article and the final ratings are used to rank the documents before displaying them to the user.

Evaluation: The authors have not discussed about any evaluations comparing their system with the then contemporary systems.

There have been several solutions to achieve personalized information filtering using the *Software Agent Technology*. Let us have a brief look into some of the major referenced works.

SmartPush - In this project, professional editors add semantic metadata to the information flow, which can be used for filtering anf providing personalized news service to the users. The articles are enriched with semantic metadata, which is expressive to facilitate the personalization process and eliminates the need to transfer the full length content. The system employs software agents for the task of managing the user profiles, matching incoming metadata records and delivering information to the user [24].

Profile - Creation, Storage and Architecture: The initial profile is created by the user either by ranking a set of sample documents and the system derives the profile information from them. Alternately it can be created by providing explicit keywords to describe a profile or by selecting one or more prototype profiles.

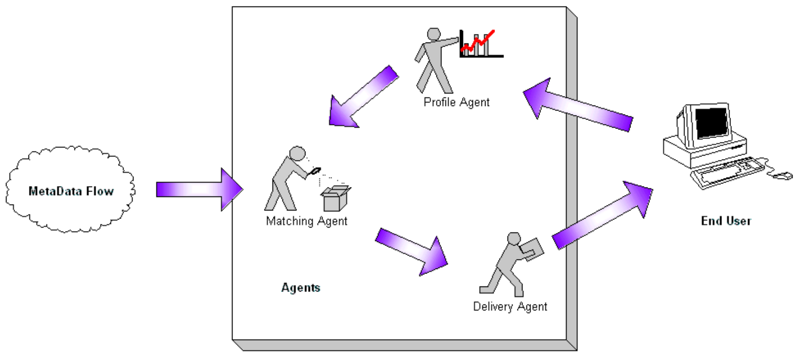

Figure 1: Agents interaction in the SmartPush System

The profile is modeled as a concept hierarchy or an ontology, and is stored in the network hosted by an information broker. This provides independence of location, available bandwidth and device.

Ranking / Filtering scheme: The matching process is carried out by the *matching agent*, whose task is to compare the incoming meta data records against the base of registered profiles. The comparisons are based on distance measure that calculates how far the document is from each profile.

NewT - This news filtering system employs software agents which act as an interface between the user and the system. The software agents have knowledge of the user's interests and perform keyword based automated filtering of news articles. The system makes use of relevance feedback and genetic learning algorithms to adapt and explore new types of information of the user's interests. [36]

Profile - Creation, Storage and Architecture: The initial profile is provided by the user in the form of listed keywords. Whenever a new article is displayed to the user, her feedback is obtained explicitly and this feedback is used to modify the profile. The profile consists of a number of weighted *fields* (like *author, newsgroup, keyword* etc.) and each of these fields is a vector of terms. The terms are weighted depending on their importance for identification purposes. A collection of these profiles is instantiated as an agent, which performs the task of filtering and suggesting documents to the user. A profile could be represented as

$$P = (F_i^p, W(F_i^p))$$

where $W(F_i^p)$ gives the weight of F_i^p in the profile P.

Ranking / Filtering scheme: The documents are also represented in the form of vectors similar to the user profiles. The *Cosine Similarity* measure (scalar product of two vectors) [35] is used to measure the similarity between the document and user profile vectors. Some approaches have been discussed for the selection of the filtered documents, for example documents which score above a pre-defined threshold are displayed to the user.

Evaluation: The system is evaluated with real users for a period of two weeks and the feedback on their experience is collected. The users reported positively for the user interface but had some reservations regarding the functioning of the software agents. Evaluation with simulated users under specific scenarios, reveals that the NewT system can be personalized to serve some of the predictable news filtering needs of the user and explicit relevance feedback can be used to learn the specialized user interests.

Alipes - This is an Information Brokering Agent, that learns the user's interests and provides personalized news articles filtered from the Internet [40]. It is a multi-agent system, with several agents carrying out specific tasks such as, learning the user's interests, interfacing with the user, coordinating and searching activities. This system models the user's profile in the form of multiple keyword vectors for each interest.

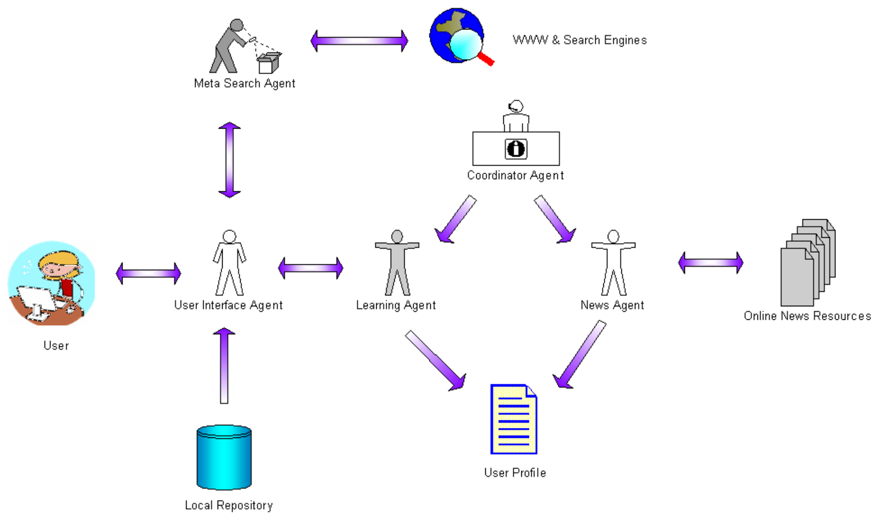

Figure 2: Agents interaction in Alipes

Profile - Creation, Storage and Architecture: Each interest of the user is modeled by three vectors - long term, short term (positive) and short term (negative). These descriptors are learned from the feedback collected from the user. They also consider negative feedback to record the user's disinterest. For adding new interest similarity measurement is done with existing interest vectors.

Ranking / Filtering Scheme: The document feature vectors (created using the TF-IDF method), the document title and some sample content is used to create a meta-document. This information is compared with the profile descriptors and the top 30 scored documents are ranked in decreasing order of the scores.

Evaluation: Experimental Evaluation is carried out on 28 artificially constructed profiles to simulate artificial users. A document collection of HTML pages (1427 documents consisting of several topics like world news, sports, weather, etc.) from various news sources was collected and used for testing the retrieval efficiency at various threshold levels. The system performance is reported to be degrading on increasing the threshold values.

WebMate [10] is a personal software agent that helps the users in browsing and searching the Internet effectively. Its architecture is similar to the ones described above, the software agents intermediate on behalf of the user. The system makes use of multiple TF-IDF vectors to keep track of user interests in different domains. It uses the Trigger Pair Model, to automatically extract keywords for refining the document search. In this system the user can provide multiple pages as a relevance criteria for the search. The system extracts keywords from these given pages and uses them for keyword expansion (query refinement) and relevance feedback to improve the search.

Profile - Creation, Storage and Architecture: The system stores multiple keyword vectors, each representing a different user interest. The system uses learning algorithms to learn a set of interests from the positively rated pages. Users provide explicit feedback to the webpages they browse and document vectors are created from these pages using the TF-IDF weighting scheme.

Ranking / Filtering scheme: *Cosine Similarity* measure is used to compare the document vectors (created after parsing the webpages) and the keyword vectors from the user's profile. If their similarity is higher than a threshold value, the system recommends these webpages in decreasing order of similarity.

