# Ontology Based Personalized Search

by

Alexander Pretschner

Dipl.-Inform., RWTH Aachen, Germany, 1998

Submitted to the Department of Electrical Engineering and Computer Science and the Faculty of the Graduate School of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

Dr. Susan Gauch
(Committee Chair)

Dr. Arvin Agah
(Committee Member)

Dr. James Miller
(Committee Member)

Date thesis accepted

# Abstract

With the exponentially growing amount of information available on the Internet, the task of retrieving documents of interest has become increasingly difficult. Search engines usually report more than 1,500 hits, and out of the top twenty results, only one half turn out to be relevant to the user. One reason for this is that Web queries are in general very short and give an incomplete specification of individual users' information needs.

This thesis is exploring ways of incorporating users' interests into the search process to improve the results. The user profiles are structured as a concept hierarchy of 4,400 nodes. These are populated by "watching over a user's shoulder" while he is surfing. No explicit feedback is necessary.

The obtained profiles are shown to converge and to reflect the actual interests quite well. One possible deployment of these profiles is investigated: re-ranking and filtering search results. The increases in performance are moderate, but they are noticeable, and they show that fully automatic creation of large hierarchical user profiles is possible.

# Acknowledgements

I would like to thank all the members of the committee for supporting this project. In particular, I feel indebted to Dr. Gauch for providing me with a research assistant position and for many valuable lessons she did not explicitly present as such.

Furthermore, I would like to thank the Fulbright Commission for their support.

This thesis would not have been possible without the help of those who regularly sent me their cache folders and helped me evaluate the system.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

As of March 1999, the Internet provides about 165 million users worldwide[1] with every imaginable type of information. In general, people have two ways to find the data they are looking for: they can *search*, and they can *browse*. Search engines index some of the documents on the Internet and allow users to enter some keywords to retrieve documents that contain these keywords. Browsing is usually done by clicking through a hierarchy of subjects until the area of interest has been reached. The corresponding node then provides the user with links to related websites. The search and browsing algorithms are essentially the same for all users.

It is unlikely that 165 million people are so similar in their interests that one approach to searching or browsing, respectively, fits all needs. Indeed, in terms of searching, about one half of all retrieved documents have been reported to be irrelevant [8]. Every user knows how time consuming and error-prone it is to locate specific information.

[1] according to Nua Internet Surveys, www.nua.ie/surveys

The main problem is that there is too much information available, and that keywords are rarely an appropriate means of locating the information in which a user is interested.

Presumably, information retrieval will be more effective if individual users' idiosyncrasies are taken into account. This way, an effective personalization system could decide autonomously whether or not a user is interested in a specific webpage and, in the negative case, prevent it from being displayed. Or, the system could navigate through the Web on its own and notify the user if it found a page or site of presumed interest.

This thesis studies ways to model a user's interests and shows how these models - also called profiles - can be deployed for more effective information retrieval and filtering.

A system is developed that "watches over the shoulder" of a user while he is surfing the Web. A user profile will be created over time by analyzing surfed pages to identify their content and by associating that content with the length of the document and the time that was spent on it. When pages about certain subjects are visited again and again, this could be an indication for the user's interest in that subject. Except for the act of surfing, no user interaction with this system will be necessary.

If such a profile accurately reflects a user's interest, it can be of tremendous advantage for the user. Imagine a recommendation service that would automatically notify you if new websites, books, or articles have been published. Other users with a similar profiles could recommend items to you. If an expert with very specific skills is needed, a query for those skills could be matched against avail-

able profiles. In addition, the profiles could be used for re-ranking and filtering search results. This is the focus of this thesis: Once the user profiles are gathered (in fact, they never stop being gathered), they are used to re-rank search results, hopefully putting personally relevant items on top of the result list. They also help to identify documents that are most probably not interesting to the user and that should therefore be excluded from the result list.

This thesis shows that it is possible to build a system that learns a user's interests without any explicit user feedback. This seems important since in the author's opinion, users are unlikely to frequently answer questions "Did you find that page relevant?" - they tend to annoy the user. It is shown how these profiles can be used to achieve search performance improvements. The increases in performance are small, but they are noticeable, and they are a first step.

Several approaches to personalization have been investigated. After giving an overview of the theoretical and practical context of this thesis in the remainder of this chapter, these different approaches are discussed in some depth in chapter 2. Chapter 3 explains how exactly the profiles are obtained. Several approaches are suggested and thoroughly evaluated. These profiles are then used for re-ranking and filtering in chapter 4. Different re-ranking and filtering mechanisms are developed, discussed and evaluated. It turns out that performance improvements with the automatically created profiles are possible. Chapter 5 summarizes the ideas and results of this thesis.

Appendix A contains additional information on profile convergence, and appendix B contains a concise description of the implementation.

## 1.1 Background

This section first briefly reviews some basic notions in information retrieval. The second part consists of a description of the OBIWAN project which forms the framework for the presented personalization system.

### 1.1.1 Information Retrieval: The Vector Space Model

Information retrieval is concerned with retrieving data that match certain (user-defined) criteria. Examples are finding documents that match a given query (a list of words) or determining similarities between documents for clustering or categorization purposes.

The most obvious way of finding documents with respect to a given query is to simply look for documents that contain words that are also contained in the query. This is what most local search services on websites (i.e., sites that index their own pages) do by performing a *grep*-like command on the set of available documents. The number of matches (words that occur in both the query and a document) is used to rank the search results. This basic model does not take into account different word frequencies, nor is there an obvious way to determine the similarity between documents.

In the vector space model [48], documents are represented as vectors of keywords. Given a set of documents which together contain $n$ different keywords, each query or document is represented as an $n$-dimensional vector where each component corresponds to one keyword in the collection. The motivation for this representation is that there is a natural means of comparing two vectors: the an-

gle, or inverse cosine of the dot product, between these two vectors. The smaller the angle is, the more similar are the two vectors - and therefore the documents they represent.

In the *Boolean vector space model*, the entries of the vectors are either 1 or 0, depending on whether or not the keyword associated with this entry occurs in the document represented by the vector. Obviously, this model does not take into account word frequencies. Assuming that terms with high occurrence frequencies describe the content of a document better than terms with low occurrence frequencies, the Boolean vector space model appears to be too coarse.

This problem can be circumvented by incorporating word frequencies in the model. One could simply divide the number of occurrences of a given keyword in a given document by the total number of words in this document and use these word frequencies as elements of the vector. However, there is a drawback with this approach: If the collection of documents is concerned with "apples", the word "apple" will probably occur many times in each document, and is therefore not suited to represent the *content* of the document precisely. Consider an application that asks for the content of a given document: An algorithm based on the above approach would probably assign the topic "apple" to all documents. But since the collection of documents consists of nothing but documents on apples, there is no real discrimination between the documents.

The solution to this problem is based on "punishing" terms that occur too often. The elements of the vectors that represent documents consist hence of a combination of term frequencies and some inverse of the overall frequency of this particular term. This ensures that terms that occur often in a document, but not

too often in the collection of documents, get the highest weights.

To be more precise, the term frequency of term $t_i$ in document $d_j$, $tf_{ij}$, is defined as

$$tf_{ij} = \text{occurrences of } t_i \text{ in } d_j.$$

The inverted document frequency of a term $t_i$ in a collection $\mathcal{D}$ of documents, $idf_i$, is typically defined as

$$idf_i = \log \frac{\text{number of documents in } \mathcal{D}}{\text{number of documents in } \mathcal{D} \text{ that contain term } t_i}.$$

These factors are then combined to calculate the *tf·idf* weight of every term in every document. The weight of term $t_i$ in document $d_j$, $w_{ij}$ is defined as their product:

$$w_{ij} = tf_{ij} \cdot idf_i.$$

In order not to retrieve long documents in preference to short ones, these weights are normalized as

$$\tilde{w}_{ij} = \frac{w_{ij}}{\sqrt{\sum_{t_i \in d_j} w_{ij}^2}}.$$

If $\mathcal{D}$ contains $m$ documents, and these documents together contain $n$ different terms, then there are $m$ $n$-dimensional vectors with their respective $w_{ij}$ entries, where $i$ ranges from 1 to $n$, and $j$ ranges from 1 to $m$.

Clearly, queries can be represented in a very similar way. The set of documents in this case consists of 1 "document", namely the query. It is essential though that the query is also represented as a vector with the same dimension as the document vectors in the document collection of interest.

If $\mathbf{d_1}$ and $\mathbf{d_2}$ are document vectors[2] with $tf \cdot idf$ weight components, the similarity $\mathrm{sim}(\mathbf{d_1}, \mathbf{d_2})$ of these two documents is expressed as the cosine of the angle between the two document vectors:

$$\mathrm{sim}(\mathbf{d_1}, \mathbf{d_2}) = \frac{\mathbf{d_1} \cdot \mathbf{d_2}}{\| \mathbf{d_1} \| \cdot \| \mathbf{d_2} \|} = \frac{\sum_{i=1}^{n} \mathbf{d_1}(i) \cdot \mathbf{d_2}(i)}{\sqrt{\sum_{i=1}^{n} \mathbf{d_1}(i)^2 \cdot \sum_{i=1}^{n} \mathbf{d_2}(i)^2}},$$

where $n$ is the number of different terms in a given collection of documents, and $\mathbf{d_k}(i)$ addresses the $i$th component of vector $\mathbf{d_k}$.

Alternative approaches to document representation [48] include probabilistic models, fuzzy set models, and n-grams.

It is common practice [12] to exclude words which do not carry any information when isolated. Examples are "and", "or", "has", and "why".

Sometimes words are "stemmed" [12] before the weight vectors are calculated. This means that common suffixes such as "ing", "e", or "ation" are removed - "personalize", "personalizing", and "personalization" all become "personaliz". The reason for this is that if a user wants to find all documents on "personalization agents", he will probably also be interested in documents on "personalizing" agents. Some Internet search engines such as AltaVista require the user to take care of this by providing wildcards in the queries.

[2] one of which can, of course, be a query vector

### 1.1.2 The OBIWAN project

The exponentially growing number of webpages available on the Internet make it unlikely that centralized information services will provide users with relevant up-to-date information. There are basically two ways of "information services": browsing services, and searching services:

- Browsing services typically maintain a hierarchy of subjects, or *ontology*. By starting at the root subject, the user can navigate through this tree until the subject of interest is found. Top level nodes typically include broad categories such as "Economics", "Arts", "Music" or "Sports", and bottom level subjects can be very narrow, for instance "Orthic Dark Gray Chernozemic Soil".[3] Examples for browsing ontologies can be found at major search engines such as www.yahoo.com, or at the virtual WWW library, www.vlib.org.

- Search services typically let the user enter one or more keywords. A given database – about roughly 30% of all available pages on the Web in the case of AltaVista, or all known Java JDK 1.2 bugs in the case of one section of Sun's JavaSoft site – is then searched, and the results are returned to the user.

These services tend to be centralized in the sense that an index of the database is stored on one centralized server which processes all requests.

Clearly, there are problems with this approach. In the case of Internet search or browsing services, the wealth of information is simply growing too fast for accurate mirroring, and the databases will eventually become problematically large.

[3] In this case, the path from the root (the WWW Virtual Library, www.vlib.org) is: Science - Earth Science - Forestry - Soils and Substrates - Soil profiles of Canada - Chernozemic order - Orthic Dark Gray Chernozemic Soil.

OBIWAN's (Ontology Based Informing Web Agent Navigation[4] [59]) approach is to distribute the various information sources. The idea is similar to Web rings[5] [22]: Websites are clustered into regions. The clustering criterion may be geographical location (e.g., all websites in Kansas), content (e.g., all sites related to Canadian soil), specific quality criteria (e.g., speed of growth), or combinations thereof. Regions may overlap. They are clustered into super regions, and super regions can be grouped into hyper regions, etc., building a hierarchy of *regions*.

### Local information retrieval

Searching and browsing on a local basis (i.e., one particular site) is done by using a hierarchy of *concepts*. The information retrieval process for regions is described in the next paragraph.

The hierarchy of concepts, or rather ontology, is based on a publicly accessible browsing hierarchy. In this case, the Magellan[6] hierarchy has been mirrored, which is comprised of approximately 4,400 nodes. The nodes of the ontology are labelled with the names of the nodes in the browsing hierarchy. The semantics of the edges of this hierarchy are not specified; in most cases, they correspond to a specialization relation (super-/subconcept). Figure 1.1 shows an excerpt from this hierarchy.

Each node of the browsing hierarchy is associated with a set of documents which are used to represent the content of this node. All of the documents for a node (in the experiments, 10 documents per node) are merged into a superdocu-

[4] www.ittc.ukans.edu/obiwan
[5] e.g., www.webring.org
[6] magellan.excite.com

```
4062 Sports
     ...
     4248 Recreation
          ...
          4314 Water-Sports
          4315 Boating
               4316 Boat-Manufacturers
               4317 Boat-Shows
               ...
               4326 Yacht-Clubs
          4327 Canoeing
          4328 Fishing
          4329 Fishing-Links
               4330 Fishing-Locations---International
               4331 Fishing-Locations---North-America
               ...
               4337 The-Tackle-Shop
          4338 Hydroplane-Racing
          4339 Kayaking
          ...
```

Figure 1.1: Excerpt from the Magellan Ontology

ment. The *tf·idf* weights (see section 1.1.1) for the superdocument are calculated to yield a single vector which describes this specific node. These vectors are precalculated using an indexing process.

A website can be *characterized* with respect to this "standard" ontology (nodes consisting of labels and their content in form of weighted keyword vectors). Each webpage for a specific site is spidered, and its *tf·idf* weights are calculated. The webpage vector is then compared with the ontology node vectors to locate the top matching node(s) or categories. [59] contains a detailed discussion of the algorithm as well as a comparison with other classification approaches.

The characterization process adds the similarity weights for the top nodes

across all pages for a specific website. This yields a weighted ontology that relates nodes in the ontology to

- *weights* representing the degree of how much the content of the site can be described by this particular subject, and to
- *documents* of this site that are related to this subject (i.e., qualitatively) together with the a measurement of *how much* the content of each document is described by this subject node (i.e., quantitatively).

This categorization process may be done regularly, e.g. once a week or a day, to ensure that the site characterization remains up to date.

Browsing this site is done by clicking on the nodes of the standard ontology, i.e., the subject hierarchy, which results in displaying the pages of the site that correspond to that subject, together with a measurement of how much they are related to that subject. This clickable hierarchy may also be visualized in three dimensions [16].

Top search a site, all of its documents indexed (with no regard to the characterization). Documents are retrieved by choosing the maximum $tf{\cdot}idf$ values of all documents for the query terms (or rather the sum over all query terms).

Figure 1.2 graphically represents local information retrieval. The personalization module can be used for both re-ranking documents for browsing purposes and re-ranking/filtering search results.

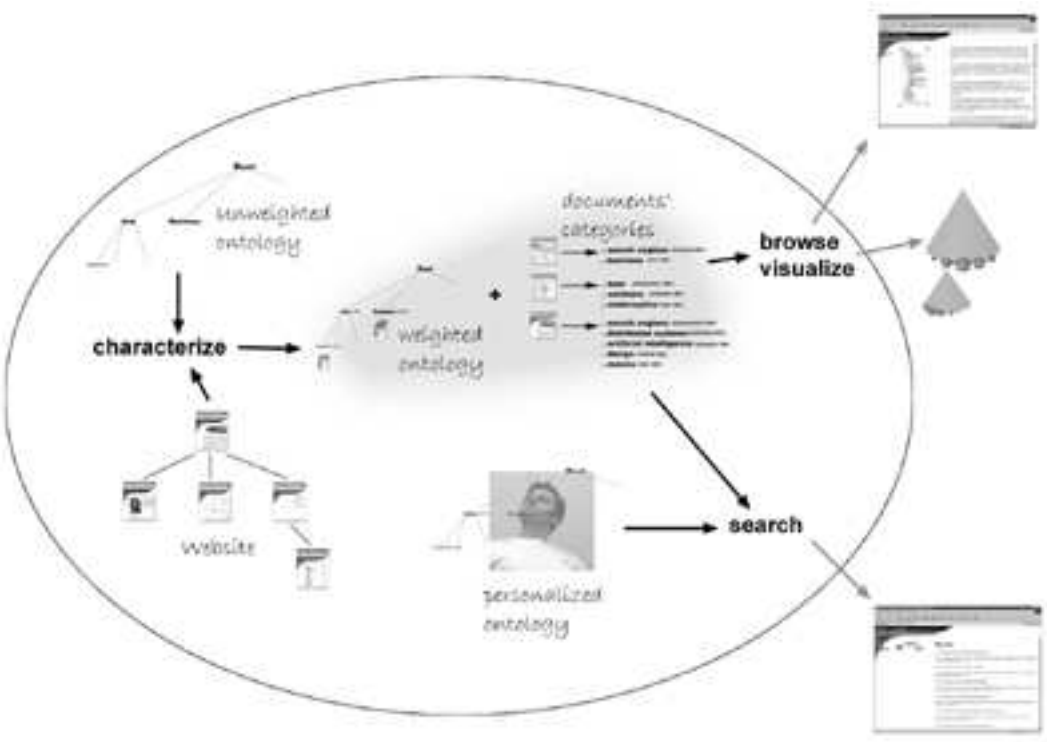


Figure 1.2: Local Information Retrieval in OBIWAN [14]

### Regional Information Retrieval

Up to this point, information retrieval has only been done within one single site. When multiple sites are clustered into one region, the search and browsing process are distributed.

Every region contains a sitemap which maps every site in this region to the subjects this particular site covers. When a query is entered, it is first categorized, i.e., the subjects related to this query are determined. The sitemap is then used to find the most promising sites for this query by matching the query's categories against the different sites (or rather the subjects that are covered by every site). The query is then brokered to these most promising candidates. The results are retrieved and after merging, they are presented to the user.

Browsing is done in a similar manner. Browsing a region means browsing the "best" sites related to each subject simultaneously (by merging the pages from the "best" sites).

Here again, the personalization module can be used for re-ranking or filtering the documents associated with a node in the browsing case, or it can be used for re-ranking or filtering search results.

Figure 1.3 shows how the regional information process works. From a conceptual point of view, there is no difference between retrieving information from a region and retrieving information from a super region, i.e., a cluster of regions. In principle, an arbitrary number of levels of regions can be established.

## 1.2 Contributions

This thesis provides a new method of modeling user interests that has not been investigated before. Its most important characteristics are the profiles' structure, a hierarchy consisting of 4,400 nodes, and the lack of a need for explicit user feedback to populate them. A novel system that maps Web browser caches to user interests has been written, is operational, and in daily use.