Evaluation: In the evaluation experiments, websites containing news articles are monitored and personalized newspapers are generated based on a set of user profiles. It is observed that the accuracy of the recommended articles increases by a factor of three.

System	Profile - Creation, Storage & Architecture	Ranking / Filtering	Evaluation
SIFT	flat-nature profile (query & control parameters) submitted by user, stored on the server	`SCORE` of a document calculated, if greater than `THRESHOLD` document is retrieved	processing times w.r.t no. of documents & no. of profiles
NewsWeeder	active feedback used to create profile, is a flat-structure, weighted vector of tokens, stored centrally	*cosine similarity* measure used to filter out documents	effect of stop words on precision of results, MDL precision results compared to TF-IDF
PSUN	profile stored as an n-gram of terms	documents are ranked based on occurrence of phrases	not described
SmartPush	many options to create profile - user submits explicit keywords, selects one or more prototypes, systems learns the profile from user-ranked documents	software agents rate documents based on similarity with profiles	not described
NewT	initially submitted by user, modified through explicit feedback, profile is a weighted vector of terms, instantiated as software agents, stored centrally	*cosine similarity* measure used to rate documents, filtered based on pre-defined threshold values	performance evaluations with real and simulated users
Alipes	vectors describing short term and long term interests are created / modified from user's explicit feedback on viewed documents and stored centrally	profile vectors compared with document meta data and sorted based on certain threshold	retrieval efficiency of the system evaluated with artificially constructed profiles on a document collection
WebMate	multiple keyword vectors representing different user interests, learning algorithms used to learn interests from positively rated documents	*cosine similarity* measure used to sort the documents	relevancy measurement on personalized news articles generated from a set of news websites

Table 1: Information Filtering Systems

1.2 Recommender Systems

Recommender Systems suggest the user to visit other related webpages based on the webpage currently being viewed. The performance of these systems depends heavily on the modeling of user's interests in order to make precise judgements about the user's needs and recommend options (websites, products, documents etc) to her. The user interests are captured through the feedback generated from implicit or explicit indicators such as bookmarking / saving / printing / emailing a page, scrolling through or searching through the text of a page, navigating through the hyperlinks from a particular page or recurring visits to a particular page.

GroupLens - In this project, a collaborative filtering approach is implemented to recommend personally relevant news articles for a particular user from the Usenet newsgroups [33]. Collaborative Filtering is a kind of social filtering process in which articles are selected based on the opinions or feedback obtained from other users.

Profile - Creation, Storage and Architecture: As the user reads the articles in the newsgroup, the news reader records the ratings entered explicitly by the client (user) and sends them to the server. The server uses these ratings to make predictions to other users about this article. The various newsgroup servers share the ratings in order to predict the interests of users across various newsgroups.

Ranking / Filtering scheme: The Pearson Correlation Coefficient is used for computing similarity of a given active user to the other existing users. In other words, it predicts how much a particular user appeared to agree with the other users who rated the same articles using the formula

$$\rho = \frac{\mathtt{covar}(r_a, r_u)}{\sigma_{r_a}\sigma_{r_u}},$$

where r_a is the rating vector for the currently active user and r_u is the rating vector for the other existing user. k-NN method is used to select the appropriate neighbourhood for the current active user to make predictions. The weighted average for the given user along with other existing users is used to make the predictions.

Evaluation: The evaluation of the system is not discussed but the experimental studies reveal that the predictions behave differently for different newsgroups. The difference between average and personalized predictions varies depending on the degree of correlation between the users. It is also observed that users inclined to give ratings to the articles for which they have recieved predictions as compared to those articles without any predictions.

Claypool et al describe a set of experiments to capture the user interests in [12]. They categorize the interest indicators based on user actions and structure of content and statistically analyze their correlation with explicit interest. The **Curious Browser** is used to capture the user actions as they browse through the web.

Profile - Creation, Storage and Architecture: A number of parameters like number of mouse clicks, mouse wheel activity, time spent scrolling, size of the file, keyboard activity etc. are used to capture the user interests implicitly. This information is stored locally on the client side, in a flat structure.

Evaluation: The experiments revealed that the interest indicators matched very closely with explicit ratings of interest. A positive relationship is observed between interest indicators like mouse activity (scrolling, clicking, etc.) and explicit ratings.

The **SiteSeer** system [34] makes use of the files and their order of appearance (organization) in the bookmarks of the user's web browser as an indication of the user's interests. It uses this information for predicting the user's needs and recommending appropriate relevant webpages.

Profile - Creation, Storage and Architecture: The system views the bookmark folders as a personal classification system which enables it to contextualize recommendations and interprets the user's grouping behavior as an indication of semantic coherency.

Ranking / Filtering scheme: It recommends pages that have been bookmarked by the user's virtual neighbors. Pages appearing in higher number of overlapping folders of the neighbors are given higher preference.

Personalized CiteSeer: Bollacker et al [5] track and recommend relevant papers in their CiteSeer system by using a heterogeneous profile to represent the user interests. The algorithm utilizes a cookie embedded in the Web Browser interface to monitor the user behavior and use the keywords of the visited papers to recommend relevant papers. The profile is displayed to the user, allowing him to modify the tracked details.

Profile - Creation, Storage and Architecture: The user profile is created during the user's browsing and searching actions while using CiteSeer, by utilizing the cookie embedded in the user's Web Browser. The cookie assigns a unique identification number to the user to keep track of her actions, update her profile and make recommendations. The profile creation process is initiated when the user specifies either a citation for tracking, keyword, documents or a URL to add to the profile. On selecting one of the several options presented to the user at the interface of the search results (Context Link, Related Documents Link etc.), the user can update and modify her profile to better reflect her current interests.

Ranking / Filtering scheme: The system employs Constraint matching (keyword matching and meta-data tags) and Feature relatedness (text and citations) to determine the relevancy of a particular document (scientific publication in this case) to the user's interests. The *term frequency - inverse document frequency* (TF-IDF) [35] distance between the abstracts and text bodies of papers determines whether they are related to the papers specified by the user.

Amalthaea: It is an Agent-based information filtering system that provides personalized information to the users from the World Wide Web or provide news filtering service [29]. The Amalthaea architecture is a closed eco-system as shown in the figure with each user having their own Information Filtering and Information Discovery Agents.

Profile - Creation, Storage and Architecture: The user profile is created in form of a single keyword vector by extracting keywords from the web pages visited by the user. These keywords are then weighted using the TF-IDF weighting scheme [35]. The user profile is essentially a weighted keyword vector of terms and can be created by the user explicitly providing the terms. The system also allows the user to explicitly specify their profiles, which is prioritized higher than the one created by the system.

Ranking / Filtering scheme: The Information Discovery Agents (IDA) interact with the search engines and search documents based on the keywords provided by the Information Filtering Agents (IFA). The retrieved documents are converted into weighted keyword vectors and presented to the IFA which requested for it. The IFAs compare these vectors with the user profile vectors using *cosine-similarity* measure [35] determined by the following formula.

$$D_{IFA_{a,b}} = \frac{\sum_{k=1}^{j} w_{ak} * w_{bk}}{\sqrt{\sum_{k=1}^{j} (w_{ak})^2 * \sum_{k=1}^{j} (w_{bk})^2}}$$

The documents which match closely to the user's interests are filtered and presented to the user.