Further contributions include the discovery that, when using read documents to model a user's interests, the length of these pages does not matter as much as the time spent on that page. Four mapping functions for interest modeling are investigated, two of which exhibit desired characteristics such as profile convergence (defined in section 3.2.2) and performance improvements in modifying

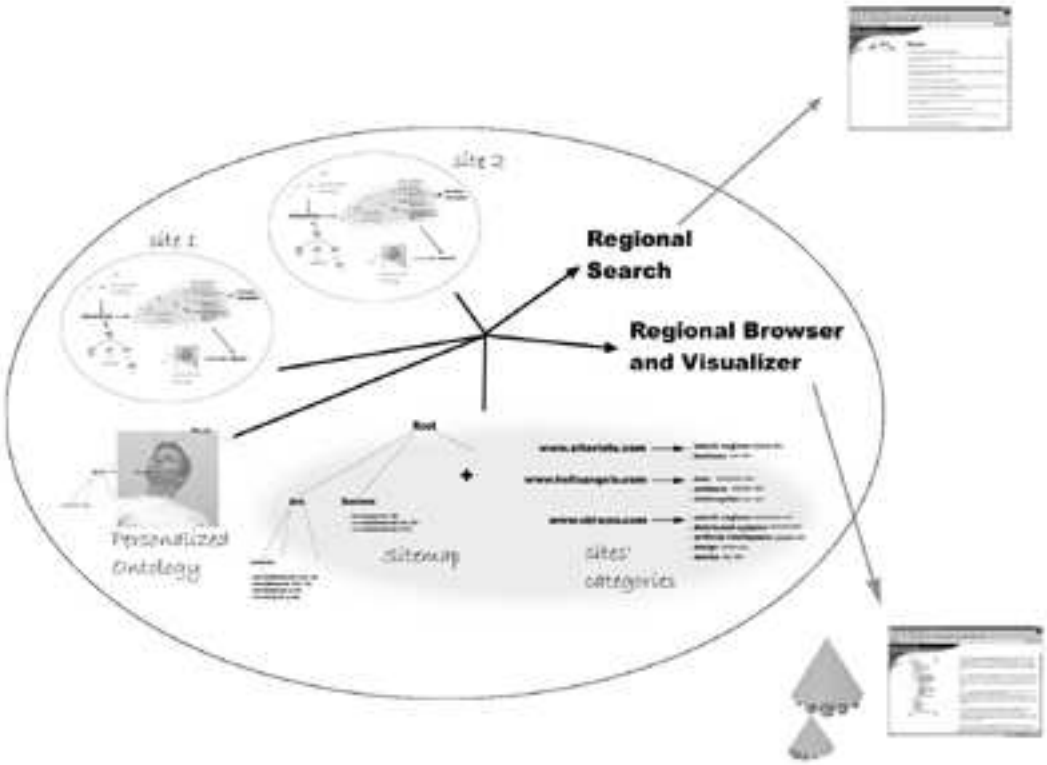


Figure 1.3: Regional Information Retrieval in OBIWAN [14]

search results. Re-ranking or filtering results that have been returned by a popular search engine (ProFusion) are only one possible application of deploying the user profiles. Several algorithms to re-rank and filter search results have been implemented. Together with four different ways of modeling user interests, this yields 20 different ways of re-ranking and 120 ways of filtering.

Unlike most personalization systems that have been presented in the literature, this thesis includes a thorough evaluation of the profiles and their application. The algorithms for evaluation have been implemented and can be reused for similar tasks in information retrieval.

# Chapter 2

# Personalization

Soon after the WWW emerged, work on personalizing the access to or views of the Web began. This chapter gives an overview over existing *systems* and *approaches* to personalization developed over the last several years. Due to the large number of systems, this chapter is necessarily incomplete, but well-known representative systems are described.

In order to structure the wealth of approaches to personalization, the discussion will be organized according to the following orthogonal dimensions:

- **application**:

  For what are the user profiles used? Application fields can broadly be divided in *personalized access* to certain resources (personalized "portals" to the Web, filesystems) and *filtering/ranking* issues: electronic newspapers, Usenet news, recommendation services (browsing, navigation), and search.

- **creation and representation of the user profiles**:

  What data is used to build the profiles? How are they built, i.e., what

is the learning mechanism (if any)? How are they stored - structured or unstructured?

  - **data source for user profiles**:

    *What* is learned, i.e. how is the user profile obtained? More precisely, does the system learn *implicitly* by observing the user's behavior, or does it learn *explicitly* by requiring the user to enter her interests [44]?

  - **learning algorithms**

    Once information on a user is gathered, *how* is it used to build the profile? Is the system adaptive in that the profile changes over time, hopefully adjusting to a user's actual interests? Examples for learning algorithms are probabilistic algorithms, genetic algorithms [33], and algorithms working in the vector space model [48]. [35] contains a detailed bibliography for all of these approaches in the context of text learning.

  - **representation of user profiles**:

    How are the interests of a user stored? Common representations include Boolean or weighted keyword vectors, semantic nets, n-grams, and keyword vectors for a small number of categories.

- **rating and filtering algorithms**:

  Which algorithms are used to decide whether or not a user is interested in a particular item? In other words, how is the matching of a document with a user profile done?

- **collaborative vs. individual filtering**:

Does the personalization and/or filtering process focus on *one* user, or is it also concerned with a community of users (*collaborative* filtering)?

- **architecture**:

  For collaborative and search issues, does the user profile reside on the *server* side, or is it local to the *user's* machine? A possible partitioning of this dimension is "agent" and "non-agent" systems, but there is no agreement on how to use this word, so this distinction is not considered in this survey.

The following sections present the systems grouped together by their application. Each of the above dimensions will be discussed. Discriminating features concerning the other dimensions will be presented together with their discussion in these brief presentations.

If the description of a product does not include a discussion on one of the dimensions, the discussion of that is also omitted in this survey.

Section 2.2 contains a summary in tabular form.

Whenever possible, references to comparisons of products in one of the categories are given. [40] is a compilation of freely available information filtering systems (some of which will not be discussed here), and [23] is an early approach to categorizing Usenet news filtering systems. [34] and [35] contain a more recent discussion on some of the systems, and [9] gives an overview on other "intelligent" information brokering systems. Finally, [41] contains a thorough discussion of trends in text filtering, in particular with respect to personalized approaches.

## 2.1 Applications of Personalization

Applications are coarsely divided into two areas: personalized access to some resources, and approaches involving filtering.

### 2.1.1 Personalized Access

As the Web continues to gain popularity, it is not surprising there do exist commercial providers for personalized information systems. Examples include **Pointcast**[1] and **InfoQuest**[1]. These programs provide the user with a desktop containing links to different sources of information (news, weather, stock market, televison programs and the like), and they allow for (explicitly) specifying topics of interest to the user.[2] The profiles here consist of a simple list of words or subjects. A very similar approach is implemented in the personal MyYahoo section of the popular search engine **Yahoo**[1]. Recently, this kind of service has been dubbed "portal".

Popular Internet browsers such as Microsoft's Internet Explorer or the Netscape Navigator allow for organizing bookmarks in a personalized manner. An early approach to personalization in this direction is the **PAINT** system [42] which views the Internet as a file system and helps in personalizing views on it. PAINT considers the navigation problem a name space management problem and therefore aims at personalizing this namespace. **BASAR** [54] assists users in managing their personal information spaces by updating links (in the bookmarks) and deleting links that are seldom or never used.

[1] URLs: *Pointcast:* www.pointcast.com, *InfoQuest:* www.inforian.com/quest, *Yahoo:* www.yahoo.com

[2] The techniques used for filtering are very simple versions of the ones used by the systems described in section 2.1.2

Finally, personalization is very common in the area of e-commerce, where a user explicitly wants the site to store information on her, such as credit card numbers and/or addresses (e.g., **Amazon.com**[3] which also regularly sends information on new books of interest (based on a list of categories a user enters) or **eBay**[3]) as well as user preferences such as "no frames" or "text only" (in the **Personal Wall Street Journal**[3]). This kind of information is typically stored in form of cookies [31]. Amazon and e-Bay are collaborative systems in the following sense: They allow for assessing books or vendors, respectively, and this collected information is visible to every user. **Firefly**[3]is a provider for personalized information systems, featuring customizable versions of MyYahoo, personalized movie recommendations, finding people with similar interests, or applications in e-commerce, e.g. **Barnes and Noble**[3]. User interests are determined by keywords, and later on, by reviews they write. This allows for personalized delivery of book recommendations and other information services.

### 2.1.2 Filtering and Rating

Filtering and rating seem to be the main focus of research in personalization. This section presents personalized newspapers, Usenet news filtering systems, recommendation systems for browsing and navigation, and search.

#### Newspapers

The electronic version of the **Personal Wall Street Journal**[3] allows for personalization in a similar way as do Yahoo or Pointcast.

[3] *Amazon:* www.amazon.com, *eBay:* www.ebay.com, *Wall Street Journal:* www.wsj.com, *Firefly:* www.firefly.net, *Barnes and Noble:* www.barnesandnoble.com

*Information sources:* The user interests have to be provided explicitly (by clicking on categories of interest), or they are inferred from a user's stock portfolio - the Wallstreet Journals proposes links or articles to follow which are related to the shares contained in the user's portfolio(s),[4] a somewhat more "intelligent" approach to personalization. Indeed, this seems to be a clever approach since the shares of a user will naturally reflect her interests!

*Learning algorithm, profile representation*: The underlying technology is not documented, but it is reasonable to assume that the companys' names are labels of classes, and these classes are presumably defined (trained) by words occurring in articles concerning a particular company. Concerning the change of user profiles over time, there seems to be no learning process (except when portfolios are updated).

*Architecture*: No explicit information is disclosed, but a list of cookies used by this site does not exhibit any cookie containing portfolio information. It seems reasonable to assume that this information is stored on the server's side.

Another electronic newspaper, **Fishwrap** [11] (the technology is used within the electronic version of the San Francisco Chronicle[6]), is quite similar in that it allows for choosing topics of interests and customizing the layout of the personalized news page.

A somewhat different approach was chosen for **Krakatoa** [20] and its successor, **Anatagonomy** [46], in that these products infer the user profiles from the user's behavior. The presentation of the articles "can be personalized in terms of contents, layout, media ..., advertisement, and so on"[20].

[4]Yahoo's *investment challenge* offers a similar service.

[6]http://www.sfgate.com

*Information sources:* An initial profile may be provided in form of a list of keywords. The user behavior is tracked while he reads: activities like scrolling, peeking at, maximizing, opening articles in new windows, or saving them to a scrapbook probably mean a user is interested in that article. Explicit feedback is also supported, and it turns out that, not surprisingly, explicit feedback yields better results than implicit feedback, and a combination of both clearly yields the best results w.r.t. recall and precision[7].

*Profile representation*: No explicit information is given, but the article suggests the profile is a list of weighted keywords (this is indicated by the fact that an initial profile can be given in form of a list, and that the user profiles and documents can be compared easily).

*Collaborative vs. individual*: Anatagonomy supports both collaborative as well as individual filtering.

*Architecture*: The user profiles are stored at the server's side (without this, collaborative rating would be much more difficult, but there are, however, privacy concerns).

[47] pushes the personalization a little further in that it assumes a user visits the newspaper several times a day and would probably like newer ("fresher") information to be visibly distinguished from known articles. This is done by putting "fresh" articles at exposed positions such as the top of the article list.

The **SmartPush** System [25] is used for information delivery of economic data as provided by a major Finnish newspaper.

*Profile representation:* An initial profile can be provided by ranking sample

[7]Performance evaluation in terms of recall and precision is undertaken in ProFilter [8], too.

documents, giving a list of keywords or choosing among a set of default profiles. The profile is stored in the form of a concept hierarchy, or rather ontology.

*Rating and filtering:* Documents are augmented with ontologies similar to the user profiles, and they are created by hand by the document's author. These meta-data describe the *content* of a document and are attached to the latter. According to the authors, the ontologies will eventually reach a size of 600 nodes (40 as of March 1999). The matching process then becomes slightly more complicated, since in this case distances between weighted hierarchies have to be calculated rather than distances between vectors ([49] proposes an asymmetric distance measure).

*Learning algorithm:* [25] emphasizes the adaptive nature of SmartPush, but no information is given on how implicit and explicit feedback actually is provided.

*Collaborative vs. individual:* At present, SmartPush is an individual system, but future versions are envisioned to support collaboration.

**WebMate** [10] spiders a URL the user wants to be monitored, typically pages that contain many news headlines such as the homepage of NewsLinx[8]. The articles associated with headlines are fetched and compared to the user's profile, resulting in a personalized presentation of news.

*Profile representation:* The system stores documents as weighted keyword vectors and clusters them. These clusters are then automatically labelled with the "most important" word and are assumed to represent one domain of a user's interests. The profile consists thus of the cluster centers together with their associated documents the number of which is bounded to save space.

*Learning algorithm, information sources:* There is some evidence the system relies on explicit feedback. The learning is done by adjusting the cluster centers

[8] www.newslinx.com

as new documents are stored.

*Individual vs. collaborative:* WebMate is an individual system.

WebMate is an integrated tool which also provides assistance in searching by expanding queries (unpersonalized).

### Usenet News

Taking into account the number of Usenet news filtering systems, this kind of information system clearly exhibits a need for personalization. This conforms to daily experience: With a hundred or even more articles per group, it is impossible to read them all.

The following briefly presents the systems NewT, SIFT, PSUN, and GroupLens (as these seem to be good representatives for the different classes of news filtering software). There are many other news filtering systems (e.g., **NewsWeeder, Browse, NewsClip, Lurker, Smart, Borges, InfoScan, RAMA, Pefna**, and **InfoScope**) which can be found in [23] or [40].

**NewT**'s [50, 51] personal profiles are initially provided by the user in form of a list of keywords. Whenever presenting a filtered article to a user, the latter decides by explicit relevance feedback if he liked or disliked this article. This feedback is then used to modify the profile.

*Source of information, profile representation:* Profiles are stored as vectors of weighted keywords (and so are the documents). Explicit user feedback is the input to the learning algorithm.

*Learning algorithm:* A user's interests are learned by means of a genetic algorithm. Several instances of the user profile (called agents) compete with each other, and an agent is rewarded when the user liked a suggested document (and

punished when the user disliked it). The common techniques of crossover and mutation then yield a generation of agents that eventually represent a user's interests suitably.

*Rating and filtering:* Documents are compared with the user profiles using the cosine measure in the vector space model.

*Collaborative vs. individual:* Clearly, NewT is an system focusing on one individual.

In **SIFT** [57], filtering is done by comparing articles to an individual user's static profile. SIFT is the representative of the earliest class of news filtering programs which use the same profile representation and exhibit no adaptivity or learning component.

*Profile representation, Rating and filtering:* The profile is represented by a Boolean or weighted vector of keywords. Matching of the profile and an article occurs w.r.t the cosine similarity of the vector space model.

*Collaborative vs. individual:* SIFT's focus is on individual users.

**PSUN** [52] differs from the previous systems in the representation of the profiles and the learning technique.

*Profile representation:* Profiles are provided initially by presenting the system with some articles a user finds interesting. Recurring words in these are stored by means of *n-grams* ($n$ words found to occur after each other a significantly high number of times and thus providing some context), and the n-grams are stored in a network of mutually attracting or repelling words, the degree of attraction being determined by the degree of co-occurrences/ Different user profiles are then stored in a way similar to Minsky's K-lines [32], connecting n-grams of different

weights. Each user has multiple profiles that compete via a genetic algorithm.

*Source of information:* Explicit feedback is needed for the learning algorithm.

*Learning algorithm:* The user profiles consisting of K-lines-like connections of weighted n-grams compete with each other. The usual operations in genetic algorithms then eventually lead to a generation of profiles that represent the user's interests accurately. (Since this is a particularly original approach, it is regrettable there is no evaluation).

*Collaborative vs. individual:* PSUN aims to support single users.

**GroupLens**[9] [24] is different from the previous approaches in that it allows for implicit rating and is an exclusively collaborative filtering system.

*Information sources:* Quality assessments of articles are based on explicit feedback and the time a user spent on a page (an approach also investigated in [37, 39, 41] and, with some modifications, implemented in this thesis. This quality assessment technique issue is discussed in some depth in chapter 3.

*Collaborative vs. individual:* GroupLens is not suited for individual personalization (and can therefore be seen as a recommendation service as discussed below).

A more recent system is **Alipes** [56] which allows for explicitly modeling *dis*interest in a particular field, and can be used for both searching and news filtering tasks.

[9] www.cs.umn.edu/Research/GroupLens

### Recommendation Services

Recommendation services usually suggest that a user follow a link on a page he is currently visiting (or suggesting that he *not* follow it). This recommendation is based on the user's interests.

In this section, the following systems are presented: Amalthaea, ifWeb, FAB, Letizia, SiteIF, Siteseer, Syskill and Webert, WebWatcher, and Personal WebWatcher.

**Amalthaea** [38] is exploring personalized data discovery and information filtering. The Web is searched for documents that might be of interest for a user, and the user profiles are also used for news filtering.

*Profile representation and Information sources:* Initially, the user provides Amalthaea with a list of keywords reflecting her interests. The profile is stored in form of weighted keyword vectors. Explicit feedback is given by the user to decide if she liked or disliked the documents presented by Amalthaea.

*Learning algorithm:* Learning is done by means of genetic algorithms which compete in representing a user's interest most accurately. Eventually, the fittest class of algorithms will suitably represent the user's interests. Amalthaea can be bootstrapped with new genetic algorithms (profiles, list of keywords) that are explicitly provided by the user.

*Rating and filtering:* The quality of a document in terms of the user's interest is assessed by calculation of cosine similarities in the vector space model.

**ifWeb** [2] supports two modes: navigation support and support in document search.

*Profile representation and Information sources:* User profiles are stored in the form of "weighted semantic networks". These semantic networks differ from those in the knowledge representation domain, since they represent terms and their context by linking nodes (words) with arcs which represent co-occurrences in some documents. The authors claim that ifWeb supports implicit feedback, but their description lacks any mention thereof. The author was unable to verify this. In addition to the unconventional method of representing profiles, ifWeb is, however, interesting for two other reasons: It takes into account not only interests, but also explicit *dis*interest, and therefore presumably reflects a user's idiosyncrasies more accurately. Secondly, it incorporates a mechanism for temporal decay, i.e., ages the interests as expressed by the user.

*Rating and filtering:* No details are disclosed, but evaluations of the personalized orderings of some search results by means of the *ndpm* comparison [58] exhibit a good performance of the system.

*Collaborative vs. individual:* ifWeb focuses on individual users.

**FAB** [3] is a collaborative recommendation service and succeeds the **LIRA** system [4].

*Profile representation and Information sources:* User profiles are stored in form of weighted keyword vectors and updated on the basis of explicit relevance feedback. Documents and user profiles are matched according to the cosine similarity in the vector space model.

*Learning algorithm:* As stated above, profiles are updated w.r.t. explicit user feedback. FAB implements a (temporal) aging function for a user's interests.

*Collaborative vs. individual:* As already stated, FAB's focus is on collaborative filtering. A central repository of recommended documents is (automatically)

updated with documents that are recommended by a user who exhibits interest in the document (whose interest profile matches the document). User profiles are compared on the basis of the cosine similarity, too.

*Architecture:* User profiles seem to be stored at the server's side (which seems inevitable in a collaborative system).

**Letizia** [29, 28] assists a user when browsing by suggesting links that might be of interest and are related to the page the user currently visits.

*Profile Representation:* No explicit information is available, but since the documents to be matched with a profile are stored as a weighted keyword vector, it is reasonable to assume that the user profile is a weighted keyword vector as well.