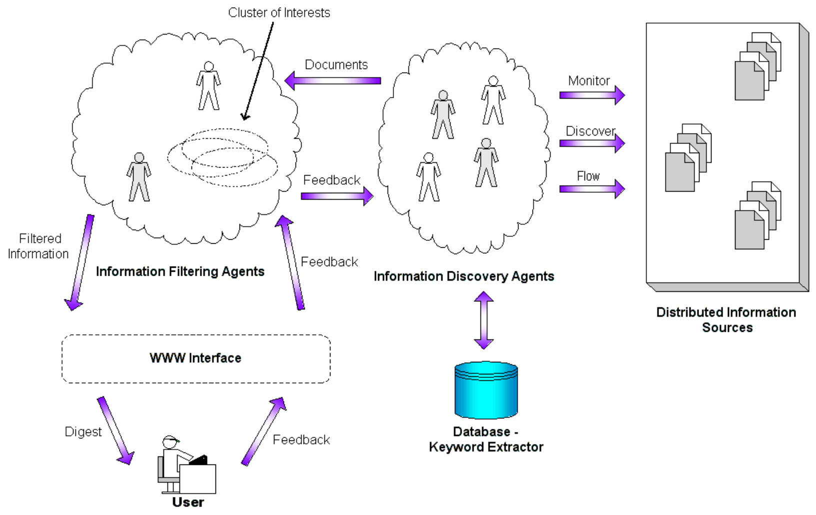
Cluster of Interests
Documents
Feedback
Monitor
Discover
Flow
Filtered Information
Information Filtering Agents
Feedback
Information Discovery Agents
Distributed Information Sources
WWW Interface
Database - Keyword Extractor
Digest
Feedback
User

Figure 3: Amalthaea Architecture

Evaluation: In the first set of experiments profiles were created for virtual users to test the scalability of the system. The system performance was evaluated when the user interests were constant or changing and under the conditions when there is no feedback from the user. In the second set of experiments real users were involved. Their interests were captured explicitly and the system performance was evaluated on the criteria of precision of the retrieved results, their correlation with the user's explicit ratings etc.

ifWeb: The ifWeb system consists of an intelligent agent which supports the user while browsing the WWW, by retrieving and filtering the documents taking into account the user's specific information needs [2].

Profile - Creation, Storage and Architecture: This system models the user profile as a weighted semantic network, where each node represents a keyword of user's interest and the arcs represent their co-occurrences across the documents. The profile is initially constructed by collecting the user's feedback (positive and negative) on a set of documents. The profile is then refined by the browsing agents as the user browses the web, by collecting feedback on the documents recommended by the ifWeb system. Keywords extracted from the pages for which the user has provided a feedback are processed and added to or updated in the user's profile.

Ranking / Filtering scheme: There is not much description provided about the filtering scheme apart from the approach that the browsed documents are compared to the user model and rated either as "interesting", "not-interesting" or "indifferent".

Evaluation: User evaluations were carried out to assess the quality of navigation and learning capability provided by the ifWeb system. Experiments were done to evaluate its document classification and the ifWeb system was compared to the traditional search engines.

SiteIF: The SiteIF system [38] utilizes the user's browsing behavior to recommend the interesting web pages for a given user. It employs software agents, which follow the user from page to page and learn the user's interests from the items viewed by her. This information is used to generate or update the user model and retrieve documents of interest for the user.

Profile - Creation, Storage and Architecture: The system does not involve the user in the learning process, instead it observes the user's browsing behaviour and takes into account the addresses of the websites visited by her. SiteIF models the user profile as a semantic network in which the nodes are called as *synsets* (synonym sets, which are a group of related words formed by referring to the WordNet) and the arcs are the co-occurrence relation of two words. The nodes and arcs have weights, which keep changing depending on the user's interest, which keep modifying over a period of time.

Ranking / Filtering scheme: During the filtering phase, the matching module of the agent compares the internal representation of the document with the user's current model and classifies the document as *'interesting'* or *'not interesting'* for the user. It uses the standard keyword matching technique i.e. comparing the occurrences of the keyword in the text of the document. They also try to capture the context in which the keyword appears using the co-occurence relationship between the words.

Syskill & Webert: This software agent identifies interesting pages for a user from the World Wide Web based on her user profile. It learns this profile from the user's explicit ratings for each visited page. It uses the profile for suggesting further links which might be of interest to the user and to construct a personalized query (using the Lycos search engine) to retrieve pages of user's interest. The Syskill & Webert system makes use of graphical symbols to annotate the HTML pages retrieved, indicating the user's level of interest on their content [30].

Profile - Creation, Storage and Architecture: The profile creation is initiated by the user's explicit feedback when she rates the viewed pages on a 3-point interest scale (*hot, lukewarm, cold*). The various interests of the user are stored as classes, with no hierarchical relationships between them. Each class consists of a set of boolean feature vectors, whose terms are the informative keywords appearing in the webpages. An information-based approach similar to the one used in NewsWeeder is used to determine which keywords are to be used as features. This approach is based on the mutual information between the presence and absence of a word and the classification page.

Ranking / Filtering scheme: After comparing several machine learning algorithms, the Bayes Classifier is selected for classification purposes. The documents are rated by classifying them into different categories of user's interests. For all the documents in a particular class, the TF-IDF vectors are created and these vectors are averaged to create the prototype vector to represent that class. The *cosine similarity* measure [35] is used to compute the similarity with users interests.

Evaluation: Experimental studies were done to study the accuracy with which the various algorithms predicted the user's interests. The results suggest that the number of training examples has a positive effect on the accuracy of some of the algorithms while increasing or decreasing the number of features also has an effect on the accuracy of the learning algorithms.

WebWatcher: It is a '*tour guide agent*' for the web. It accompanies the user from page to page, providing assistance based on partial understanding of user's interests and content of the webpages. Its behavior is analogous to a museum guide giving a tour around the various sections of a museum, pointing to the interesting items for a given user. It learns from the experiences of multiple users to improve its advice-giving skills [15]. WebWatcher provides information about other user's behavior for a given page the current user is viewing and other features such as *"find similar pages"*.

Profile - Creation, Storage and Architecture: The user's interests are specified explicitly in the form of keywords, while begining to use this service. These interest keywords are treated as *"goals"* to be accomplished during a browsing session and after a page has been viewed the user is asked explicitly if her search goal is completed or not. These goals are stored at the server side. WebWatcher derives training examples from the browsing behaviour of past users, their stated interests, the hyperlinks viewed by them and the underlying words associated with these hyperlinks. It generalises these training examples to improve its ability to suggest appropriate hyperlinks for subsequent users.

Ranking / Filtering scheme: It uses metrics from the information retrieval field [35] to measure the similarity between the user's interests and the description of the hyperlinks from a given webpage. User interests and the hyperlink descriptions are represented as high-dimensional feature vectors, each dimension representing a particular word in the English language. TF-IDF is used to calculate the weights of terms in the feature vectors which are represented in a vector space. The *cosine similarity* measure is used to measure the similarity between the user interests and the document feature vectors.

Evaluation: The accuracy in learning and predicting the user's interests through various approaches was evaluated. Though the reinforcement learning approach performs better, a combination of the various methods (reinforcement learning, popularity of weblinks, TF-IDF *cosine similarity* measure between text and user interests) achieves highest accuracy.

Personal WebWatcher: It is also a agent-based recommender system which assists a user while browsing the World Wide Web, by suggesting pages that might be of interest to the user. It extends the WebWatcher by adapting its behavior to one user. It learns the user model from the pages viewed by the user and uses this model to suggest further pages or links to that user [28].