*Information sources:* Letizia relies on implicit feedback: Links followed from the currently visited page are assumed to reveal interest in the document containing the link. Bookmarking a page also means this page is interesting. Furthermore, as (Western) users tend to read from the top left corner to the right bottom corner, links that are omitted during the reading process might express disinterest in the referenced document.

*Rating and filtering:* There is no ordinal scale for the importance of suggested links but rather a (cardinal) preference ordering. It is reasonable to assume the filtering mechanism involves cosine similarities in the vector space model since the documents to matched are stored as weighted keyword vectors.

*Collaborative vs. individual:* Letizia is for individual use. **Let's Browse** [27] extends Letizia for collaborative use.

**SiteIF** [53] strongly resembles ifWeb, except that explicit user interaction is avoided. The cited paper does not contain information on actual deployment

of the system, so its current objective seems to be gathering and maintenance of user profiles.

*Profile representation:* As in ifWeb, profiles and documents are stored as semantic networks (terms and correlated terms, their context). SiteIf also involves a decay function for aging user interests.

*Information sources:* The profile is built in terms of the links followed by the user.

*Architecture:* Since users must enter a login name and a password is required, the author assumes that profiles are centrally stored.

*Collaborative vs. individual:* Like ifWeb, SiteIF is concerned with individual users.

**SiteSeer** [45] is another collaborative webpage recommendation system.

*Information source, Profile representation:* User profiles are extracted from their bookmark files, taking into account the *content* of the referenced documents, and the *structure* of the bookmark file. The folders in the bookmark files are used to identify the user's categories of interest. Even though there is no technical information on how the representation is done, it is said that the system does not derive any semantic value from the content of the stored URLs. The profiles thus seem to consist of a list of URLs together with their structure.

*Collaborative vs. individual:* SiteSeer is a purely collaborative system. Recommendations occur when the profiles (as derived from the bookmarks) of two users match in terms of the URLs contained therein (and thus measuring the overlap), by giving additional weight to URLs that do not occur frequently, an approach similar to the *tf*idf* approach for content determination of documents [48].

*Architecture:* Being a collaborative system, the user profiles are most likely

stored on a central server in order to allow for matching users.

**Syskill and Webert** [43] allows for both personalized search and recommendation (navigation). Search results, returned by Lycos, are annotated with symbols reflecting the assumed interest (good, okay, don't know, poor). In the recommendation mode, the system suggests links to follow on "index pages" which contain many links related to a given topic. Recommendation is done as in the search mode by graphically annotating the links on the index page. Examples for index pages are the pages contained in Yahoo's Browsing hierarchy or many overview pages in the WWW Virtual Library.[10]

*Information sources:* Syskill and Webert relies on explicit user feedback on a three point scale.

*Profile representation:* A user's interests are divided into classes (which simply coexist; there is no hierarchical relationship between classes). These interest classes describe the content of the index page, the links contained in which will be annotated later. Within each class, the profiles consist of boolean keyword vectors.

*Learning algorithm:* A thorough investigation[11] of which learning algorithm to choose resulted in choosing a naïve Bayes Classifier. Classification is done w.r.t. to the different categories of a user's interests. It is found that for good classification results, it is not necessary to characterize whole documents, but that the first 96 words of a document are sufficient[12]. Interestingly, this yields better results than

[10] http://vlib.org, a good example for an index page is the complete index for medicine related issues: http://www.ohsu.edu/clinicweb/wwwvl/all.html

[11] The discussion in [43] compares Bayesian classifiers, Nearest Neighbors, PEBLS, Decision Trees, *tf*idf*, and Neural Nets. [6] focuses on probabilistic user models. Text Learning techniques in this context are discussed in [35, 59].

[12] A similar observation is made in ProFilter [8].

working with entire documents.

*Rating and filtering:* Rating is done by classifying the documents w.r.t. to the user profile and determining the degree of membership.

*Collaborative vs. individual:* Syskill and Webert is an individual system.

**WebWatcher** [1, 19] is a popular browsing assistant. For a particular site, WebWatcher takes the role of a museum guide, pointing the visitor to interesting documents.

*Profile representation:* The interests of a user are given at the beginning of the tour in form of a list of keywords (and therefore represent rather a "goal" than an "interest").

*Information sources, Learning algorithm:* Individually, the interests of a user are given at the beginning of the tour. No further learning takes place w.r.t. to a user's profile. In terms of collaboration, all hyperlinks are annotated with the profile (the goal in form of keywords) of the user who followed them. WebWatcher uses reinforcement learning to associate links with the content of the underlying documents. This aims at finding "paths through the Web which maximize the amount of relevant information encountered" [19].

*Collaborative vs. individual:* WebWatcher combines collaborative and individual aspects. Whenever a user selects a link, his interests (in form of some keywords) are annotated with that link. This information is subsequently used in the recommendation process by matching the annotation with the interests as expressed by the user. Personal WebWatcher is an adaptive version of WebWatcher.

*Architecture:* User profiles are rather a goal for one browsing session, and this goal is stored at the server's side. The same is true for the collaborative implicit feedback (which links were chosen).

**Personal WebWatcher** [34] augments WebWatcher with adaptive behavior towards one user. It is thus a recommendation service, too. The suggestions are restricted to links that already exist on a page, and if the system considers them interesting, these links are highlighted.

*Information source:* To build and update profiles, Personal WebWatcher uses the content of links that have been followed as examples for interesting pages, and links that have not are considered boring.

*Learning algorithm, Profile representation:* Learning is done by a naïve Bayes classifier where the documents are represented as weighted keyword vectors, and the classes are "interesting" and "not interesting". The profile is then described by these two classes with the associated sets of documents (their vector representation).

*Rating and Filtering:* Bayesian Classification is used to distinguish between interesting and uninteresting pages.

*Collaborative vs. individual*: Personal WebWatcher is an individual system related to the collaborative WebWatcher.

References to other collaborative browsing assistants such as **Firefly**,[13] **Webhound** and **Ariadne** can be found in [27]. Since they are similar to other approaches, their description is omitted here.

### Search assistance

**ProFusion Personal Assistant** [8] is a filtering tool for results returned by the meta search engine ProFusion [15]. It decides which results to present to the user

[13] http://www.agentsinc.com

and which to discard. This judgement is done for the results of queries that are *re*submitted regularly.

*Information source:* Explicit relevance feedback is used to determine the areas of interest.

*Profile representation, Learning algorithm:* User profiles are stored as sets of two classes of documents: interesting and rather boring ones. Documents are stored as weighted keyword vectors, and for both classes, every term is assigned a weight representing its membership to "its" class. Explicit feedback updates the two classes by simply adding the document to its class and possibly modifying the weights of the occuring terms in both classes.

*Rating and filtering:* For each term in the retrieved documents (or rather their summaries), its weight in the irrelevant set and in the relevant set are used to assess how interesting the document is. This is done by calculating similarities with the two classes in the vector space model.

*Architecture:* The user profile is stored on the server's side.

**PEA** [36] is similar to Syskill and Webert in that search results are augmented with icons indicating a possible interest of the user. PEA is intended to work on top or together with other personalization services.

*Profile representation, Information sources:* Profiles are essentially bookmark files, similar to Siteseer. Different folders represent different classes of interest. Documents contained in these classes are stored as weighted keyword vectors. PEA also allows for adding interesting search results to an index which initially contains the bookmarks.

The system described in [**31**] re-ranks search results rather than filtering them.

*Profile representation, Information Sources:* Profiles are stored as weighted keyword vectors. These vectors contain the frequencies of all the words occuring in a user's entire filesystem, and therefore, no explicit feedback is required. No adaptation takes place.

*Rating and filtering:* For all documents (URLs) that were returned by a search engine, every word contained in them is looked up in the profile. If it exists in the user profile, its weight in the retrieved document is added to the URLs score. This yields a new personalized ranking.

*Architecture:* Profiles are stored on the client machine.

As **ifWeb** [2] and **Syskill and Webert** [43] are both recommendation services and personalized search engines, their characteristics were discussed on pages 30 and 34, respectively.

*Rating and Filtering:* Concerning ifWeb, it is not clear if the personalization process is done by filtering or re-ranking of the returned results. Syskill and Webert annotates search results graphically, in a way similar to PEA.

### 2.1.3 Other

This section presents systems that do not fit in one of the other categories: expertise location, e-mail filtering, and machine-dependent link annotation.

The system described in **[55]** exhibits a quite different form of personalization. Its aim is to find experts in a given field, e.g., the JAVA programming language.

*Profile Representation and Information Source:* Profiles are built by scanning a user's JAVA source code and storing the classes and/or constructs he uses in

form of weighted keyword vectors.

*Rating:* Users in need of an expert submit their query which is then matched with all user profiles. The person with the presumably best knowledge to answer this question is determined by calculating a cosine similarity between the query and all user profiles.

Other similar projects use papers written, emails and citations to determine the field of expertise of a particular user [21].

**Information Lens** [30] is a tool for filtering and ranking e-mails.

*Profile representation:* Profiles are stored as rules on structured lists of keywords, where the structure is determined by the components of mails: sender, subject, etc.

*Information source, Learning algorithm:* Rules have to be built by hand.

**WBI** [5] provides the user with rudimentary browsing assistance by recording his entire surfing history (as done in the most recent versions of Netscape Navigator and Microsoft Internet Explorer) and thus allowing for shortcuts. An interesting feature is the annotation of links with "their" network speed or download time.

Finally, document filtering systems of various kinds can be perceived as personalizing systems. So-called **cybersitters**[14] allow parents to enumerate categories or rather keywords that should not be contained in documents their kids retrieve. SurfWatch[15] allows companies to restrict the Internet access of their employees. As in the case of the cybersitters, the goal of this product is to "block objectionable

[14] e.g. www.solidoak.com
[15] www1.surfwatch.com

sites".

## 2.2 Summary

This chapter summarizes the described systems in tabular form. The first column contains the **name of the system**, if available, and the second column briefly describes its **application, or purpose**. The third column gives a brief description of some **technical internals**, such as how the profile is built, and what data it is built on. Whether a system is **adaptive or static**, i.e. if a profile changes over time or not, is indicated in the fourth column. The fifth column indicates how the **matching process** of a profile with a document is done, and finally, the sixth column indicates if a system is a **collaborative or an individual** one.

| **System** | **Application** | **profile: what+how?** | **ad./ st.** | **rating model** | **coll./ ind.** |
|---|---|---|---|---|---|
| **Alipes** | news and search | cat. of interest, keyword vectors, expl. feedback | ad. | based on cosine sim. | ind. |
| **Amalthaea** | data disc. + news | keywords, expl. feedback, genetic alg. | ad. | cosine sim. | ind. |
| **Amazon** | e-comm. | cr.card#, book rev.+ ass., stored as keywords | st. | | both |
| **Anatagonomy** | newspaper | browsing beh. | ad. | | both |
| **eBay** | e-comm. | vendor ass., stored as keywords | st. | | both |

| System | Application | profile: what+how? | ad./ st. | rating model | coll./ ind. |
|---|---|---|---|---|---|
| **BASAR** | bookmarks | URLs+their usage | ad. | | ind. |
| **Borges** | news retrieval | keywords, based on SMART | st. | cosine sim. | ind. |
| **Browse** | news | articles read or not read, neural network | ad. | sim. of pairs of words | ind. |
| **FAB** | recomm. | pages, weighted keywords, explicit feedback | st. | cosine sim. | coll. |
| **FireFly** | multiple | | both | | both |
| **FishWrap** | newspaper | keywords | st. | | ind. |
| **GroupLens** | news | pages + time spent, expl. feedback | ad. | | coll. |
| **ifWeb** | nav. + search | pages, expl. feedback?, sem. networks | ad. | | ind. |
| **InfoQuest** | pers.acc. | interests stored as keywords | st. | | ind. |
| **Information Lens** | e-mail filtering | mail components, hand built rules to connect them | st. | | ind. |
| **InfoScope** | | | | | |
| **Krakatoa** | newspaper | browsing beh. | ad. | | ind. |
| **Letizia** | recomm. | pages, links followed, keyword vectors? | ad. | cosine sim.? | ind. |

| System | Application | profile: what+how? | ad./ st. | rating model | coll./ ind. |
|---|---|---|---|---|---|
| **LIRA** | recomm. | keywords | st. | cosine sim. | ind. |
| **Lurker** | news | rules (boolean conn. of keywords) | st. | | ind. |
| **MS Internet Explorer** | bookm.+hist., portal | URLs, interests as keywords | st. | | ind. |
| **Netscape Navigator** | bookm.+hist., portal | URLs, interests as keywords | st. | | ind. |
| **NewsWeeder** | news | expl. feedback on articles, stored as keywords | ad. | cosine sim. | ind. |
| **NewT** | news | keywords, expl. feedback, gen. alg. | st. | cosine sim. | ind. |
| **PAINT** | bookmarks org. | URLs | st. | | ind. |
| **PEA** | search | bookmarks + their structure, stored as keywords | ad. | cosine sim.? | ind. |
| **Pefna** | news | explicit feedback on articles in different categories | ad. | cosine sim. | ind. |
| **Personal WebWatcher** | browsing ass. | links followed, Bayes class. | ad. | Bayes class. | both |
| **Pointcast** | pers. access | interests as keywords | st. | | ind. |

| System | Application | profile: what+how? | ad./ st. | rating model | coll./ ind. |
|---|---|---|---|---|---|
| **ProFusion Pers. Ass.** | Search | search results, expl. feedback, stored as keyword vectors in 2 classes | ad. | based on cosine sim. | ind. |
| **PSUN** | news | few art. of interests, stored as K-Lines, expl. feedback | ad. | based on n-grams | ind. |
| **SIFT** | news | keywords | st. | cosine sim. | ind |
| **SiteIF** | recomm. | pages, profiles stored as sem. networks | ad. | | ind. |
| **SiteSeer** | recomm. | bookmarks+their structure | st. | bookmark overlap | coll |
| **SmartPush** | econ. newspaper | interest cat. stored as ontology | st. | | ind. |
| **Syskill + Webert** | search + recomm. | pages, explicit feedback, 1 prof. per user interest stored as weighted keywords, prob. learning and others | ad. | cosine sim. | ind. |
| **Wall Str. J.** | newspaper | portfolio, interest cat., stored as keywords | st. | | ind. |

| System | Application | profile: what+how? | ad./ st. | rating model | coll./ ind. |
|---|---|---|---|---|---|
| **WBI** | browsing ass. | visited URLs | ad. | | ind. |
| **WebMate** | newspaper | interest categories learned automatically, explicit feedback? | ad. | cosine sim. with multiple categories | ind |
| **WebWatcher** | browsing ass. | keywords repr. interests/goals, links annotated with prof., reinforcem. learning | st. | | coll. |
| **Yahoo** | pers.access, portal | keywords | st. | | ind. |
| **[31]** | search | local file system, stored as keywords | st. | based on freq. | ind. |
| **[55]** | expertise location | JAVA source codes | st. | | coll. |

Table 2.1: Systems with personalization services

## 2.3 Discussion

Personalization is a very active and broad area of research with many applications. The main applications are

- customizing access to information sources such as articles in newspapers or products,

- filtering news or e-mails
- recommendation services for the browsing process, and
- search.

This chapter introduced a classification methodology and briefly described approximately 45 personalized information systems. Different models of profile representation and learning algorithms were discussed and put in context with their respective application, mainly *rating*, *ranking*, or *filtering*.

Unfortunately, only a few systems evaluate and discuss their results scientifically - as [17] puts it, "... we laud with our hearts, not with our heads." This is in part due to the fact that it actually is hard to determine how well a personalization systems works, as this involves purely subjective assessments.

However, some approaches are discussed. These discussions then include comparisons of different learning algorithms, of personalized orderings vs. non-personalized ones, and discussions of well known measures from IR, recall and precision.

Due to a lack of data, a comparison of the systems with respect to performance is currently impossible.

# Chapter 3

# The User Profile

User profiles store approximations of the interests of a given user (see chapter 2 for alternative approaches). The proposed generation of user profiles differs from the majority of other approaches in that it is

1. hierarchically structured, and not just a list of keywords,
2. generated automatically, without explicit user feedback, and
3. dynamic, i.e. the learning process does not necessarily stop at a given moment in time.

In Section 3.1, this chapter describes how user profiles are generated and maintained. Section 3.2 establishes a notion of convergence for user profiles and investigates if the proposed approach yields convergent profiles. In addition, the overlap between the generated profiles and actual user interests is investigated. Section 3.3 contains implementation details, and Section 3.4 concludes this chapter with a discussion and a brief review of the results.

## 3.1 Creation and Maintenance

Profiles are generated by analyzing the surfing behavior of a user. "Surfing behavior" here refers to the length of the visited pages and the time spent thereon. No user feedback is necessary. It is the author's belief that a system with an explicit feedback mechanism does not encourage the user to deploy such a system – even if a simple assessment "relevant" or "non-relevant" does not take more than a second, it considerably disrupts the user's workflow and is hence annoying.

The Netscape Navigator caches contain data about the last access date as well as the content length of each page in the cache. Two subsequent last access dates are used to determine the time the user spent on the page with the first access date. [24] shows that there is a strong correlation between the time spent on a page and the actual user interest. It is unlikely that a user spends more than ten minutes on one page. If the calculated time exceeds ten minutes, the entry does not count for the user profile. This guarantees that idle times – the user has other work to do, or there is a night between two access times – do not influence the user profile. Since there is a lot of data gathered over a long period of time (i.e., four weeks), one can afford to ignore some possibly relevant pages.

In principle, the profile generation and adaptation work as follows:

1. On a regular basis, store the cached (i.e., surfed) pages in a database. The length of the time intervals depends on the cache size and on how much the user surfs. It is a good idea to store the cached files before they are overwritten. With a cache size of 5 MB, two to three days seem to be sufficient for most users.

2. Whenever a new set of documents has been stored in the database, characterize it (cf. section 1.1.2). Again, characterizing the page means assigning subject areas, or categories, to this page. The strength of the match between the page and the category is also recorded, i.e., how well the content of a page belongs to that category.

3. The top five category matches are then combined with

   - the time a user spent on the page, and
   - the length of the page.

   This yields an update value for the five categories. Currently, weights can only increase: it is not attempted to infer from the browsing behavior whether or not a user *dis*liked a page and the associated categories

Four different combinations of time, length, and subject discriminators are used. In the following dicussion, *time* refers to the time a user spent on a given page, and *length* refers to the length of the page (i.e., the number of characters). Let $\gamma(d, c_i)$ be the strength of the match between the content of document $d$ and category $c_i$.[1] The adjustment of the interest $\iota$ in a category $c_i$, $\iota(c_i)$, will be denoted by $\Delta\iota(c_i)$.

1. $\Delta\iota(c_i) = \frac{time}{length} \cdot \gamma(d, c_i)$.

   This measure takes into account the intuitive idea that the longer a user visits a given page, the more he is probably interested in this page. The denominator reflects the fact that more time is needed to read a longer page.

[1] This value is a result of the characterization process of a page.

2. $\Delta\iota(c_i) = \log \frac{time}{length} \cdot \gamma(d, c_i)$.