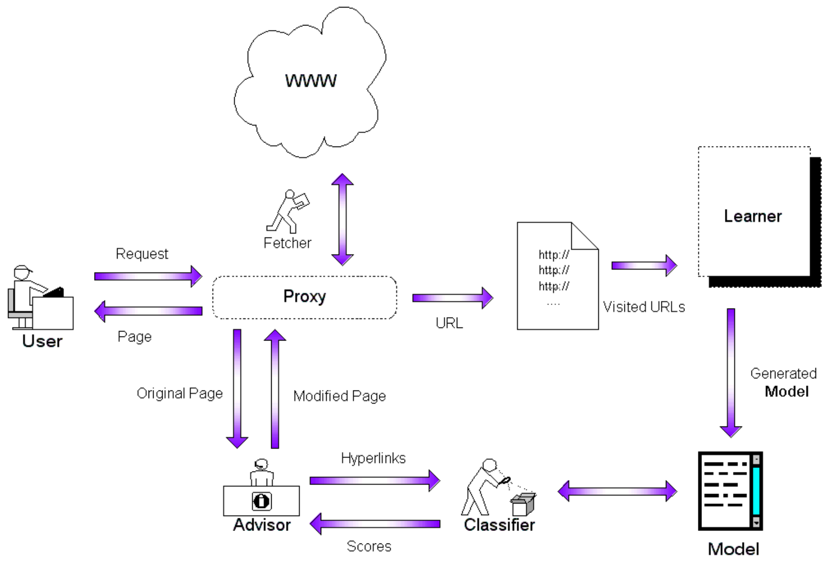

Figure 4: Personal Web Watcher System

Profile - Creation, Storage and Architecture: The system does not involve any input or feedback from the user to construct the user profile. It records the addresses of the pages viewed by the user and in the learning phase (overnight) it extracts the keywords from these pages to construct weighted feature vectors. The *bag-of-words* approach is used to construct the feature vectors for the documents, where the weights in the vectors are determined based on the TF-IDF metric [35]. The naive Bayes classifier is used to sort the documents into *interesting* and *uninteresting* classes. The user profile is represented as a set of these classes with their associated documents.

Ranking / Filtering scheme: The text from the documents is parsed and TF-IDF vectors are created for each of the documents. Using these document feature vectors, he naive Bayes Classifier is used to classify the documents into *interesting* and *not-interesting* classes.

Evaluation: The effect of vector length (number of features selected) on classification accuracies of traditional machine learning algorithms was studied. It is observed that k - Nearest Neighbour (k-NN) provides higher accuracy than the naive Bayesian approach. It is also observed that the precision values of k-NN approach increases with longer feature vectors.

Letizia: Similar to the WebWatcher, Letizia also accompanies the user while browsing, suggesting webpages based on the user's interests and similar to Personal WebWatcher it serves to only one user, learning her current interests. It is a user interface agent, which tracks the user behavior and predicts items of user's interests [26].

Profile - Creation, Storage and Architecture: A description of the user profile representation is missing, but since the documents are stored as weighted keyword vectors, it is assumed that the same approach is applied to store the user profiles as well. It relies on implicit feedback to capture the user's interests. It keeps track of the pages viewed and the bookmarking of the pages as an implicit indication of user's interest. This profile information is stored locally at the user side of the system unlike the WebWatcher system.

Ranking / Filtering scheme: The filtering strategies are not described, but since weighted keyword vectors are used, it can be assumed that *cosine similarity* measure of these vectors are used to filter interesting documents for a user.

System / Authors	Profile - Creation, Storage & Architecture	Ranking / Filtering	Evaluation
GroupLens	no profile stored, user ratings are stored	correlation coefficients used to measure similarity of a user with other existing users	system not evaluated, proof of concept discussed
Curious Browser	implicit indicators like mouse and keyboard activity used to capture user interests	not discussed	not discussed
SiteSeer	bookmarks in the user's browser inferred as an indication of user interests	priority given to webpages appearing in higher number of overlapping folders of virtual neighbours	not discussed
Personalized CiteSeer	cookie embedded in web browser tracks user activities and updates the profile, which is initiated by user specifying keywords and citations of interest	keywords & metadata tags used to filter documents, TF-IDF distance used to measure the similarity of textual content with user interests	not described
Amalthaea	profiles represented as keyword vectors, extracted from the viewed pages or entered specifically by the user, stored centrally	agents filter documents based on keywords, *cosine similarity* measure for calculating relevancy	system performance with changing and constant user interests, precision values of the retrieved results
ifWeb	profile is a weighted semantic network, nodes represent keywords of user's interest, arcs represent their co-occurrences across the documents, profile initiated by user feedback on a set of documents	based on occurence of keywords documents are rated as interesting or not	experiments done to measure the learning ability of the system and quality of document classification

Table 2: Recommender Systems

System / Authors	Profile - Creation, Storage & Architecture	Ranking / Filtering	Evaluation
SiteIF	profile constructed by observing the user's browsing habits and extracting words representing user's interests from the pages viewed	keyword matching is done to select and filter documents of interest	not discussed
Syskill & Webert	explicit feedback on the pages viewed by the user used to construct the profile, which is a vector of keywords extracted from documents marked interesting by user	uses Bayes classifier to categorize documents, TF-IDF vectors created for the documents and *cosine similarity* used to measure the similarity	effect of training examples and number of features on the accuracy of the learning algorithms
WebWatcher	user's interests specified explicitly in form of keywords, treated as *"goals"* to be achieved during a browsing session, these goals are stored at the server side	*cosine similarity* used to measure the distance between user interest and document feature vectors, feature vectors created using TF-IDF approach	accuracy in predicting interesting pages to the user
Personal WebWatcher	user profile is a set of classes and their associated documents	Naive Bayesian Classifier uses the document feature vectors to classify interesting and not interesting documents	effect of feature vector length on precision of results and classification accuracy
Letizia	pages viewed and bookmarked considered as implicit indicators of interest, profile stored as weighted keyword vectors, profile stored locally at user side	not described, since vectors are involved probably *cosine similarity* is used	not discussed

Table 3: Recommender Systems

1.3 Personalized Web Searches

The initial idea of integrating User Models in Information Retrieval process to improve the effectiveness of the system was introduced by Croft and Thompson in their I^{3}R Sytem [13] and by Brajnik et al in [6]. These systems constructed the user model based on stereotypes i.e. descriptions of classes of users. This was later extended to the information extraction process in [4]. The **UMIE prototype** integrates the User Modeling technique into Information Extracting Process.

Profile - Creation, Storage and Architecture: The UMIE user model consists of the users name, the stereotypes she belongs to (stereotypes are groups of users who have similar interests, organized in the form of single-rooted hierarchical knowledge base) and the domain categories which specify the domain of interest. The profiles also contain a rating that shows if the category is *interesting / not interesting / indifferent* for the user and a number indicating the confidence of UMIE in the given rating. The models are retrieved and stored in a database for each session. The profile is initially constructed by supplying a set of documents to the user and collecting her ratings for them. This information is appended with some personal information about the user in order to classify her into one or more of the stereotypes. All these sources of information are used to calculate the confidence factor of the system for a given category or domain for the given user.

Glover *et al* modify their existing meta-search engine **INQUIRUS**, by using user preferences to define a search strategy that specifies the source selection, query modification and result scoring [14]. By specifying their preferences, different users can use the same query to search different search engines and have different ordering for their results.

Profile - Creation, Storage and Architecture: Though the description of how the user preferences effect their meta-search engine results has been discussed, the actual structure of the user profiles or how this information is stored is not described. The user's preferences are considered as a list of choices or categories in the form of keywords that are prepended or appended to the query keywords.

Ranking / Filtering scheme: After the pages are retrieved by the several search engines, they are sorted based on the knowledge of user preferences before being displayed to the user. The document ordering task is resolved by employing the *'sort by value'* ordering policy of the utility theory [16]. All the categories of the user's information needs have associated value function given by the formula

$$\mathcal{U}(d_j) = \sum w_k v_k(x_{jk})$$

where w_k is the weight and v_k is the value of the k_{th} attribute and x_{jk} is the level of the k_{th} attribute for the j_{th} document.

Evaluation: Experiments for evaluating the system have not been done or discussed. Some results have been reported describing the selection of attributes and their effects on the retrieved results.

METIORE [8] is a prototype of an information filtering system that allows for personalized access to a library of research publications. The system asks the user to specify a keyword *objective* for each session. It uses this keyword along with the past history of the user to sort the retrieved documents, ranking the most interesting one at the top.

Profile - Creation, Storage and Architecture: The user model stores some general information about the user (personal data such as name, login, interface language, etc.). For each *objective*, all the evaluated documents are stored and their features (such as keywords, authors, etc.) also inherit the corresponding evaluations.

Ranking / Filtering scheme: The system stores the prior evaluations of the documents for the given *objective* for each user. It uses the Naive Bayes theory to predict the probability of the user's evaluation for a given document under the context of the current *objective*. The prediction technique makes use of precedent user evaluations to calculate the degree of relevance of a given document to the current objective of the user.

Evaluation: The prediction accuracy of the system was evaluated in a user study involving researchers on a dataset of scientific publications. The results indicate that in more than 50% of the cases the system was able to correctly predict the documents matching to the user's interests.

Pretschner, Gauch and Chaffee present an ontology based approach for constructing user profiles to assist in personalized search systems. The profile trees are built using a '*watching-over-your-shoulder*' approach, utilising a plug-in embedded into the user's web browser to process the browser's cache history [32]. The personalized system is built over the **Profusion** search engine. As an extension work [9], software agent technology in employed to build the personal ontologies for browsing the web in the **OBI-WAN** project[2]. The system uses distributed agents (local and regional) to organize the information on the web.

Profile - Creation, Storage and Architecture: The user profile is structured as a concept hierarchy of 4400 nodes. Each of these nodes contain a weight (indicating the user's interest in that topic) and a set of keywords (representing the content of that node) which are stored in the form of vectors. The profile is constructed by analyzing webpages in the browser's cache folder and altering the node weights in the profile using the information about the time spent on the page, length of the page etc. The profile is stored at the user side and is used to re-rank the results retrieved by the search engine.

Ranking / Filtering scheme: The re-ranking algorithm considers the user's interest in the categories representing the document and how closely these categories match to the document in addition to the original rank given to it by the search engine. The modified ranking considering the user's interests is calculated in this manner.

$$\varrho(d_j) = w(d_j) * (0.5 + 0.25 \sum_{i=1}^{4} * \pi(c_i) * \gamma(d_j, c_j))$$

where $w(d_j)$ is the original rank of the search engine, $\gamma(d_j, c_j))$ gives the measure of how closely a category c_j describes the content of the document d, $\pi(c_i)$ is the interest of the user in the category c_i.

Evaluation: In a set of experiments conducted with real users, the 11-point precision average along with normalized distance based performance measure is used to evaluate the system. The process involved clustering of documents into relevant and non - relevant sets and checking the number of relevant documents appearing at the top of the re-ranked list. The experiments reveal an increase in the retrieval performance due to the re-ranking of results. In another set of experiments, precision and correctness measures were used to evaluate the system. Precision is the number of relevant pages viewed as compared to the total number of pages viewed by the user. Correctness measures the total number of correct pages viewed plus the number of incorrect pages not viewed by the user.

Liu et al attempt to improve the retrieval effectiveness of a search engine by augmenting the query keyword with a set of categories, which are inferred as a context for the query. The system utilizes the ODP category hierarchy to build the user profile from the search history. [27]

Profile - Creation, Storage and Architecture: The system stores both the users search history and the user profile. It collects the search history from the search records of the search engine. The user

[2]http://www.ittc.ku.edu/obiwan/

profile consists of a set of categories of terms representing the user's interest in that domain. These terms are weighted to indicate their significance for that user. Both the search history and the user profile are constructed in the form of matrices containing rows of weighted term vectors. The system also has a general profile, which is derived from the ODP hierarchy. It is used to categorize the user queries, in order to understand their contexts. The user profile is learnt from the user's search history by applying algorithms like LLSF, Rocchio, kNN etc.

Ranking / Filtering scheme: The system uses *cosine similarity* measure to determine the categories from the general profile to be appended to the query (context information). The ranking scheme considers the number of documents retrieved under a particular category, the rank of that category and similarity of the query with that category.

Evaluation: The accuracy of mapping user queries to categories is evaluated by

$$\sum \frac{1}{1+rank_{ci}-idealrank_{ci}}/n$$

where n is the number of related categories to the query, $rank_{ci}$ is the rank for category ci and $idealrank_{ci}$ is the highest possible rank for ci. The efficiency of web page retrieval is measured by taking the average precision of retrieved results for all the queries. It is observed that on increasing the size of the training data the accuracy of the user profiles increases.

Persona is a personalization module wrapped around a search engine i.e. it lies between the search engine and the end user, refining the results retrieved. In this system user interests are modeled as a collection of weighted concepts which are derived from the ODP ontology (Open Directory project[3]). It uses the tree-coloring approach to keep track of the nodes visited by the user and colors the nodes indicating the number of visits by the user, the number of positive or negative ratings given by the user etc [39].

Profile - Creation, Storage and Architecture: The information about the user's interests or disinterests is obtained explicitly in the form feedback of positive or negative ratings given by the user. This feedback information is used to update the user profile. The user profile is stored as a collection of weighted concepts based on the ODP ontology. The queries entered by the user generate context words and each context word is associated with a node in the ontology. Hence, based on the user's actions only a few of the nodes are effected and these are stored instead of the entire ODP ontology, thus allowing scalability in the system. This profile information is stored in the form of tables.

Ranking / Filtering scheme: When a query is entered a table look-up is done to find the context and the results are given a bias (high or low weighting) depending on the user's previous feedback.

Evaluation: Since the user profile is a collection of weighted concepts on the ODP ontology, evaluations are done to measure the change in the average ranking of the relavant and non-relevant nodes. Experiments were done with an actual search engine to test the learning ability of the Persona algorithm. Upon entering the feedback of the likes and dislikes for a given concept, the system was able to retrieve more relevant results for queries on that concept.

Outride is a personalized search system, which acts as an interface between the user and the search engine [31]. It does *query modification* based on the user profile and *result re-ranking* of the results retrieved by the search engine before displaying to the user. The system is integrated into the web browser as a side-bar component, having access to the user interactions and allowing direct manipulation.

Profile - Creation, Storage and Architecture: The initial profile is provided by the user in the form of bookmarks, which the system classifies into the ODP hierarchy, accordingly adjusting the weights.