   This measure is very similar to the previous one, except for the fact that the interest adjustments are smaller.

3. $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$.

   This measure is similar to the previous one, but the length of a page matters less. [24] shows that there is a strong correlation between an explicit user ranking and the time the user spent on a page. This is to say, time matters greatly - which in this context might also be expressed as "length matters less"!

4. $\Delta\iota(c_i) = \log \frac{time}{\log(\log length)} \cdot \gamma(d, c_i)$.

   This measure takes the time even more into account than the length. In fact, length does not really matter since $\Delta\iota(c_i)$ can be rephrased as $\big(\log(time) - \log\log\log(length)\big) \cdot \gamma(d, c_i)$.

In practice, these measures are modified to guarantee a positive interest value.

## 3.2 Evaluation

The evaluation of the user profiles consists of two parts. Section 3.2.2 introduces a notion of convergence with respect to which 16 actual user profiles are discussed. Section 3.2.3 examines the relationship between the calculated user interests and the actual user interests.

### 3.2.1 Experimental Setup

A group of 16 users have been monitored for 26 days. These 16 users together surfed 7,664 documents (which may contain double counts). Figure 3.1 shows that on average, users tend to spend a relatively short period of time on one page.

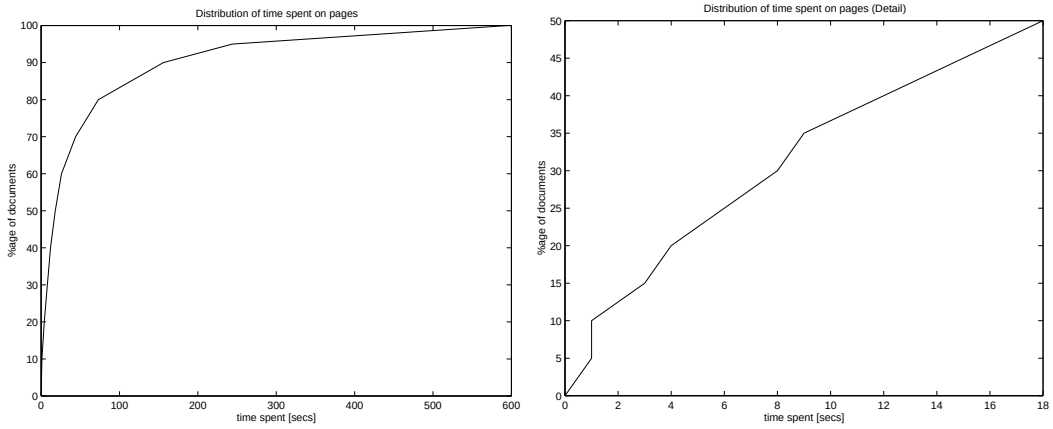


Figure 3.1: Times spent on web pages. 20% of the pages are visited for less than 5 seconds. The right figure is a detail of the left one, ranging from 0 to 50%. Total number of pages: n=7,664.

The mean time is $\mu = 54.59$ with median $\tilde{x} = 18$ seconds. The standard deviation for this set of data is $\sigma = 93.94$ seconds which is rather large, as reflected in figure 3.1.

Figure 3.2 shows the distribution of the pages' lengths. The mean length is $\mu = 7,887$ bytes with median $\tilde{x} = 3,987$ bytes. The standard deviation is $\sigma = 14,633$ bytes which is due to the fact that there are a few very large files (these are very large literature databases).

The subject hierarchy mirrors the Magellan browsing hierarchy. It contains 4,385 nodes. A subset of it is shown in figure 1.1.

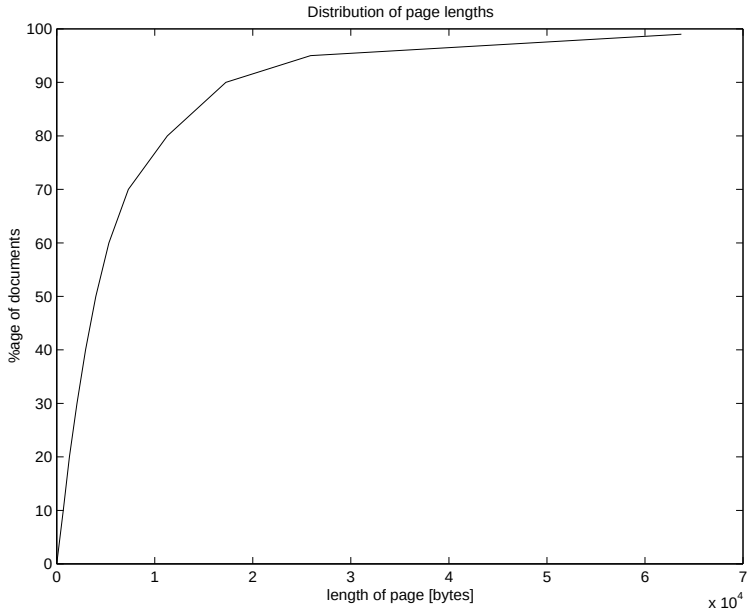


Figure 3.2: Distribution of the length of web pages. 30% are shorter than 2,000 bytes. Less than 1% of the pages are larger than 100,000 bytes; for representational reasons, these are not included in the graph. Total number of pages: n=7,664.

### 3.2.2 Convergence

One would assume that every human has a relatively stable collection of interests which may change over time [26]. Thus, the evaluation of the profiles will be based on a notion of convergence. In the following, a *node* always refers to a node in the subject hierarchy and hence to a category or a subject.

A user profile is said to be *convergent* if the number of nodes with non-zero interest values $\iota$ converges over time. Note that this is not a technical definition since the notion of "convergence" of a set of values is not specified.

Two main factors necessitate a slight modification of this very simple model:

- The characterization process is not necessarily accurate, i.e., there is no guarantee that a document has been classified correctly [59].

- Only pages in the English language can be characterized correctly. If a user

happens to surf non-English pages, the characterization process will usually determine categories that do not describe the page in question very well.

This leads to the adoption of a "noise filter": the above strong characterization is weakened into the following notion of convergence:

A user profile is said to be *convergent* if the number of nodes that account for 95% of the total accumulated interest value, $\sum_{i=1}^{N} \iota(c_i)$, converges over time. $N$ denotes the total number of categories, in this case $N = 4,385$.

Figures 3.3, 3.4, and 3.5 show sample profiles of three users. The number of categories that together account for 95% of the total accumulated interest values is 50, 100, and 150, respectively. Categories are numbered subsequently.

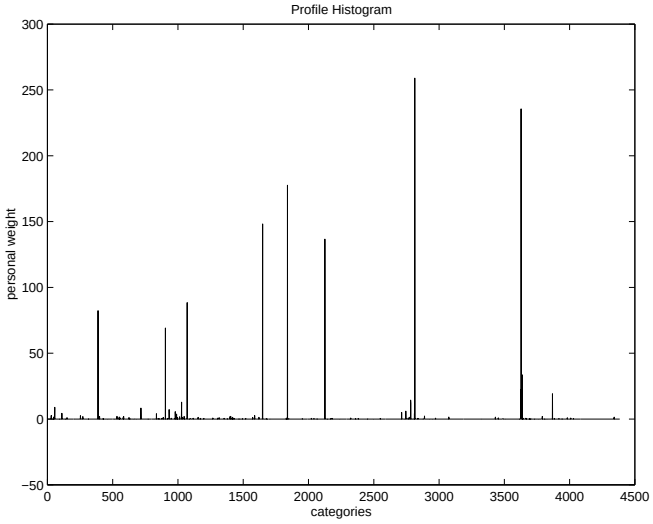


Figure 3.3: Sample user profile: less than 50 categories

These three profiles were created by using the adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log(\log length)} \cdot \gamma(d, c_i)$. Changing the adjustment function leads to very similar histograms where only the number of categories slightly changes.

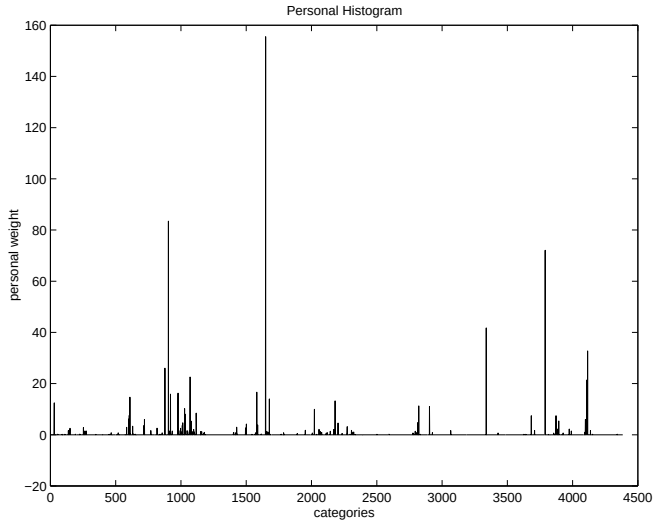


Figure 3.4: Sample user profile: less than 100 categories

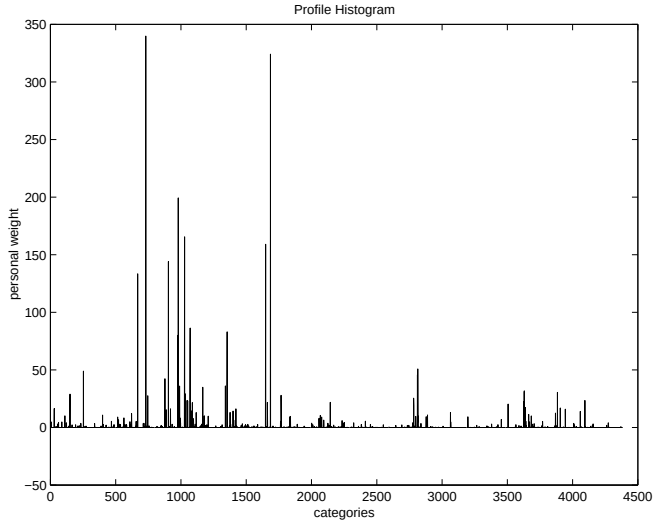


Figure 3.5: Sample user profile: less than 150 categories.

For representational purposes, the sixteen profiles have been clustered into three large groups. The clustering criterion was the number of categories that account for 95% of the total accumulated interest value. Figures 3.6, 3.7, and 3.8 show these three groups.

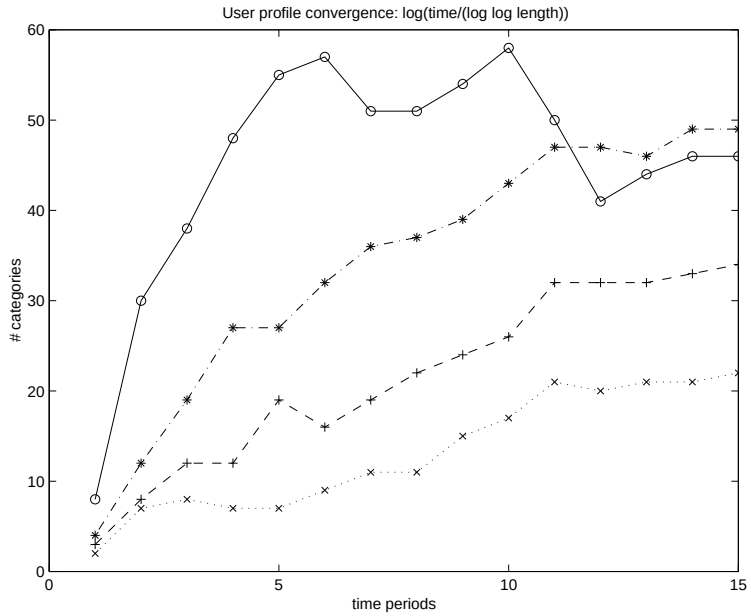


Figure 3.6: Convergence of four profiles with less than 50 categories

The time intervals in figures 3.6-3.8 are actually not clock time but rather represent periods of activity in which an equal number of documents (on average, about 20) have been surfed. In this way, idle times like weekends or vacations do not confuse the overall image, and the evaluation is is consistent between users who surf at different times.

Again, the interest adjustment function was in all cases chosen to be $\Delta\iota(c_i) = \log \frac{time}{\log(\log length)} \cdot \gamma(d, c_i)$. For $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$, the graphs are very similar, but not for the other two adjustment functions. The convergence graphs for all of the adjustment functions are shown in Appendix A.

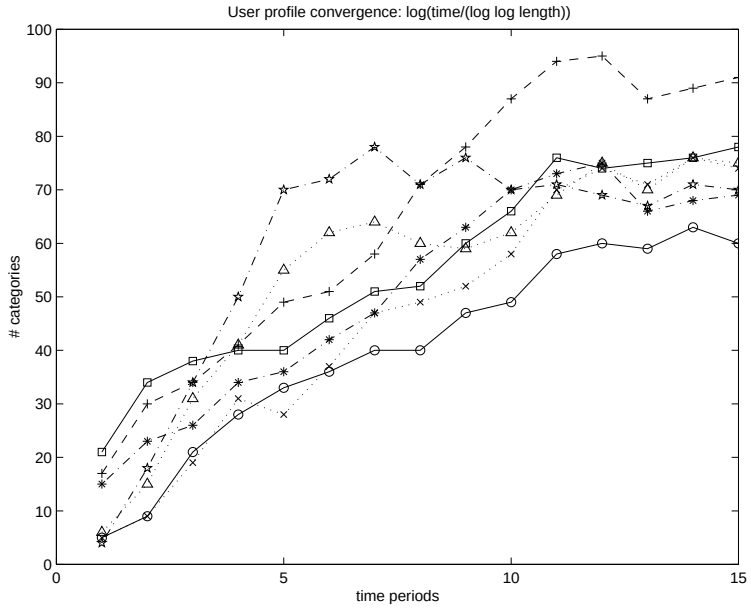


Figure 3.7: Convergence of seven profiles with less than 100 categories

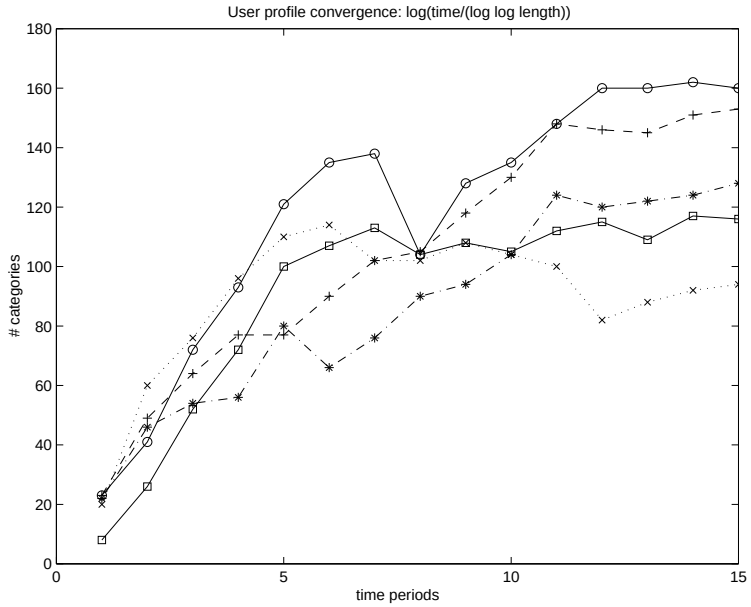


Figure 3.8: Convergence of five profiles with less than 150 categories

With this interest adjustment function, all profiles show a tendency to converge after roughly two thirds of all documents have been surfed: The curves eventually become "flatter" after ten units on the x-axis. On average, that corresponds to roughly 320 pages, or 17 days of surfing. At the beginning, this seems to be a rather high number, but by taking into account the number of categories that have been identified, this accounts for $\frac{320}{50} = 6$ pages per category in the case of profiles with up to 50 categories, for $\frac{320}{100} = 3$ pages in the case of profiles with up to 100 categories, and for $\frac{320}{150} = 2$ pages in the case of profiles with up to 150 nodes. Since every document is assigned more than one category, these numbers do not exactly reflect reality. Nonetheless, the number of pages necessary to determine areas of interests is rather small.

The reason for presenting the profiles with the adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log(\log length)} \cdot \gamma(d, c_i)$ is that two of the four functions do not make the profiles converge. Figures A.1–A.6 suggest that the adjustment functions $\frac{time}{length}$ and $\log \frac{time}{length}$ result in an ever increasing number of categories. *In terms of convergence, these functions should therefore not be used for calculating the profiles.* Figures 3.9 and 3.10 show typical curves.

The adjustment functions $\log \frac{time}{\log \log length}$ and $\log \frac{time}{\log length}$ eventually converge, whereas the other two functions do not. This is due to the characteristics of the logarithmic function. Table 3.1 summarizes the convergence properties (numbers have been determined graphically). In terms of profile convergence, both functions seem to be equally suited.

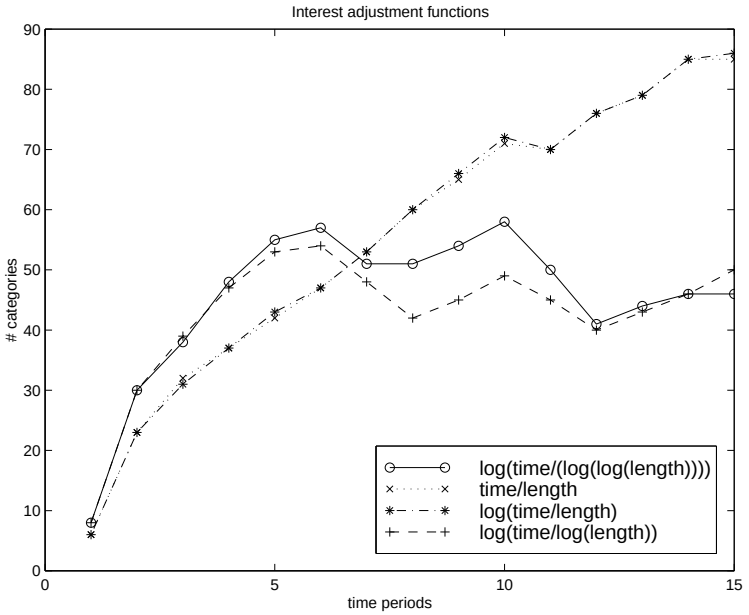


Figure 3.9: Quality functions for a sample profile (1)

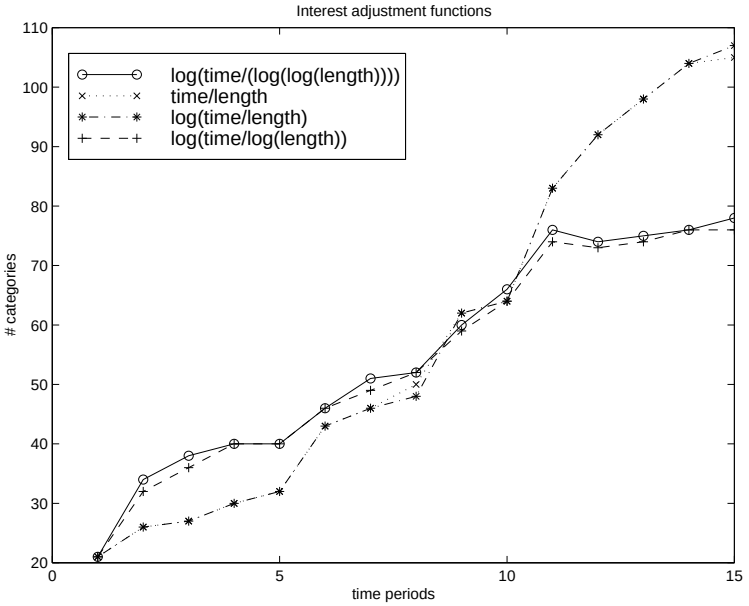


Figure 3.10: Quality functions for a sample profile (2)

| function | convergence after units (average) |
|---|---|
| $\frac{time}{length}$ | no convergence |
| $\log \frac{time}{length}$ | no convergence |
| $\log \frac{time}{\log length}$ | 9.6 |
| $\log \frac{time}{\log \log length}$ | 9.4 |

Table 3.1: Convergence of interest adjustment functions

### 3.2.3 Comparison with actual user interests

Convergence of profiles is a rather technical criterion. Indeed, there is no direct way to evaluate the relation between the user and her profile. The sixteen users have been presented with the top twenty subjects of their profiles (figure 3.11) and been asked how appropriately these inferred categories reflect their interests.