[3]http://www.dmoz.org/

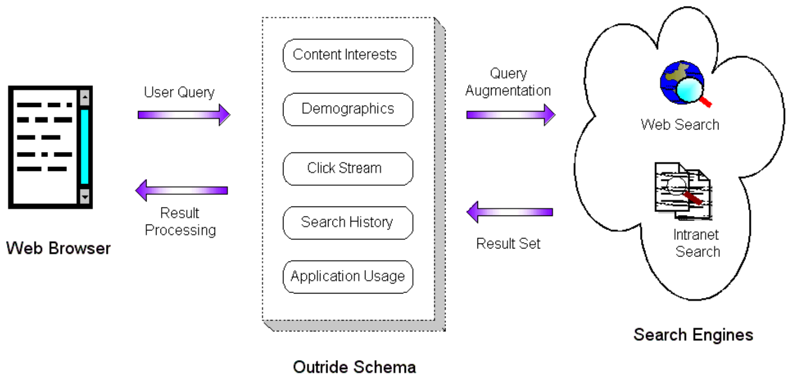

Figure 5: The Outride System

It is updated with information extracted from pages browsed by the user. The system uses this profile information to augment the query entered by the user before feeding it to the underlying search engine.

Ranking / Filtering scheme: The re-ranking or filtering of search results is done by employing vector space methods to compare the titles and other meta data of the web pages with the contents of the user profile.

Evaluation: The Outride approach is evaluated using metrics such as time elapsed for successful search and number of interface actions (mouse clicks, keyboard entries etc.) were used. The performance of Outride was compared with other search engines like Google, Yahoo etc. It was observed that the participants in the experiments found answers to their queries much quickly with Outride than with the other search engines and they needed to perform fewer actions for doing so.

PresTo implements personalized web search by re-ranking the search results from a given search engine based on a user profile [17]. It is a plug-in tool for Internet Explorer and maintains the profile based on user interactions with web search sites like (Yahoo, Google, CiteSeer etc.). This profile is used to re-rank the results retrieved by these engines and the personalized ranking is shown in the sidebar along side the actual results in the main window. Users can select the results displayed in both the window panes.

Profile - Creation, Storage and Architecture: The user profile is stored at the client-side and is kept private only to the user. The profile is learned while the user is browsing and searching using the web browser. The tool logs the search activity for the user and the profile adapts itself to the changing interests of the user. For each queried keyword, PresTo! keeps track of the URLs visited, the keywords associated with these URLs, the number of user's visits to that page, etc. All this data is stored in a flat file which is used as a look-up file and it is indexed as a collection of single-rooted word-suffix trees. The number of times a query has been issued by the user is considered as an indication of the user's interest in that topic.

Ranking / Filtering scheme: The re-ranking algorithm uses the Vector Space Model to measure the similarity between the user profile and the retrieved results. The user profile vectors are constructed from the previously visited URLs and their weights are determined from the parameters like the number of times the user has viewed the page previously, the no. of times the query was issued, the depth of the node in the tree where the URL was found etc. Similarly a vector is created for the retrieved results using the terms from the title, URL and page summary. These terms are weighted using the usual TF-IDF metric. The scores for these search results are measured based on their *cosine similarity* to the vectors from the user profile along with the original ranking given to that page by the search engine.

System / Authors	Profile - Creation, Storage & Architecture	Ranking / Filtering	Evaluation
UMIE	user model consists of user name, stereotypes she belongs to and domain categories specifying domain of interest, initially constructed from ratings collected from the user for a set of documents	not discussed	not discussed
INQUIRUS	not described, profile is probably stored in form of keyword lists	sorting done based on value functions of utility theory, weightage is given to value derived from the content of the retrieved pages	not discussed
METIORE	stores personal data like name, login, etc., document evaluations also stored along with their features like author, keywords etc.	no ranking done, Naive Bayes theory used to predict the relevancy of a document to user's interests	accuracy in predicting interesting pages for the user
OBIWAN	concept hierarchy of 4400 nodes, containing weights and set of keyword vectors, built by analyzing webpages in browser's cache folder, stored at the user side	re-ranking done using the user's interests on the various categories, how close the categories match the document in addition to the original rank given by the search engine	precision values and correctness of retrieved results

Table 4: Personalised Websearch

System / Authors	Profile - Creation, Storage & Architecture	Ranking / Filtering	Evaluation
Liu et al	set of categories of terms representing the user's interest, terms are weighted based on significance, profile learnt from search history, represented as matrix containing rows of weighted vectors	*cosine similarity* measure determines categories for appending to the query, ranking scheme considers the rank of that category and its similarity with the query	accuracy of mapping queries to categories, average precision of retrieved results
Persona	profile stored as weighted concepts on ODP ontology, weights determined from user feedback on viewed documents, profile stored centrally in form of tables	results are biased (high or low) depending on user's previous feedback	learning ability of the system evaluated, effect of the feedback on the weightage of nodes studied
Outride	profile initiated by user submitting browser bookmarks, built by modifying the ODP ontology and adjusting weights based on user's browsing activity	vector space methods are used to compare titles & meta data of the web pages with contents of the user profile	time elapsed for a successful search and user activity (mouse clicks, keyboard actions)
Presto	learnt from user's browsing activity, adapts to the changing interests of the user, stored at client side in form of flat file	Vector Space Model used to measure similarity between weighted vectors representing user profile and retrieved results	not discussed

Table 5: Personalised Websearch

Make	Year	Price	Color	Miles
Benz	2004	35K	Black	20000
Benz	2002	25K	Silver	32000
Benz	2003	32K	Grey	26000
BMW	2003	35K	Grey	30000
BMW	2002	32K	Grey	35000
BMW	2004	40K	Black	22000
Opel	2002	20K	Black	35000
Opel	2003	25K	White	27000
Opel	2001	17K	Silver	43000
VW	2004	22K	Grey	27000
VW	2002	18K	Black	35000
VW	2003	20K	Black	30000

Table 6: The Cars relation

1.4 Personalization in Databases

Unlike in IR, queries in Databases are characterized by hard constraints, delivering the objects if available or rejecting the user's request. Implementing of personalization in a Database System, requires a flexible modeling technique for user preferences. There have been two distinctive approaches towards the modeling of user preferences. We shall discuss these approaches utilising example queries built on the *Cars* relation schema as described in Table 6

Qualitative Approaches:

The personal preferences in the "real world" negotiate for the best possible match and are hence treated as soft constraints in [18]. **Kießling et al** propose a model which treats preferences as strict partial orders. They present several constructors (HIGHEST(A), BETWEEN (A,[low,up]), AROUND(A) etc.) and algebraic laws for modeling of preferences. To bypass the problems posed by *empty-result, exact-match query models* and *flooding effect* in e-commerce applications, a "Best Matches Only" (BMO) query model is proposed, which makes the suitable matches for the preferences [18]. It introduces clauses in the SQL and XPATH query languages using which the user can specify her preferences while issuing the query [20], [19].

Profile - Creation, Storage and Architecture: The manner in which the preferences are modeled or captured from the user is not discussed. It is assumed that the user explicitly enters the keywords in the *preference*-clause while writing the query.

Ranking / Filtering scheme: A preference P is interpreted as a directed acyclic graph, called as a *'better than' graph G*, such that **'x <P y'** is interpreted that *'I like y better than x*' where *x* and *y* are nodes of *G*. A *"top-k"* query model is applied to return k - best matching objects.

Evaluation Metrics: In an implementation of PreferenceSQL [20], experiments are done to test the proposed approach with a German job search engine. It is observed that the performance of SQL queries is improved through implementing the Pareto-Optimal $PREFERRING$ clause.

Examples: This following examples below indicated the manner in which preferences are implemented using this approach when written in PreferenceSQL

Make	Year	Price	Color	Miles
Benz	2003	32K	Grey	26000

Table 7: Result set for Query 1

Make	Year	Price	Color	Miles
BMW	2003	35K	Grey	30000

Table 8: Result set for Query 2

Example Query 1: Find the cheapest cars manufactured by Benz prior to 2005, but not before 2003.

```
SELECT * FROM Cars
WHERE make = 'Benz'
PREFERRING LOWEST(price)
   AND year BETWEEN 2003, 2005
```

would return the tuple shown in Table 7 and when written in Preference XPATH, would return the tuple shown in Table 8

Example Query 2: Find the least used cars manufactured by BMW in price range of 30K.

```
/CAR #[ (@make) in ("BMW") and (@miles) minimal and
(@price) around 30K ]#
```

A similar framework for formulating preferences has been suggested by **Chomicki** in [11], where preferences are specified using first-order logical formulas. These *preference formulas* are easily embedded into relational algebra or SQL by introducing $\succ$, a new operator called *winnow*, that selects the most preferred tuples. Complex preference queries are easily formulated by utilizing existing SQL constructs. The author focuses mostly on intrinsic preferences, based only on values occurring in tuples. The semantics of the *winnow* operator are similar to the BMO Query model proposed by Kießling et al. Several properties of preferences (intrinsic, extrinsic, etc.) including iterated preferences and general algebraic laws governing the *winnow* operator are discussed in [11]. But this framework is limited to applications where the preferences are modeled within the relational model of data. Any implementation with existing query languages like SQL or XPATH is not discussed.

Ranking / Filtering scheme: The notion of ranking is implicitly implemented using the *iterated preferences*. A ranking operator $\eta_C(R)$ is introduced, for a Relation schema *R* and *C* which is a preference formula defining a preference relation $\succ_C$. The properties of the ranking operator are similar to the *winnow* operator which is used for selecting the preferred tuples.

Examples: The preference query is a relational algebra query containing at least one occurrence of the *winnow* operator. Considering the schema described in the Table 6, the following preference query w_C(*Cars*) using the preference relation C defined by the formula
Example Query 3: Find the least priced cars of all the manufacturers.

$$(m, y, p, c, mi) \succ (m^1, y^1, p^1, c^1, mi^1) \equiv (m = m^1 \wedge p < p^1)$$

Make	Year	Price	Color	Miles
Benz	2002	25K	Silver	32000
BMW	2002	32K	Grey	35000
Opel	2001	17K	Silver	43000
VW	2002	18K	Black	35000

Table 9: Result Set for Query 3

would produce the result set shown in Table 9

Quantitative Approach:

A preference model for implementing quantitative preferences in databases was defined initially by **Agrawal and Wimmers** in [1]. In this framework, the user can indicate her degree of interest on an entity using a numeric score between [0,1] which is stored in a *score* datatype or by *veto*ing it, or by explicitly stating indifference. The entity is described by a set of named fields, '*' symbol is used to match any element of that type. The framework also permits combining of preferences using a generic *combine* operator.

Profile - Creation, Storage and Architecture: It is assumed that the user explicitly provides her preferences to the system. A preference function maps the user's preference to a score indicating the user's interest on that preference. The system does not deal with the modeling and storage of this information of the user's interests, working under the assumption that this data is entered by the user explicitly while making the query.

Example: The user's preference functions are obtained explicitly and stored in separate tables. Simple SQL queries over these tables having constraints on preference scores are used to execute preference queries.

Koutrika and Ioannidis present a framework which makes use of structured user profiles(storing simple, unconditional preferences) for optimization of database queries based on user preferences [23]. There is a possibility of deriving preferences using the 'and', 'or' constructs. This framework provides one of the first solutions towards modeling of user preferences in Database systems in the form of a structured profile. In the generalized preference model [21], preferences over a database contents are modeled as a directed graph *Personalization Graph* - $\mathcal{G}\langle\mathcal{V},\mathcal{E}\rangle$, where the nodes in $\mathcal{V}$ are relation nodes and attribute nodes and edges in $\mathcal{E}$ are selection edges (representing a possible selection condition from an attribute to a value node) and join edges (representing a join between attribute nodes). The model described in [21] generalizes the earlier model of [23], which provides the possibility of more sophisticated preference queries such as 'I like films with duration *around 2h*', 'I *do not* like thrillers', 'I like movies *without* violence', etc.

Profile - Creation, Storage and Architecture: The user preferences are obtained explicitly from the user. These preferences are stored at the level of atomic query elements along with the degree of interest [22]. A typical user profile looks like $<q$, $d_T(u)$, $d_F(u)>$, where the first entry (q) of the tuple describes the atomic preference, the second entry ($d_T(u)$) indicates the user's concern for the presence of the tuple and the 3rd entry ($d_F(u)$) indicates the user's indifference to the presence or absence of the tuple. Using these tuples, the system can capture all types of preferences (*positive, negative* and *indifference*). These preferences over the contents of a database are expressed on top of the *Personalization*

Graph, which is an extension of the database schema graph, as described above. The user can also specify the *degree of criticality* for a given preference, which is used for ordering the user's multiple preferences.

Ranking / Filtering scheme: For a given Query Q, the retrieved tuples are filtered in a manner such that they satisfy at least L of the top-*K* preferences of the user. These tuples are then ranked based on the decreasing order of degree of interest.

Evaluation Metrics: In the first set of experiments described in [23], the execution time of their algorithm is measured*w.r.t* size of profiles, the number of tuples retrieved for increasing number of preferences, the execution time performance for *SingleQuery (SQ)* and *MultipleQuery (MQ)* types of personalized queries. It is observed that the query execution time decreases with increasing profile size, the number of retrieved tuples increases along with the increase in the number of preferences listed and the time performance of *MQ* is better than *SQ* as the latter involves a lot of $join$ operations depending on the number of preferences. In the second set of experiments [21] the general preference algorithms are evaluated by measuring the execution time for different top-*K* preferences and number of preferences to be satisfied (*L*), the effectiveness of the personalized queries through a user study, comparing their ranking functions with the user's degree of interest. The user study reveals that personalization reduces the effort required by the people in finding information and their ranking functions were closely matching the user's interests. All these experiments were carried out on a dataset from the *Internet Movie Database*[4] and implemented on Oracle9*i*.