All users have been presented with their profiles and been asked the following questions:

1. *How many* of the above 20 subjects do reflect your actual interests?
2. *How well* does that subset (i.e., the subjects describing your interests) reflect your actual interests (0=very bad ... 5=very good)?
3. *How well* does the entire set of 20 categories describe your actual interests (0=very bad ... 5=very good)?
4. *How many* of the above subjects do not reflect your interests at all?
5. Please answer questions 1-4 by only looking at the top 10 categories, i.e., discard the second half of the list!

In terms of categories, the profiles are almost the same for all interest adjustment functions. This is to say, the inferred *subjects* are the same, but the *weights*

```
1. news-&-reference -> libraries-&-reference -> libraries,-us
     -> university-libraries -> midwest
2. science -> engineering -> academic-programs -> united-states
     -> south
3. entertainment -> music -> styles-a-z -> classical
4. entertainment -> movies -> movie-making -> behind-the-scenes
5. hobbies -> cars-&-trucks -> antiques-&-classics
6. computing -> pcs-&-oss -> win95-&-nt
7. business -> products -> electronics
     -> miscellaneous-componentry
8. computing -> the-biz -> web-presentation-services
     -> web-page-design -> d
9. science -> engineering -> ee-&-computer -> academic-programs
     -> united-states
10. science -> engineering -> academic-programs -> united-states
      -> east
11. shopping -> computer-mart -> business-specialists
      -> western-usa
12. news-&-reference -> geography-&-maps
13. computing -> internet -> for-net-novices
14. news-&-reference -> geography-&-maps -> maps
15. books -> publishers -> computer
16. computing -> access-providers -> international -> canada
      -> alberta
17. computing -> the-biz -> web-presentation-services
      -> web-page-design-&-hosting -> g
18. life-&-style -> religion-&-spirituality
      -> mystics,-paganism,-&-new-age
      -> other-old-&-new-age-religions-&-thoughts
      ->  new-age-resources
19.  computing -> computer-science
      -> courses-&-information-resources -> resources
20.  life-&-style -> religion-&-spirituality -> christianity
      -> denominations -> catholicism
```

Figure 3.11: Sample profile: categories of interest. The last entry in each line represents the area of interest, all other entries are its parent nodes in the concept hierarchy.

| | how many good ones? | how well? (subset) | how well? (all) | how many bad ones? | ratio bad/good |
|---|---|---|---|---|---|
| $\mu_{20}$ | 10.5 (53%) | 3.7 | 2.8 | 5.3 (27%) | 1:2 |
| $\sigma_{20}$ | 4.8 | 1.0 | 1.0 | 5.3 | - |
| $\tilde{x}_{20}$ | 9 | 4 | 3 | 3 | - |
| $\mu_{10}$ | 5.2 (52%) | 3.5 | 2.5 | 3 (33%) | 1:1.7 |
| $\sigma_{10}$ | 2.3 | 1.0 | 1.4 | 2.4 | - |
| $\tilde{x}_{10}$ | 5 | 4 | 3 | 2 | - |

Table 3.2: Profiles vs. actual interests for 20 (subscript 20) and 10 categories (subscript 10). n=16.

of these subjects do differ.

Table 3.2 shows mean $\mu$, standard deviation $\sigma$, and median $\tilde{x}$ for the answers to the above questions with the top 10 and top 20 categories, respectively ($n = 16$). Since these questions do not take into account the category weights, it is sufficient to present the tables for just one interest adjustment function.

In both cases, approximately one half of the categories represent actual interests. However, with about a quarter of the maximum values, the standard deviation is quite high. The reason for this is most likely the suboptimal accuracy of the categorization algorithm. Bearing in mind that the "good" categories have been chosen out of as many as 4,400 categories, this result is still surprisingly accurate. One half of 20 categories chosen reflect actual interests even though these represent only .5% of all possible categories.

If emphasis is put on these "good" categories, users feel represented well - a value of 3.5 might be verbalized as "pretty good". Since roughly one half of the categories do not represent user interests, it is not surprising that the entire set does neither represent nor misrepresent actual interests.

Finally, only a quarter to a third of all the categories do not represent inte-

rests at all.[2] Together with the percentage of accurate categories, this number indicates that the created profiles actually represent their human counterparts fairly well. The goal of chapter 4 is to evaluate whether this qualitative feedback translates into quantitative improvements for some task (in this case, re-ranking and filtering).

## 3.3 Implementation

The first approach to "watching over a user's shoulder" was to modify the freely available Mozilla Netscape source code.[3] Unfortunately, it was not possible to even compile the C++ source code. This led to the second approach of using the caches instead, an approach that exhibits two minor drawbacks:

- Cache files only yield an approximation of the user's surfing behavior since revisiting a cached file does not affect the cache.[4]
- There is no way of monitoring other user interactions, i.e., scrolling, moving the mouse, highlighting, printing, etc. Anatagonomy [46] and Krakatoa [20] are systems that use this kind of interaction for inferring their profiles.

Choosing the caches as a source for user interests results in a different problem: The cache format is not documented, and it seems to change from release to release. Moreover, the decision of the Netscape designers to use hardware dependent encoding formats (little vs. big Endian) yields problems when the browser

[2] There is a difference between "not representing at all" and "not representing well".

[3] http://developer.netscape.com/source/index.html

[4] Since (a) most pages have an immediate expiration date and are hence reloaded every time the browser is restarted, (b) many users actually do restart their browsers more than once a day, and (c) users are monitored over a long period of time, this does not impose a major problem.

is deployed on heterogeneous platforms. As a result, the caches are destroyed whenever there is a change from a little-endian machine to a big-endian one.

Appendix B contains a detailed description of the cache format.

After reading the data, non-HTML pages are thrown away, and an array of cache entries is sorted with respect to the date of last access. Then the time spent on that page is calculated as the difference of two subsequent last access times.

The profiles are updated with each invocation of this cache reader program. Every new entry is then characterized, and the categories' weights for this document are updated with respect to the time spent and the length of the page.

## 3.4 Discussion

This section summarizes the previous ones and concludes with some remarks on privacy issues as well as future extensions.

### 3.4.1 Summary

This chapter described the profile generation mechanism and discussed the generated profiles. In terms of the "profile dimension" of the classification criteria that have been introduced in chapter 2, this system can be described as follows:

- Data source:

  The profiles are obtained by characterizing the documents a user surfed. No explicit user feedback is necessary.[5] The author's opinion is that users are

[5] As of now, the program regularly has to be started manually. Future versions will include mechanisms that make the program run in the background.

unlikely to use such a system on a regular basis.

- Learning algorithm:

  Once the surfed paged have been extracted and stored, they are characterized: their content is determined. Together with the time a user spent on a given page and the length of the latter, an adjustment is made to the weights of the categories that best describe the content of the page. The profile detection runs on the client side (the user's machine, see the next section) and adjusts to the user interests over time. A long term deployment of the profile generator is proposed since this allows to filter temporary interests (such as buying a car, or looking for a school for Ph.D. studies). The weights assigned to the subject nodes do never decrease, i.e., only the "positive" parts of a user's idiosyncrasies are taken into account. Furthermore, since a distinction between long term and short term interests is difficult to quantify, this distinction is not made.

- Profile representation:

  The profiles are stored as a weighted subject hierarchy.

### Profile creation

The weights associated with the nodes of the subject hierarchy are calculated by determining the content of a page. This process yields a list of categories together with their discrimination values. Four different functions are then used to combine them with the time a user spent on a page, and its length. The resulting value is the weight adjustment for the categories in question.

### Profile convergence

Sixteen actual user profiles have been evaluated using a notion of convergence. Roughly speaking, a profile converges if, after some time, the number of categories of interest remains constant. Two out of four functions have been shown to make the profiles converge. The sixteen users have been monitored over a period of 26 days. On average, convergence started after two thirds of the total number of documents have been surfed. Thus, about 48 surfed pages were necessary to achieve convergence. The profiles contained up to 150 categories of interest. This rather large number is probably due to the number of very small, related categories in the hierarchy.

### Accordance with actual interests

One half of the top ten/top twenty categories have been reported to represent actual user interests pretty well. With a total number of 4,400 categories, this is a surprising performance. Less than one third of the top categories have been reported not to represent a user's interests at all. Again, this is probably due to the accuracy of the classification algorithm.

### Alternative approaches

To the author's knowledge, SmartPush [25] is currently the only system to store profiles as concept hierarchies. These are much smaller (40-600 nodes), and weight adjustments are done with respect to data that *explicitly* describes the document contents (cf. page 25). It is doubtful that hand-made hierarchical content annotation[6] of data will be done on a large scale.

[6] i.e., not just lists of keywords as in the case of XML

Systems that use structured information rather than simple lists of keywords include PEA [36] and SiteSeer [45] (bookmark structure), PSUN [52] (K-lines), and SiteIF [53] (semantic networks).

Browsing behavior is used for data acquisition in Anatagonomy [46], GroupLens[7] [24], Letizia [29, 28], Krakatoa [20], Personal WebWatcher [34], and WBI [5].

Chapter 2 and table 2.1 in particular contain a detailed description of these as well as many other systems. Many systems do mix explicit with implicit feedback which makes the list of purely implicit systems rather short.

Very few systems are thoroughly evaluated, and if so, the metrics used is incompatible to the one used in this chapter. A comparison of the "performance" of personalization systems is hence impossible.

### 3.4.2 Privacy

Cache data can become very sensitive information. It is not easy to obtain cache files for experimental purposes as users are aware of the wealth of information they may disclose.

In a system that involves personalization, it appears to be the best idea to create and store profiles locally [31]. Available systems store them either with the user, or at a central server. The advantage of locally stored profiles is that they are less likely to be stolen (for criminal hackers, it is presumably more interesting to crack a site with thousands of profiles than to steal one single profile).

The decision about where to store the profiles depends, of course, on the application. For re-ranking or filtering search results, the user profile forms a single isolated part of the overall system. In particular, it can be used *after* the outside world has been contacted.

[7] focusing on Usenet news

This is not the case for systems that attempt to match profiles (e.g., some collaborative systems). Matching profiles makes sense for recommendation services: The books a user with particular interests bought could be recommended to all users with similar interests (provided they want the system to do so). Or, if an expert is sought in a particular field, there must be a database which can be matched against this particular field.

This problem could be addressed by creating anonymous users. For instance, the LPWA system [13] creates multiple users which seemingly are not related one to the other, and these virtual users can be used for e-mail contacts etc. Furthermore, it proxies all Internet accesses and therefore hides the IP address of the original user. [7] contains a collection of related papers.

It should be pointed out that even if there is a risk of a "big brother monitors every user" scenario, the advantages of personalization services can have a tremendous impact on the quality of information retrieval on the Internet.

### 3.4.3 Future Extensions

Three areas of future work are obvious:

1. *Incorporating explicit feedback into the system*

   Since the accuracy of the categorization algorithm needs to be improved, explicit feedback could help to obtain better profiles. Two levels of feedback are conceivable:

   - Users could directly modify the weights in the concept hierarchy. For instance, in the evaluation process described above, users could simply click on the categories they do not consider to represent their interests to decrease these subject's weights. Furthermore, browsing tools could

be used to determine areas of interests that have not been detected. Later on, these hand made modifications have to be assigned a higher importance than automatically inferred data.

- On the application level, users could give feedback to tell the system whether or not it is performing well. For instance, search results could be assigned relevant/irrelevant values. This information would then be used in adjusting the ontology's weights.

The explicit feedback should be kept to a minimum since it is likely to annoy the user.

2. *Personalizing the structure of the ontology and the underlying thesaurus*

   The structure of the ontology could be adjusted by tracking not only the weights of nodes but also the documents that have been characterized into the nodes (splitting or coalescing nodes). A node with a large weight and a large number of documents associated with it might actually be too large. This means that the subject it represents is too broad for the user. By characterizing and clustering all documents associated with that node, new children nodes may be created. Conversely, nodes at lower levels of the hierarchy may refer to subject areas that are too narrowly defined for the user. Currently, the system stores copies of the surfed documents associated with each node and creates indexes for nodes if a certain number of documents for that node was exceeded. What is still needed, are clustering algorithms for splitting and coalescing purposes.

   Finally, the training set of documents for the characterization algorithm can be extended or replaced to retrain the characterizer. This is likely to result in a better (personalized) characterization, in particular, if the learning

process is supervised by the user.

3. *Integrating the profile generator into a web browser*

   This would allow the system to follow the user's activities in more detail (scrolling behavior, mouse movements, highlighting). Since cached files are accessed only once per session (provided they have an immediate expiration date, which is the case for most documents), information on accessing a document multiple times during one session is not reflected in the caches.

It is noteworthy that the deployment area of such profiles is ubiquitous (see Chapters 1 and 2). Since the profile creation process is not coupled with the application, the profiles created by the above system can be used for a broad range of applications.

# Chapter 4

# Personalized Search

The wealth of information available on the web is actually too large: when entering a query into a search engine such as AltaVista, too many results are retrieved. The number of results regularly exceeds 1,500, and the top ranked documents a user can have a look at, usually are not relevant to this user. Besides the enormous amount of documents to be retrieved, this happens due to an inherent problem in the keyword based search: search terms are ambiguous; their meaning depends on the context and, more importantly, on the meaning a user assigns to them.

In the evaluation of the proposed system, 48 query results have been judged by 16 users, the judgement being either "relevant" or "irrelevant". On average, only $\mu = 8.7$ out of 20 result pages were considered to be relevant (median $\tilde{x} = 8.5$, standard deviation $\sigma = 3.0$). This is consistent with the findings in [8] which reports that roughly 50% of the retrieved documents are irrelevant (with a statistically more significant set of 1,425 queries and 27,598 judged results).

There are three common approaches to address this problem:

- Re-Ranking

Re-Ranking algorithms apply a function to the ranking numbers that have been returned by the search engine. If that function is well chosen, it will bring more relevant documents to the top of the list.

- Filtering

  Filtering systems determine which documents in the results sets are relevant and which are not. This is usually done by comparing the documents to a list of keywords that describe a user or a set of documents that the user previously judged relevant or irrelevant, respectively. Good filters filter many non-relevant documents and do keep the relevant ones in the results set.

- Query Expansion

  Often, queries are very broad. Consider the query "Harley Davidson". With a database as large as the Web, there will be thousands of documents that are related to Harley Davidsons. If a query can be expanded with the user's interests, the search results are likely to be more narrowly focused. However, this is a very difficult task since query reformulating needs to expand the query with relevant terms. If the expansion terms are not chosen appropriately, even more irrelevant documents will be returned to the user.

This Chapter uses the profiles of chapter 3 to implement the first two ideas. Section 4.1.1 explains the re-ranking and filtering algorithms that will be evaluated in section 4.2.

## 4.1 Re-Ranking and Filtering

### 4.1.1 Re-Ranking

Re-ranking is done by modifying the ranking that was returned by the underlying search engine, ProFusion in this case. The idea is to characterize each of the returned documents (or rather their title together with their summary[1]) and, by referring to the user profiles, to determine how much a user is interested in these categories. The user's average interest in the document's top categories is assumed to be an approximation to the actual user interest in the whole document.

Remember that $\gamma(d, c_i)$ denotes a measurement of how well category $c_i$ describes the content of a document $d$. Let $\pi(c_i)$ be the personal interest assigned to category $c_i$. The interest values for the estimated top 4 categories of interest, $c_1, \ldots, c_4$, in a given document $d$, $\iota_1(d), \ldots, \iota_4(d)$, are defined as

$$\iota_i(d) = \pi(c_i) \cdot \gamma(d, c_i).$$

Since the values of $\gamma$ are usually rather small ($\leq .5$) and the values of $\pi$ rather high (up to 350.0), the values of $\pi$ are normalized with respect to the maximum personal interest of all categories, $\iota_{max}$. They are not normalized with respect to the highest interest of the top function since this would mean that a user is interested in all his categories to a same extent. The normalized interests, $\tilde{\iota}_i$, are then calculated according to

$$\tilde{\iota}_i(d) = \frac{\iota_i(d)}{\iota_{max}}.$$

[1] [8] and [43] indicate that this is sufficient for classification purposes.

The four values of $\tilde{\iota}$ are hence the interests in the top 4 categories that have been determined to describe the content of a document best. Summing them up and dividing by four (hopefully) yields an approximation to the actual user interest in that document.

These values are then combined with the ranking that is returned by the search engine (ProFusion). Let $w(d_j)$ be the rank that ProFusion assigned to document $d_j$. Four ranking functions will be evaluated:

- $\varrho_1(d_j) = w(d_j) \cdot \left(.5 + \frac{1}{4}\sum_{i=1}^{4} \tilde{\iota}_i(d_j)\right)$
  multiplies both values which are hence not weighted,
- $\varrho_2(d_j) = w(d_j) + \frac{1}{4}\sum_{i=1}^{4} \tilde{\iota}_i(d_j)$
  is the unweighted sum of both values,
- $\varrho_3(d_j) = 3 \cdot w(d_j) + \frac{1}{4}\sum_{i=1}^{4} \tilde{\iota}_i(d_j)$
  assigns a slightly higher weight to ProFusion's ranking,
- $\varrho_4(d_j) = w(d_j) + \frac{3}{4} + \sum_{i=1}^{4} \tilde{\iota}_i(d_j)$
  assigns a slightly higher weight to the personal interest, and
- $\varrho_5(d_j) = w(d_j) + \frac{3}{2}\sum_{i=1}^{4} \tilde{\iota}_i(d_j)$
  assigns an even higher weight to the personal interest.

Adding 0.5 to the second factor of $\varrho_1$ seems reasonable since, in general, approximately three out of four values for $\iota_i$ have values in the proximity of 0.0. Such an adjustment is not needed for the weighted sums (since this would only "shift" the whole function graph).

### 4.1.2 Filtering

Filtering is done by using the rankings $\varrho_1 - \varrho_5$ that have been described in the previous chapter. The idea is straightforward: All weights of the personalized rankings are normalized to 1.0. A threshold is introduced which divides the relevant documents from the non-relevant ones. Note that this machine judgement need not neccesarily be the same as the user judgement. By ranging over some reasonable thresholds, the best value will be determined. The judgement that relates to this threshold is then to be compared with the user judgement.

## 4.2 Evaluation

The results that have been produced by the different re-ranking systems must be evaluated. Since these results are in the form of rank-ordered URLs, it is necessary to select an objective measure for the relative quality of two rank-ordered lists.

The eleven point precision average (section 4.2.1) is one such measure. The basic idea is to cluster documents into two groups, the relevant and the non-relevant ones, and to check how many relevant documents appear at the top of the re-ranked list. This measure has one disadvantage in that it considers all relevant documents to be equally relevant. The n-dpm (section 4.2.2) measure overcomes this restriction.

The filtering algorithm is evaluated by determining how many relevant and how many irrelevant documents have been filtered.

All the approaches are used to evaluate the system with actual users and queries.

### 4.2.1 Re-Ranking: Eleven point precision average

This section presents a brief review of the 11 point precision average which will then be used for evaluation of the personalized ranking mechanism.

#### Introduction

The *11 point precision average* [18] is an approach used to compare information retrieval systems. Initially, it was used to evaluate systems presented at the Text Retrieval Conferences.

The basic idea is to partition the set of documents (in this case, the search results) in two classes, the relevant and the non-relevant ones. The average measure then yields a set of values that reflect how many *relevant* documents have been highly rank ordered. This is, the more relevant documents appear on top of the result list, the better the ranking (and hence the system).

The documents are manually partitioned into two classes - relevant and irrelevant. Within each class, there is no ordering between two documents: they are equally important. Two evaluation measures are used: *recall* and *precision*:[2]

$$\text{recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}}, \text{ and}$$

$$\text{precision} = \frac{\text{number of relevant items retrieved}}{\text{total number of items retrieved}}.$$

Let $\rhd= \{d_1, d_2, \ldots, d_n\}$ be a rank ordered set (or rather sequence) of documents, where document $d_1$ is more important than document $d_2$, document $d_2$ is

[2]The third popular measure, *fallout*, is not needed with this measure.

more important than document $d_3$, etc.

For each relevant item in $\triangleright$, precision and recall are calculated, where

- the number of relevant items is counted with respect to all elements in $\triangleright$ that appear *before* this relevant item in question (the latter is included in this list as well),
- the denominator of the expression for recall refers to the entire set of relevant documents retrieved, and
- the denominator of the expression for precision refers to all documents in $\triangleright$ that appear *before* the relevant item in question (the latter being included).

For instance, with three relevant documents retrieved (in the case of comparing search results, the set of relevant items retrieved and the set of relevant items are identical) at ranks 4, 7, and 9, the three recall points are 1/3, 2/3, and 3/3. The precisions are 1/4 at recall point 1/3, 2/7 at recall 2/3, and 3/9 at recall 3/3, respectively.

A uniform recall scale is needed to evaluate the performance of a system w.r.t. to different queries (with different numbers of relevant items in the result list), as well as for the comparison with other systems. The interval [0;1] is divided into 10 intervals of equal length (which yields 11 interval boundary points), and the precisions are calculated at each of these recall points. Since the actual recall points will usually not coincide with these interval boundaries, the values are interpolated: the interpolated precision at a recall cutoff $R$ is defined to be the maximum precision at all points $\geq R$.

In the above example, precision value 1/4 is assigned to the interval boundaries 0.0, 0.1, 0.2, and 0.3 (since the maximum precision at recalls greater than these

values is 1/4), precision value 2/7 is assigned to the boundary points 0.4, 0.5, and 0.6 (since the maximum precision at recalls greater than these values is 2/7), and precision value 3/9 is assigned to the boundary points 0.7, 0.8, 0.9, and 1.0 (since 3/9 is the maximum precision of all recall points $\geq 0.7$.

Note that in practice, precision values for recall cutoff values of 0.0 do not exist. They are introduced by the above interpolation rule.

These precision vectors of length 11 are averaged component-wise (there is one such vector per query). These average vectors can then be used to compare retrieval systems by graphically superposing two of these (interpolated) recall-precision vectors. The system's vector that is on top of the other one is the "better" one.

Usually, these eleven values are then averaged to yield one performance indicator for a system. Since there are few documents being considered per query, the following discussion will also take into account all recall cutoff values.

**Using the 11 point average to evaluate the presented system**

Sixteen users were asked to judge the results for three queries which they chose freely. The results were presented in random order. The judgement was either "relevant" or "non-relevant". The re-ranking algorithm is evaluated by

1. Training:

   Two thirds of the queries (two queries per user) are chosen to determine which combination of interest adjustment function/re-ranking formula is the best one.

2. Testing:

The remaining query results are then re-ranked by the algorithm that is considered to be the best one.

Each of the four interest adjustment function (p. 48) was coupled with the five re-ranking functions $\varrho_1 - \varrho_5$ (p. 71).

Figures 4.1-4.4 show the recall-precision graphs for 16 users and $n = 16 \cdot 2 = 32$ judged queries. The 11 point averages have been averaged to yield one single 11 point average curve for all 32 queries. In the legend, *ProFusion* refers to the ranking that is provided by ProFusion, *multiplication* refers to $\varrho_1$, *addition* refers to $\varrho_2$, *more ProFusion* refers to $\varrho_3$, *more personal* refers to $\varrho_4$, and *even more personal* refers to $\varrho_5$.

If a curve is "above" another curve, the corresponding system is "better" in terms of recall and precision since with more retrieved documents, more relevant documents have been retrieved. Figure 4.1 and 4.2 are examples where the personalized ranking curves lie above the system ranking curve.

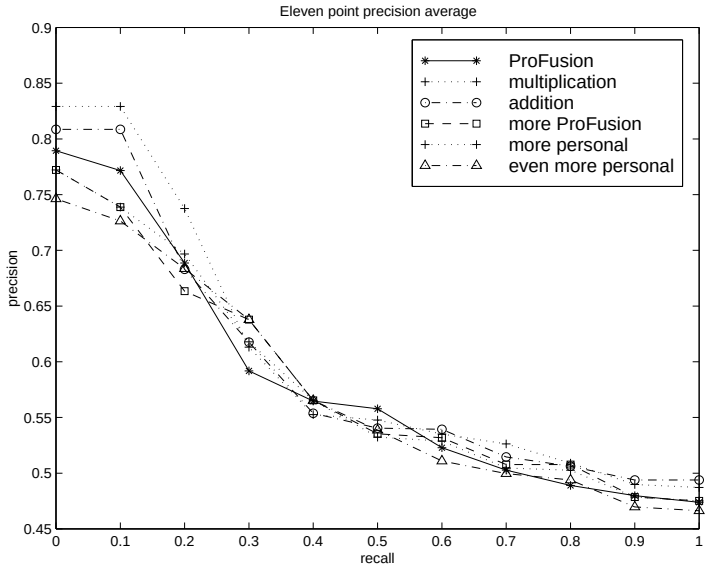


Figure 4.1: 11 point average precision for interest adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$ and 5 different ranking formulae.

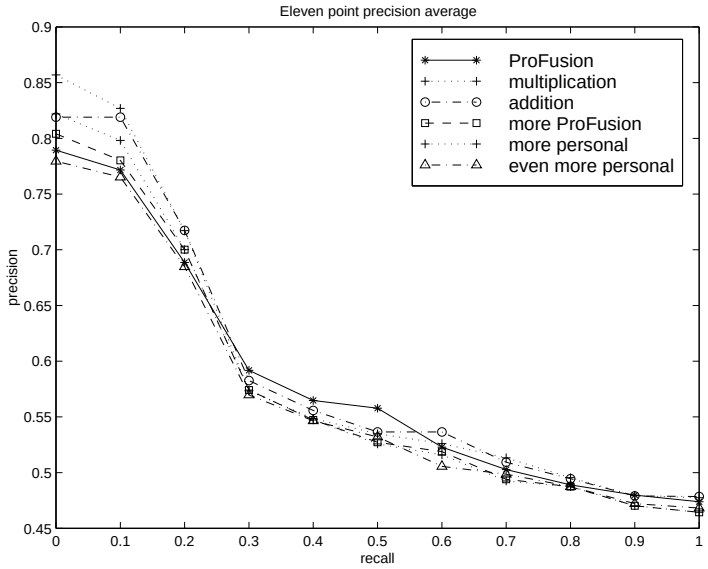


Figure 4.2: 11 point average precision for interest adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log(\log length)} \cdot \gamma(d, c_i)$ and 5 different ranking formulae.

The corresponding interest adjustment function for figure 4.1 is $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$, one of the two functions for which profile convergence has been detected.

$\varrho_1$ and $\varrho_5$ exhibit higher precisions for recall cutoffs less than .4 and greater than .5. In the first interval, the improvement gets as high as 6.5% for $\varrho_1$ (up to 5.0% for $\varrho_2$). In the second interval, the rankings are almost identical. ProFusion's ranking is slightly better for the recall cutoff values between these two intervals (but not better than 1.8%).

The personalized rankings do not, at any recall point, considerably fall below the ProFusion ranking. Figure 4.5 shows the increases and decreases for all recall cutoffs. Again, $\varrho_1$ and $\varrho_2$ exhibit the best performance since they improve the precision for small recall cutoff values; the overall performance increase is roughly 8%. For this graph, it can be seen that the increase for high recall cutoffs is larger than for low ones. This undesired result is probably due to the small set of sample

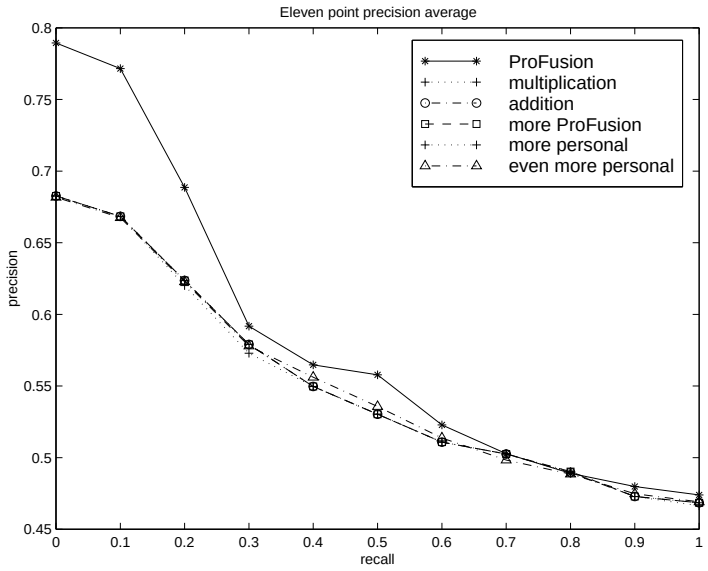


Figure 4.3: 11 point average precision for interest adjustment function $\Delta\iota(c_i) = \log \frac{time}{length} \cdot \gamma(d, c_i)$.

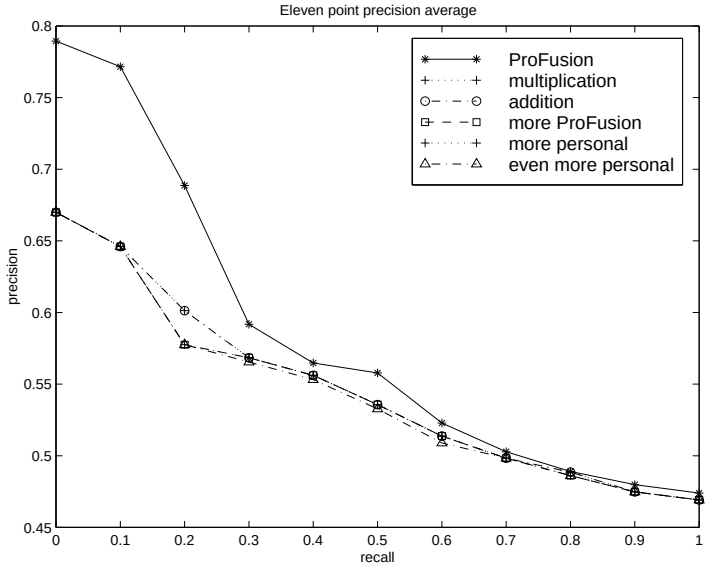


Figure 4.4: 11 point average precision for interest adjustment function $\Delta\iota(c_i) = \frac{time}{length} \cdot \gamma(d, c_i)$ and 5 different ranking formulae.

queries, and the behavior is not exhibited for the other function (Figure 4.6).

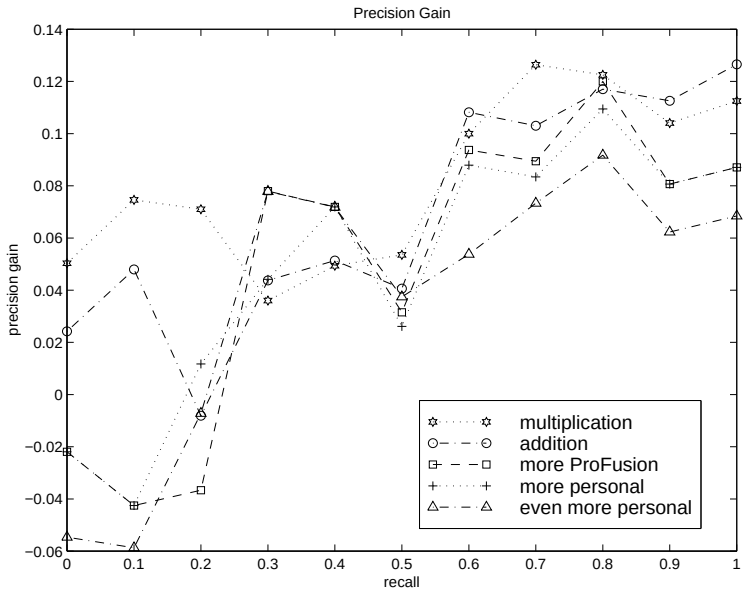


Figure 4.5: Relative precision increase for $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$

It is likely that users want the relevant items right at the beginning of the results list. This leads to the assessment that $\varrho_1$ and $\varrho_2$ are probably the best choices: For them, the precision values at low recall values are the highest, and in particular, they are better than the corresponding ProFusion values.

The corresponding interest adjustment function for figure 4.2 is $\Delta\iota(c_i) = \log \frac{time}{\log(\log length)} \cdot \gamma(d, c_i)$, the second of the two functions for which profile convergence has been detected.

Except for $\varrho_5$, all ranking formulae exhibit higher precision values for recall cutoffs less than .3. This means that the personalized ranking includes more relevant documents in the top five results than does the ProFusion ranking. The gains in precision for the best ranking function, $\varrho_1$, vary from 1.4% at recall cutoff

.2 to 8.9% at recall cutoff .0 (and since that value does not really exist, the highest gain in precision is 7.8% at recall cutoff .1).

For recall cutoffs between .3 and .6, ProFusion's ranking is slightly better. At cutoff value .5, it outperforms the best candidate, $\varrho_1$, by 4.8% (see below). Precisions are almost the same for all functions at cutoff values greater than .6.

The personalized rankings do not, at any recall point, considerably fall below the ProFusion ranking. Figure 4.6 shows the gains and losses for all recall cutoffs.

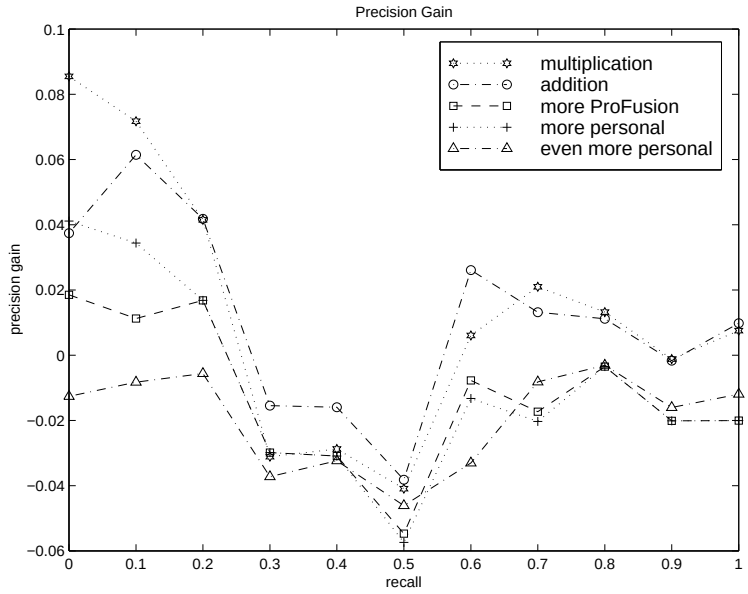


Figure 4.6: Relative precision increase for $\Delta\iota(c_i) = \log \frac{time}{\log\log length} \cdot \gamma(d, c_i)$

As pointed out above, it is likely that users want the relevant items on top of the result list. Hence, $\varrho_1$ and $\varrho_2$ are probably the best choices: Similar to Figure 4.1, the precision values at low recall values are the highest, and in particular, they are better than the corresponding ProFusion values.

Figures 4.3 and 4.4 provide strong evidence that the interest adjustment functions $\Delta\iota(c_i) = \log \frac{time}{length} \cdot \gamma(d, c_i)$ and $\Delta\iota(c_i) = \frac{time}{length} \cdot \gamma(d, c_i)$ are poor choices not only in terms of profile convergence, but also in terms of re-ranking. Both are clearly outperformed by the unpersonalized ProFusion ranking.

It can therefore be concluded that both $\Delta\iota(c_i) = \log \frac{time}{\log \log length} \cdot \gamma(d, c_i)$ and $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$ in combination with $\varrho_1$ or $\varrho_2$ are the best choices for the interest adjustment/ranking function combination.

For both functions, relevant results are more likely to appear on top of the result list with $\varrho_1$, but in the middle part of that list, $\varrho_2$ yields better results.

An interesting aspect of this is that the length of the page does not seem to matter very much (its contribution in both formulae is very small, a subtraction of a double or even triple logarithm). [24] do not use at all the length of items that have to be ranked, but focus on the time spent. The goal of incorporating the length into the above formulae was to normalize time by length. This seems unneccessary because users can tell at a glance that a page is irrelevant and, in general, reject it quickly, regardless of its length. There seems to be no need for normalization because if they spend longer on a page it is because it is relevant, and not because it is longer.