Examples: Under this model, preference queries produce personalized answers that satisfy at least L of the top-*K* preferences and are ranked based on the *doi*. Consider a simple query

Example Query 4: Find the black-colored cars manufactured by Opel or VW after the year 2002.

```
SELECT * FROM Cars
```

under the following preferences (among which *L*=2 should be satisfied)

```
(P1) Cars.Make = 'Opel' or Cars.Make = 'VW'(presence)
(P2) Cars.Year >= 2002 (absence 1-1)
(P3) Cars.Colour = 'Black'(absence 1-1)
```

The following sub queries will be built for each preference type, where the $degree$ value represents the user's degree on interest in this preference

Example Query 5:

```
Q1: SELECT * 0.7 degree
FROM Cars
WHERE Cars.Make = 'Opel' OR Cars.Make = 'VW'

Q2: SELECT * 0 degree
FROM Cars
WHERE Cars.Year >= 2002

Q3: SELECT * 0.8 degree
```

[4]available at http://www.imdb.com

Make	Year	Price	Color	Miles
Opel	2002	20K	Black	35000
Opel	2003	25K	White	27000
VW	2003	20K	Black	30000
VW	2004	22K	Grey	27000

Table 10: Result set of Query 6

```
FROM Cars
WHERE Cars.Make NOT IN (SELECT C1.Make
FROM Cars C1
WHERE C1.color = 'Black')
```

The expected results are obtained by taking the union of the partial results of the sub-queries, grouped by the attributes of the initial query and excluding groups with less than L rows, where *r* is a ranking function provided by a user defined aggregate function.

Example Query 6:

```
SELECT make, r(degree)
FROM Q1 UNION ALL Q2 UNION ALL Q3
GROUP BY make
HAVING count(*)>=2
ORDER BY r(degree)
```

Authors	Profile - Creation, Storage & Architecture	Ranking / Filtering	Evaluation
Kiessling	no modeling or storage	no ranking, $X <_P Y$ means *'X'* is better than *'Y'*	mostly theoretical work, PreferenceSQL implemented in a search website
Chomicki	no modeling or storage	ranking operator exists to select the preferred tuples	not discussed
Agrawal and Wimmers	no modeling or storage	not discussed	not discussed
Koutrika and Ioannidis	atomic preferences and user's concern for its presence or absence is stored as a tuple; expressed as a *Personalization Graph* on a DB schema; user can specify *degree of interest* used for ordering preferences	ranking done based on *degree of interest*; filtering done to satisfy *L* of the top-*K* preferences	query execution time *w.r.t* - size of profiles, *SingleQuery* or *MultipleQuery* types, various values of top-*K* and *L*; user study for effectiveness of personalized queries

Table 11: Preferences in Databases

References

[1] R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *SIGMOD Conference*, pages 297–306, 2000.

[2] F. Asnicar and C. Tasso. ifweb: a prototype of user model-based intelligent agent for documentation filtering and navigation in the word wide web. In *Proc. of 1st Int. Workshop on adaptive systems and user modeling on the WWW*, 1997.

[3] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, 1992.

[4] E. Benaki, K. A. Vangelis, and C. D. Spyropoulos. Integrating user modeling into information extraction: The UMIE prototype. In *Proceedings of UM*, pages 55–57. Springer, 2-5 1997.

[5] K. D. Bollacker, S. Lawrence, and C. L. Giles. A system for automatic personalized tracking of scientific literature on the web. In *Proceedings of ACM Conference on Digital Libraries*, pages 105–113. ACM Press, 1999.

[6] G. Brajnik, G. Guida, and C. Tasso. User modeling in intelligent information retrieval. *Inf. Process. Manage.*, 23(4):305–320, 1987.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[8] D. Bueno and A. David. METIORE: A personalized information retrieval system. In *User Modeling*, pages 168–177, 2001.

[9] J. Chaffee and S. Gauch. Personal ontologies for web navigation. In *Proceedings of CIKM*, pages 227–234, New York, NY, USA, 2000. ACM Press.

[10] L. Chen and K. Sycara. WebMate: A personal agent for browsing and searching. In K. P. Sycara and M. Wooldridge, editors, *Proceedings of Agents'98*, pages 132–139, New York, 1998. ACM Press.

[11] J. Chomicki. Querying with intrinsic preferences. In *EDBT*, pages 34–51, 2002.

[12] M. Claypool, P. Le, M. Waseda, and D. Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40, 2001.

[13] W. B. Croft and R. H. Thompson. I3R - A new approach to the design of document retrieval systems. *Journal of the American Society for Information Science*, 38(6):389–404, 1987.

[14] E. J. Glover, S. Lawrence, W. P. Birmingham, and C. L. Giles. Architecture of a metasearch engine that supports user information needs. In *Proceedings of CIKM*, pages 210–216. ACM, 1999.

[15] T. Joachims, D. Freitag, and T. M. Mitchell. Web watcher: A tour guide for the world wide web. In *IJCAI (1)*, pages 770–777, 1997.

[16] R. Keeney and H. Raiffa. *Decisions with multiple objectives: Preferences and value tradeoffs*. J. Wiley, New York, 1976.

[17] K. Keenoy and M. Levene. Personalization of web search. *Commun. ACM*, pages 201 – 228, 2005.

[18] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[19] W. Kießling, B. Hafenrichter, S. Fischer, and S. Holland. Preference XPATH: A query language for e-commerce. In *Internationale Tagung Wirtschaftsinformatik*, pages 427 – 440, 2001.

[20] W. Kießling and G. Köstler. Preference SQL - design, implementation, experiences. In *VLDB*, pages 990–1001, 2002.

[21] G. Koutrika and Y. Ioannidis. Personalized queries under a generalized preference model. *ICDE*, pages 841–852, 2005.

[22] G. Koutrika and Y. E. Ioannidis. Personalization of queries based on user preferences. In *Preferences*, 2004.

[23] G. Koutrika and Y. E. Ioannidis. Personalization of queries in database systems. In *ICDE*, pages 597–608, 2004.

[24] T. Kurki, S. Jokela, R. Sulonen, and M. Turpeinen. Agents in delivering personalized content based on semantic metadata, 1999.

[25] K. Lang. NewsWeeder: learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning*, pages 331–339. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995.

[26] H. Lieberman. Letizia: An agent that assists in web browsing. In *In proc. of 14th International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.

[27] F. Liu, C. Yu, and W. Meng. Personalized web search for improving retrieval effectiveness. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):28–40, 2004.

[28] D. Mladenic. Personal webwatcher : Design and implementation. In *Technical Report IJS-DP-7472, School of Computer Science, Carnegie Mellon University*, Pittsburgh, USA, 1996.

[29] A. Moukas and P. Maes. Amalthaea: An evolving multi-agent information filtering and discovery system for the WWW. *Autonomous Agents and Multi-Agent Systems*, 1(1):59–88, 1998.

[30] M. J. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert : Identifying interesting websites. In *Proc. of 13th National Conference on Artificial Intelligence*, 1996.

[31] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized search. *Commun. ACM*, 45(9):50–55, 2002.

[32] A. Pretschner and S. Gauch. Ontology based personalized search. In *ICTAI*, pages 391–398, 1999.

[33] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[34] J. Rucker and M. J. Polanco. Siteseer: personalized navigation for the web. *Commun. ACM*, 40(3):73–76, 1997.

[35] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[36] B. D. Sheth. A learning approach to personalized information filtering. Master's thesis, January 1994.

[37] H. Sorensen and M. McElligott. Psun: A profiling system for usenet news (extended abstract).

[38] A. Stefani and C. Strapparava. Personalizing access to web sites : The siteif project. In *Proc. of 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98*, Pittsburg, USA, 1998.

[39] F. Tanudjaja and L. Mui. Persona: A contextualized and personalized web search. volume 3, page 53, 2002.

[40] D. Widyantoro, J. Yin, M. Nasr, L. Yang, A. Zacchi, and J. Yen. Alipes: A swift messenger in cyberspace, 1999.

[41] T. Yan and H. Garcia-Molina. SIFT—A tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conference*, pages 177–186, New Orleans, 1995.