To verify the above recommendation for a ranking formula, the remaining 16 queries and their results are re-ranked and compared with the user ranking. Figure 4.7 shows a result that is very similar to the ones obtained in the above "training" part. The personalized systems yields a slightly higher precision at low recall cutoff values (8.6% increase in precision for recall values less than .2). The function that was chosen in this case is $\Delta\iota(c_i) = \log \frac{time}{\log \log length} \cdot \gamma(d, c_i)$; the graph

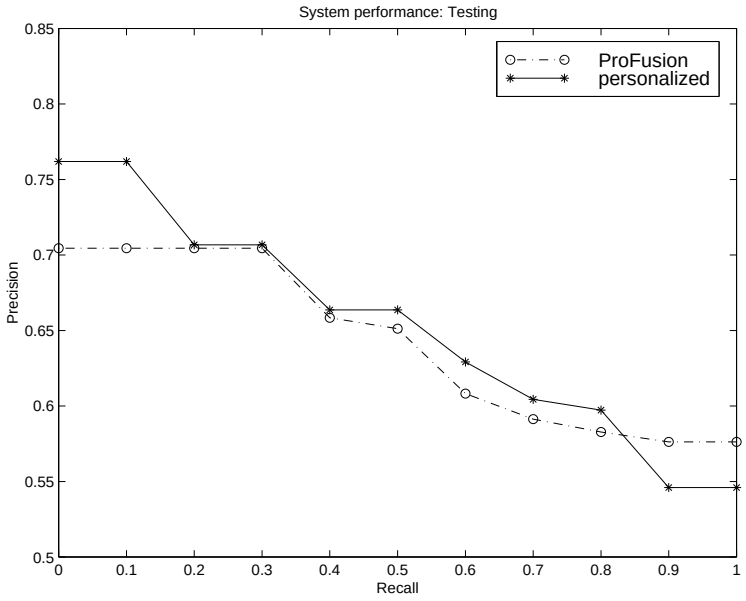


Figure 4.7: 11 point average for the testing set: 16 queries

for the other function is almost identical.

The result of this evaluation is that personalized re-ranking is possible. With the described functions, a performance increase of up to 8% can be expected. This means that at every recall cutoff, 8% more relevant documents are retrieved.

### 4.2.2 Re-Ranking: n-dpm

This section consists of a review of the normalized distance-based performance measure (*n-dpm*), of implementation issues, and of the results obtained by evaluating the presented system in terms of the *n-dpm*. The experiments will be exactly the same as the ones in the previous chapter, but an additional evaluation method will be used.

### Introduction

The more recent *n-dpm* measure (normalized distance-based performance measure, [58]) is based on two fundamental ideas:

1. The distinction between "relevant" and "non-relevant" documents is too coarse.

2. Studies[3] show that users are not able to reproduce a *cardinal* ranking of a list of documents. This is to say, they cannot assign the same absolute values (representing the relevance) to a list of documents when asked to do so twice. However, the relative positions of the documents in the two rankings exhibit strong similarities, i.e., the two rankings are the same on an *ordinal* scale.

In order to evaluate the performance of a ranking system, it is necessary to compare the obviously perfect ranking - the ranking the user would have chosen - with the ranking as provided by a search engine (or a different retrieval system).

The idea behind *n-dpm* is to compare two rankings by comparing every document in ranking 1 with every document in ranking 2. For each pair of documents, the following test is performed:

- If both rankings agree on which document is more important, the distance between these two documents is zero.

- If one ranking does not exhibit any ordering between the documents (and they are hence indifferent), but the other does, there is a slight distance

[3] see [58] for references

between these two documents.

- If the preference relationship in one ranking is the inverted relationship in the other ranking, then there is a large distance between these two documents.

These distance values are then basically added for each pair, and the resulting sum is the overall distance between the two rankings.

Formally, a finite set of documents $\mathcal{D}$ is required to be ordered w.r.t. to a weak[4] ordering $\rhd \subseteq \mathcal{D} \times \mathcal{D}$. $d_1 \in \mathcal{D}$ is said to be *preferred* to $d_2 \in \mathcal{D}$ iff $d_1 \rhd d_2$. If neither $d_1 \rhd d_2$ nor $d_2 \rhd d_1$ hold, $d_1$ is said to be *indifferent* to $d_2$ (and vice versa). An indifference relation $\sim \subseteq \mathcal{D} \times \mathcal{D}$ can be defined as follows: $\sim = \left\{(d_1, d_2) \middle| d_1 \not\rhd d_2 \wedge d_2 \not\rhd d_1\right\}$, but it might also be given explicitly.

$d_1 \rhd d_2$ might be interpreted as "the user prefers document 1 to document 2", and $d_1 \sim d_2$ can mean both $d_1$ and $d_2$ are equally important or incomparable.

In the example of a binary ranking where every document is either "relevant" or "non-relevant", the documents within each partition are indifferent[5] one to the other, and the relevant class is possibly "larger" w.r.t. to this ranking than the "non-relevant" class.[6]

Two weak orderings $\rhd_1$ and $\rhd_2$ are then compared by means of two auxiliary functions, $cmp_{\rhd_1,\rhd_2} : \mathcal{D}^2 \to \{agreement, compatibility, contradiction\}$ and $\delta_{\rhd_1,\rhd_2}$ :

[4] A *weak* ordering is antisymmetric and negatively transitive, i.e., $d \rhd d' \to (d' \not\rhd d)$, and $\big((d \not\rhd d') \wedge (d' \not\rhd d'')\big) \to (d \not\rhd d'')$. Note that the original notation $\neg(d \rhd d')$ is somewhat misleading since its usual interpretation is "the statement $\neg(d \rhd d')$ holds". This is not equivalent to "the statement $d \rhd d'$ does not hold" $(d \not\rhd d')$ which is chosen here and was probably intended in the original paper.

[5] Actually, this would be the *definition* of the indifference relation.

[6] In other words, if the definition of $\sim$ makes it a congruence relation, then it factorizes $\mathcal{D}$ into two equivalence classes, the "relevant" class and the "non-relevant" class.

$\mathcal{D}^2 \to \{0, 1, 2\}$.

If both orderings are the same for the two documents, *cmp* is evaluated to *agreement*. If, in one ordering, they are indifferent one to the other, and in the other ordering, one document is actually preferred to the other, it is evaluated to *compatibility*.

Finally, if in one ordering document 1 is preferred to document 2, and in the other ordering, the inverse relation holds, there is a *contradiction*.

$$cmp_{\rhd_1,\rhd_2}(d,d') = \begin{cases} agreement, & \text{iff } \big(d \rhd_1 d' \wedge d \rhd_2 d'\big) \vee \big(d' \rhd_1 d \wedge d' \rhd_2 d\big) \\ & \quad \vee \big(d' \sim_1 d \wedge d' \sim_2 d\big) \\ compatibility, & \text{iff } \big(d \sim_1 d' \wedge \big(d \rhd_2 d' \vee d' \rhd_2 d\big)\big) \\ & \quad \vee \big(d \sim_2 d' \wedge \big(d \rhd_1 d' \vee d' \rhd_1 d\big)\big) \\ contradiction, & \text{iff } \big(d \rhd_1 d' \wedge d' \rhd_2 d\big) \vee \big(d' \rhd_1 d \wedge d \rhd_2 d'\big) \end{cases}$$

Note that the three cases cover all possible cases, i.e., *cmp* is a total function.

The difference between two documents is then mapped to natural numbers by assigning numeric values to the three outcomes of *cmp*:

$$\delta_{\rhd_1,\rhd_2}(d,d') = \begin{cases} 0, & \text{iff } cmp_{\rhd_1,\rhd_2}(d,d') = agreement \\ 1, & \text{iff } cmp_{\rhd_1,\rhd_2}(d,d') = compatibility \\ 2, & \text{iff } cmp_{\rhd_1,\rhd_2}(d,d') = contradiction \end{cases}$$

That is, the more "obvious" the difference between the two rankings for two documents is, the higher the value of $\delta$ gets.

It is then easy to accumulate the distance $\beta$ between two orderings for a set of documents:[7]

[7] The factor 1/2 is added since with this definition, every comparison is performed twice.

$$\beta(\rhd_1, \rhd_2) = \frac{1}{2} \cdot \sum_{d \neq d' \in \mathcal{D}} \delta_{\rhd_1, \rhd_2}(d, d')$$

$\beta$ is motivated by showing that it reflects certain rationality axioms, an approach that is widely used in decision theory [60]. Moreover, $\beta$ is shown to be the only function reflecting these rationality axioms.

Consider the following two rankings $\rhd_1$ and $\rhd_2$ on the set of documents $\{d_0, d_1, d_2, d_3\}$:

$$\left\{ d_0 \rhd_1 d_1 \rhd_1 \begin{matrix} d_2 \\ d_3 \end{matrix} \right\} \text{ and } \left\{ \mathrm{d}_1 \rhd_2 \begin{matrix} \mathrm{d}_0 \\ \mathrm{d}_2 \end{matrix} \rhd_2 \mathrm{d}_3 \right\}.$$

The values for $\delta$ are

$$\delta_{\rhd_1, \rhd_2}(d_0, d_1) = 2,\ \delta_{\rhd_1, \rhd_2}(d_0, d_2) = 1,\ \delta_{\rhd_1, \rhd_2}(d_0, d_3) = 0,$$
$$\delta_{\rhd_1, \rhd_2}(d_1, d_2) = 0,\ \delta_{\rhd_1, \rhd_2}(d_1, d_3) = 0,\ \delta_{\rhd_1, \rhd_2}(d_2, d_3) = 1.$$

Summing them up yields the distance $\beta(\rhd_1, \rhd_2) = 4$.

Clearly, a system would be considered to have ranked a set of documents perfectly if the distance between the ranking as calculated by the system and the ranking as desired by the user was zero. The smaller the value of $\beta$ for a system ranking and a user ranking is, the better the system performs in terms of representing a user's interests.

The above "perfect" criterion (user ranking and system ranking are the same) can be weakened to the so-called "acceptable ranking criterion". With this cri-

terion, a system is only required to rank preferred documents higher than the non-preferred ones. Such a ranking can be derived by arbitrarily ranking the documents in a same equivalence class. This leads to an evaluation of the system which does not take into account the system's ordering of documents that belong to the same equivalence class in the user's ranking.[8]

An acceptable ranking $\rhd_{acc}$ can be constructed as

$$\rhd_{acc} = \rhd_{user} \cup (\sim_{user} \cap \rhd_{system}),$$

where the subscripts indicate the origin of the ranking. The *distance-based performance measure, dpm,* is then defined as[9]

$$dpm(\rhd_{user}, \rhd_{system}) = \beta(\rhd_{acc}, \rhd_{system}).$$

*dpm* is appropriate for comparing retrieval systems with respect to a fixed query. The problem with this approach is, for example, that a performance improvement, which reduces the distance from 200 down to 100 for one query, is considered to be the same as the improvement that reduces the distance from 2 to 1 for 100 queries.[10] This problem can be remedied by using relative - normalized - distance measures. This is achieved by scaling the actual distance w.r.t. to the worst distance, which is the inverse of the user ranking, $\rhd_{user}^{-1} = \{(d_1, d_2) | d_2 \rhd_{user} d_1\}$:

$$ndpm(\rhd_{user}, \rhd_{system}) = \frac{dpm(\rhd_{user}, \rhd_{system})}{dpm(\rhd_{user}, \rhd_{user}^{-1})} = \frac{dpm(\rhd_{user}, \rhd_{system})}{2 \cdot \mid \rhd_{user} \mid}$$

[8] Clearly, if there are only two equivalence classes - relevant and non-relevant - one major drawback of the 11 point average measure is reintroduced.

[9] In practice, it is not necessary to actually calculate $\rhd_{acc}$.

[10] Assuming that distances for different queries are averaged.

defines the *normalized distance-based performance measure*. By definition of $\rhd_{user}^{-1}$, the rightmost equation only holds if $\rhd_{user}$ does not contain explicit indifference relationships (in this case, every relation between two elements is inverted in the inverted ranking, yielding a "punishing" value of 2 for each comparison. Note that it is possible to express every explicit indifference relation by a set of ordering relations).

Calculating the average *ndpms* for a given set of $N$ queries (and therefore, $N$ user rankings[11] $U = \{\rhd_{user}^1, \rhd_{user}^2, \ldots, \rhd_{user}^N\}$ and $N$ system rankings $S = \{\rhd_{system}^1, \rhd_{system}^2, \ldots, \rhd_{system}^N\}$, the average distance is expressed as

$$\Delta_{ndpm}(U, S) = \frac{1}{N} \sum_{i=1}^{N} ndpm(\rhd_{user}^i, \rhd_{system}^i).$$

A low value of $\Delta_{ndpm}$ indicates a strong similarity between the system and the user ranking (the system is "good" in terms of reflecting the user's interests).

### Implementation

Documents are represented as nodes of a directed graph, and the (directed) edges of this graph represent the preference relation (i.e., $n_1 \rightarrow n_2$ indicates that the document associated with node $n_1$, $d_1$, is preferred to $d_2$, the document that is associated with node $n_2$: $d_1 \rhd d_2$). This graph may consist of several components which are not connected one to the other (they represent the "threads" of the partial ordering).

[11] Set notation is used to denote sequences.

The transitive closure of such a graph is then calculated (in order to reflect the rationality axioms) The node-by-node comparison of the transitive closures of two partial orders is straightforward.

### Using n-dpm to evaluate the presented system

n-dpm was chosen because more than one query is taken into account. The n-dpm values have been averaged for 32 queries (training data). Figure 4.8 shows the distance of the personalized rankings to ProFusion's ranking.

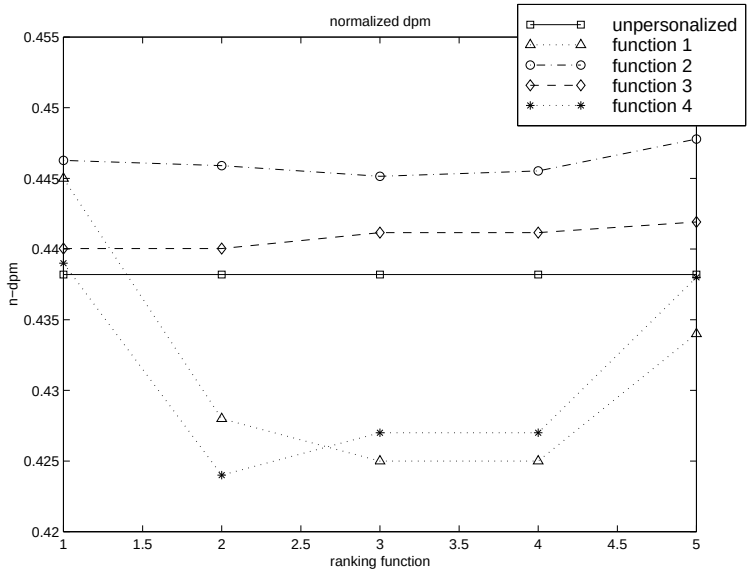


Figure 4.8: n-dpm for the training set: 16 queries

As in the case of the 11 point average measurement, the interest adjustment functions $\Delta\iota(c_i) = \frac{time}{length} \cdot \gamma(d, c_i)$ and $\Delta\iota(c_i) = \log \frac{time}{length} \cdot \gamma(d, c_i)$ (functions 2 and 3 in the graph) worsen the results (the distances are shown on the y-axis; the smaller the distance to the user ranking, the better). Another commonality is that the other two functions result in improvements for the rankings $\varrho_2$, $\varrho_3$, and $\varrho_4$. However, they do not exceed an improvement of 3% (in the case of the 11 point average, an improvement of up to 8% was discovered). It is worth observing

that the differences between all the rankings do not exceed 5%.

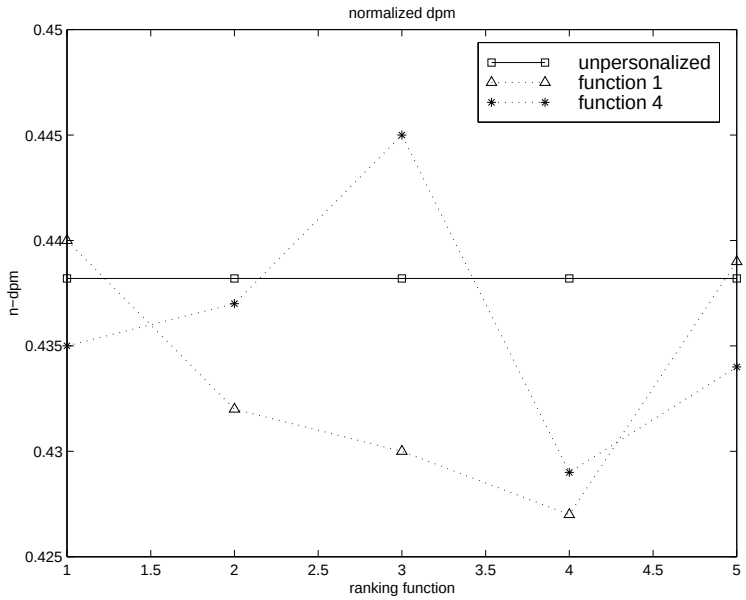


Figure 4.9: n-dpm for the testing set: 16 queries

This assessment is verified with the test set of the remaining 16 documents (Figure 4.9). The ranking of $\varrho_1$ yields slightly worse results than does $\varrho_2$. A conclusion of this evaluation together with the evaluation of the 11 point average measure would therefore suggest to use ranking $\varrho_2$ together with one of the two adjustment functions that yielded converging profiles and improved the search results slightly.

### 4.2.3 Filtering

Filtering is done as described above. Normalized personalized rankings together with threshold values provide a basis to decide whether or not a document is relevant. For the experiments, the thresholds ranged from .4 to .9.

In the following, $\mathcal{D}$ denotes a set of result documents, $\mathcal{R} \subseteq \mathcal{D}$ the set of relevant documents (as judged by the user), $\mathcal{I} = \mathcal{D} - \mathcal{R}$ the set of non-relevant documents (as judged by the user), and $\mathcal{F} \subseteq \mathcal{D}$ denotes the set of documents that are judged by the system to be irrelevant and should thus be excluded from the result list.

The following values are of interest:

- $\left|\mathcal{R} \cap \mathcal{F}\right| = \left|\mathcal{R} - (\mathcal{D} - \mathcal{F})\right|$

  How many relevant documents have been considered to be irrelevant? A low value indicates good performance.

- $\left|\mathcal{R} \cap (\mathcal{D} - \mathcal{F})\right| = \left|\mathcal{R} - \mathcal{F}\right|$

  How many relevant values have been considered to be relevant? A high value indicates good performance.

- $\left|\mathcal{I} \cap \mathcal{F}\right| = \left|\mathcal{I} - (\mathcal{D} - \mathcal{F})\right|$

  How many non-relevant items have been considered to be irrelevant? A high value indicates good performance.

- $\left|\mathcal{I} \cap (\mathcal{D} - \mathcal{F})\right| = \left|\mathcal{I} - \mathcal{F}\right|$

  How many irrelevant documents have been considered to be relevant? A high value indicates bad performance.

When comparing the different filtering algorithms, it is convenient to refer to the relative number of documents that have been filtered, i.e., by dividing the first two sets by $\left|\mathcal{R}\right|$ and the third and fourth set by $\left|\mathcal{I}\right|$. In this relative case, both $\frac{\left|\mathcal{R}-(\mathcal{D}-\mathcal{F})\right|}{\left|\mathcal{R}\right|} = 1.0 - \frac{\left|\mathcal{R}-\mathcal{F}\right|}{\left|\mathcal{R}\right|}$ and $\frac{\left|\mathcal{I}-(\mathcal{D}-\mathcal{F})\right|}{\left|\mathcal{I}\right|} = 1.0 - \frac{\left|\mathcal{I}-\mathcal{F}\right|}{\left|\mathcal{I}\right|}$ hold.[12]

The discussion will hence concentrate on the first and on the third value. The first value describes how many relevant documents have incorrectly been

[12] That is because of $(\mathcal{R} \cap \mathcal{F}) \cup (\mathcal{R} - \mathcal{F}) = \mathcal{R}$ and $\mathcal{R} \cap \mathcal{F} = \mathcal{R} - (\mathcal{D} - \mathcal{F})$.

considered to be irrelevant. The third value represents the number of irrelevant documents that correctly have been considered to be irrelevant.

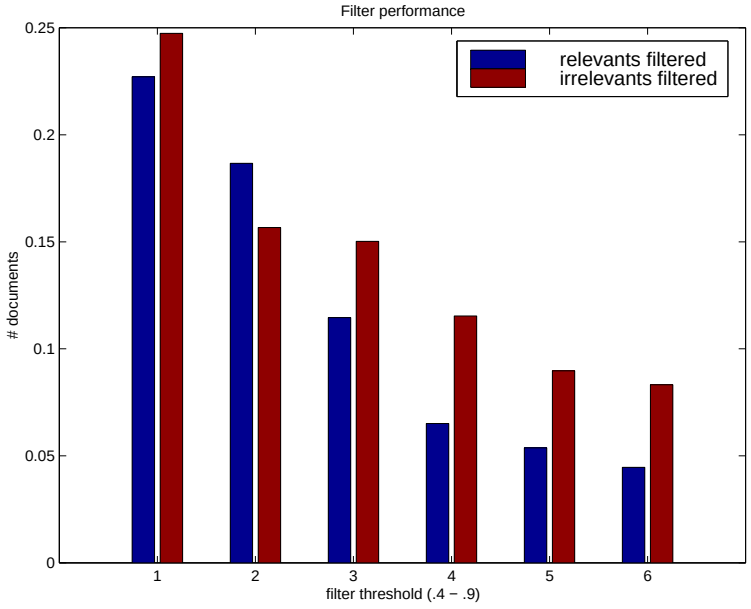


Figure 4.10: Average filter performance: Training. Adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$, ranking $\varrho_1$. The x-axis represents threshold values from .4 to .9, the y-axis represents the ratio of relevant (non-relevant) documents.

Figures 4.10 and 4.11 show the best performances (out of 4*5=20 such figures) for the training set of 32 documents. They are very similar in that they exhibit good filtering performance for thresholds greater than .7. For a threshold of .9, both filters filter twice as many irrelevant documents than relevant ones. However, with that threshold, only about one to two non-relevant documents out of twenty (6.2% to 8.1%) are filtered. For smaller thresholds, both the value of relevant and non-relevant documents that are filtered increase.

It is not surprising that the best combination of adjustment function and ranking is the same as in the evaluation of the eleven point precision average since the filter is based on the rankings that have been evaluated in the previous section.

Figure 4.12 shows the filtering performance of Figure 4.10 for the testing set.

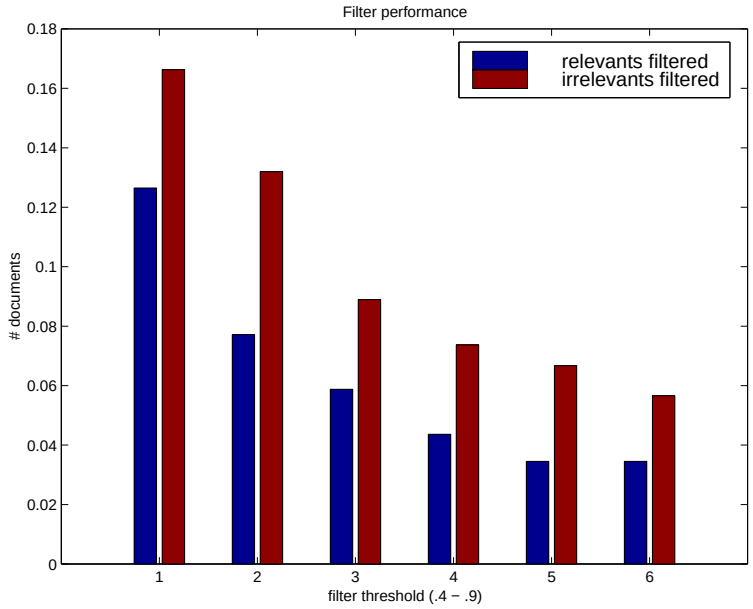


Figure 4.11: Average filter performance: Training. Adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$, ranking $\varrho_2$. The x-axis represents threshold values from .4 to .9, the y-axis represents the ratio of relevant (non-relevant) documents.

The figure indicates that, for large threshold values, there are two to three times more irrelevant than relevant documents filtered. However, one should note that the absolute number (8% of 20, or 1.6 documents per query) is rather small.

Although the filter improves search results, these improvements are rather small. This suggests that the system performs better in ranking than in filtering. This is likely due to the fact that the decision "relevant" vs. "non-relevant" is very coarse and that mistakes are easily made. In the case of re-ranking, switching the position of two items does, in general, not greatly affect the quality of the results.

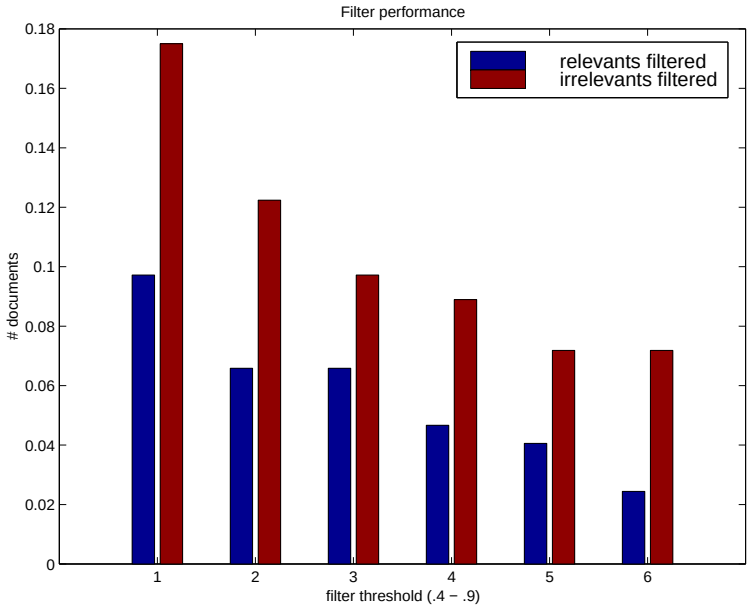


Figure 4.12: Average filter performance: Testing. Adjustment function $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$, ranking $\varrho_1$. The x-axis represents threshold values from .4 to .9, the y-axis represents the percentage of relevant (non-relevant) documents.

## 4.3 Discussion

This chapter introduced two applications for the user profiles: Re-ranking and filtering search results. It has been found that the quality of Internet search engines (in the tests, ProFusion was chosen, but it is most likely that the results carry over to other search engines) is rather poor: only one half of all retrieved documents are relevant to the user. Reasons for this include the sheer size of the database to be searched (the Internet) and a lack of personalization.

With the presented system, search results can be slightly improved.

Four different ways of determining user profiles were combined with five different re-ranking functions. Two interest adjustment functions performed better

than the other two: $\Delta\iota(c_i) = \log \frac{time}{\log length} \cdot \gamma(d, c_i)$ and $\Delta\iota(c_i) = \log \frac{time}{\log \log length} \cdot \gamma(d, c_i)$.

These better functions share the commonality of assigning less weight to the length of the surfed pages than to the time spent on them.

In combination with the above adjustment functions, two re-rankings have been found to outperform the others: $\varrho_1(d_j) = w(d_j) \cdot \left(.5 + \frac{1}{4} \sum_{i=1}^{4} \tilde{\iota}_i(d_j)\right)$ and $\varrho_2(d_j) = w(d_j) + \frac{1}{4} \sum_{i=1}^{4} \tilde{\iota}_i(d_j)$.

They share the commonality of assigning equal importance to the ranking that was returned by ProFusion and the (calculated) personal interests in the corresponding documents. These findings were confirmed using two evaluation strategies applied to the system: the eleven point precision average, and the normalized distance-based performance measure.

Personalization improved search results in terms of the above criteria: an increase of up to 8% was detected for the 11 point precision average, and an increase of 5% for the n-dpm was found.

The improvements were less compelling for the second application: filtering. The chosen filtering mechanism uses the rankings as provided by the re-ranking algorithms and introduces thresholds to determine which documents are relevant and which are not. Not surprisingly, the same combinations of interest adjustment function and re-ranking functions turned out to yield the best results.

However, this binary partitioning of a set of documents seems to be too coarse for the filtering algorithm. The improvements were minor: Of the 20 top documents in a query's result set, on average only one document is filtered, and on

average, every third filtered document will be relevant (since, in general, twice as many irrelevant than relevant documents are filtered).

A major result of this chapter is that even though performance improvements are not enormous, they are at least *possible*. This is insofar noteworthy as it should not be forgotten that profile creation occurred fully automatically, without any explicit user feedback.

It is unlikely that the poor improvements are a result of badly chose re-ranking or filtering algorithms. Rather, the reason for this lies probably in the lack of accuracy in the user profiles. Search result improvements will hence most likely be the result of more accurate user profiles.

# Chapter 5

# Conclusions and Future Work

This thesis presented a novel way of modeling user interests as a weighted ontology. With this approach, except for the daily surfing activities, no interaction with the user is required.[1]

A notion of *convergence* for user profiles has been developed, and the profiles have been evaluated in terms of this convergence as well as the actual interests they should represent. Convergence means that, after some time, there is an almost constant set of categories that have been determined to reflect the user's interests. Of course, user interests shift, and this is why it is proposed that the profile maintenance program runs continuously, i.e., is trained perpetually. Two out of four approaches to profile generation have been shown to yield converging profiles. Sixteen user profiles have been evaluated, and they were found to converge and to reflect actual interests fairly well (3.5 on a scale from 0 to 5).

The most important characteristics of the proposed approach include lack of explicit interaction and the hierarchical structure of the profile representation (4,400 nodes).

[1] As of now, the profile maintenance has to be run every second day to extract data from the caches, but in the future, this will be done by a continuously running background process.

Because of the intimate nature of data, privacy issues must be considered. The present system leaves the profiles on the user's machine and only post-processes the data that is to be personalized. However, there are applications where profiles must be made publicly available in order to match them with other profiles or queries.

In principle, there is a wide range of applications for user profiles. This is reflected in the fact that there are many personalization systems, of which 45 have been classified and described.

For this thesis, the field of information retrieval was chosen. The profiles are used to re-rank and filter search results as returned by the ProFusion meta search engine. The four ways of creating profiles have been combined with five ways of re-ranking the results, and with six ways of filtering them. The re-ranked results were evaluated by referring to the 11 point precision average and the normalized distance performance measure. In terms of the first measure, a performance increase of up to 8% was discovered.

With this measure, document sets are partitioned into two sets, the relevant and the non-relevant ones. Within each of these two sets, all documents are considered to be equally interesting (or not). The n-dpm measure addresses this problem by defining a distance between two rankings. In comparison with ProFusion's ranking, the personalized rankings turn out to be 3% more "similar" to the user ranking.

The above rankings have then been used to implement a simple filtering algorithm by introducing thresholds. Retrieved documents above that threshold are considered to be relevant, and the others are considered to be irrelevant. The filtering results were rather modest. When emphasis is put on avoiding filter-

ing relevant documents, the best filtering approach filters roughly one irrelevant document out of a total of 20.

The performance improvements that have been achieved are modest. However, it has been shown that profile generation without explicit user interaction is at least possible. There are only a few systems that fully automatically create profiles, and they are rarely thoroughly evaluated. Personal WebWatcher [34] is the system that is closest to the presented one. However, the profiles are not very detailed. In fact, there are only two classes, relevant and irrelevant, and their centers are represented by weighted keyword vectors.

Future work will focus on the profile generator. Several suggestions have been presented and discussed in section 3.4.1. The most promising approach seems to be related to personalizing the structure of the profiles rather than only the weights that are associated with the nodes. Furthermore, it seems possible to use the surfed documents for re-training the classification algorithm. Finally, it should be evaluated whether ot not the results would improve if some restricted explicit user feedback was taken into account.

There is also a need to make the programs more usable. In its current state, the user explicitly decides when and what parts of the database (copies of the caches) have to be characterized. Minor modifications are necessary to make the program run in the background.

# Appendix A

# Profile Convergence

This appendix contains figures showing the convergence of user profiles for different interest adjustment functions. The graphs show how many categories account for 95% of the accumulated interest values for all categories. The "time periods" actually represents chunks of a same number of documents, ordered w.r.t. to the time of access.

Figures A.1, A.2, and A.3 show the number of categories over time for the adjustment function $\Delta\iota(c_i) = \frac{time}{length} \cdot \gamma(d, c_i)$. Figures A.4, A.5, and A.6 show the profiles for the adjustment function $\Delta\iota(c_i) = \log\frac{time}{length} \cdot \gamma(d, c_i)$. Finally, figures A.7, A.8, and A.9 show the profiles for the adjustment function $\Delta\iota(c_i) = \log\frac{time}{\log length} \cdot \gamma(d, c_i)$. The adjustment function $\Delta\iota(c_i) = \log\frac{time}{\log\log length} \cdot \gamma(d, c_i)$ has been discussed in chapter 3 (figures 3.6, 3.7, and 3.8).

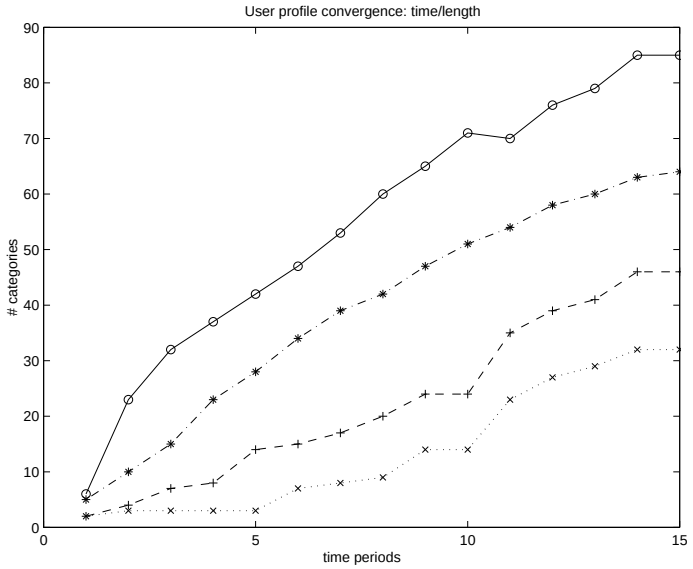


Figure A.1: Convergence of four profiles with less than 50 categories; adjustment function $\frac{time}{length}$

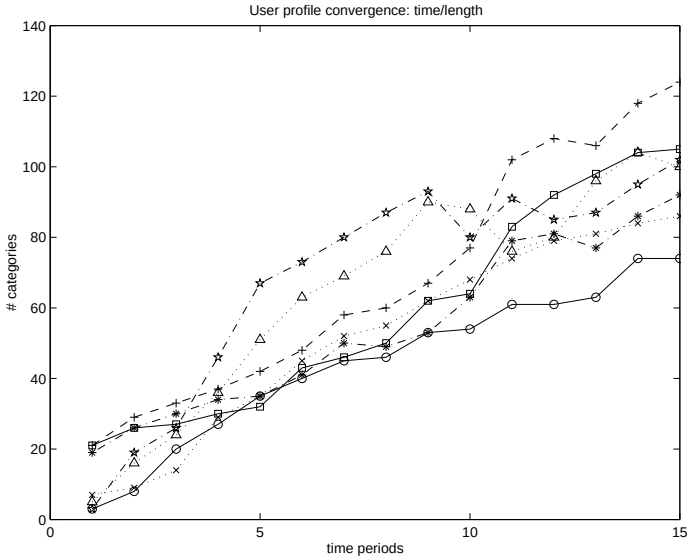


Figure A.2: Convergence of seven profiles with less than 100 categories; adjustment function $\frac{time}{length}$

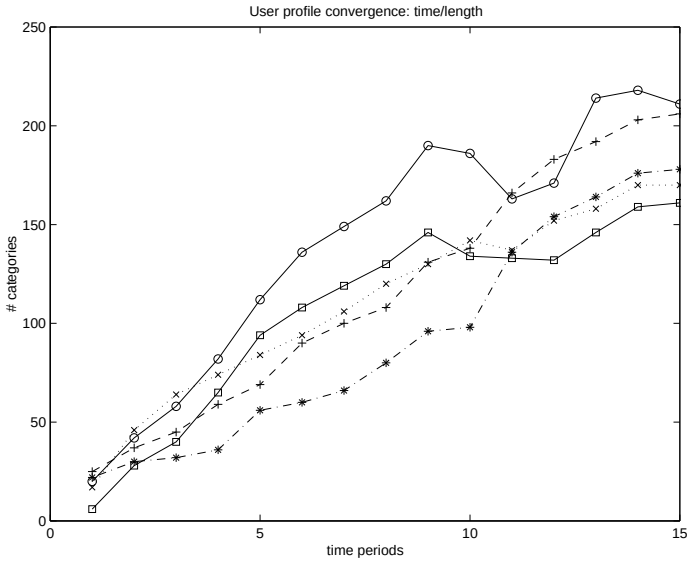


Figure A.3: Convergence of five profiles with less than 150 categories; adjustment function $\frac{time}{length}$

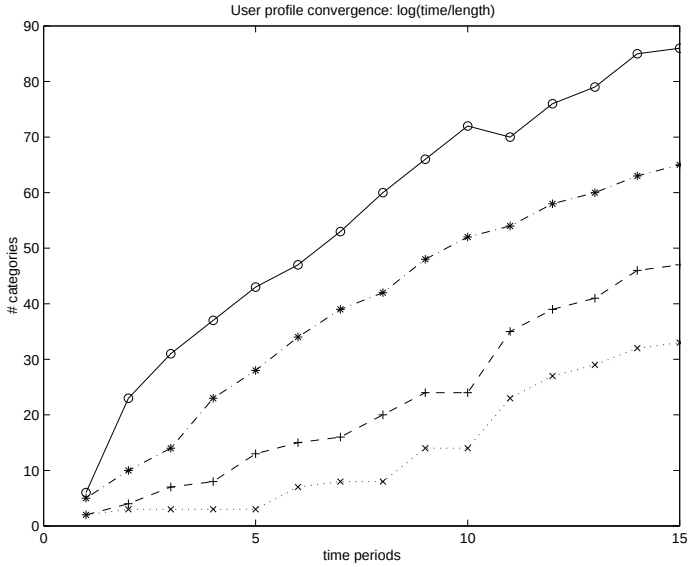


Figure A.4: Convergence of four profiles with less than 50 categories; adjustment function $\log \frac{time}{length}$

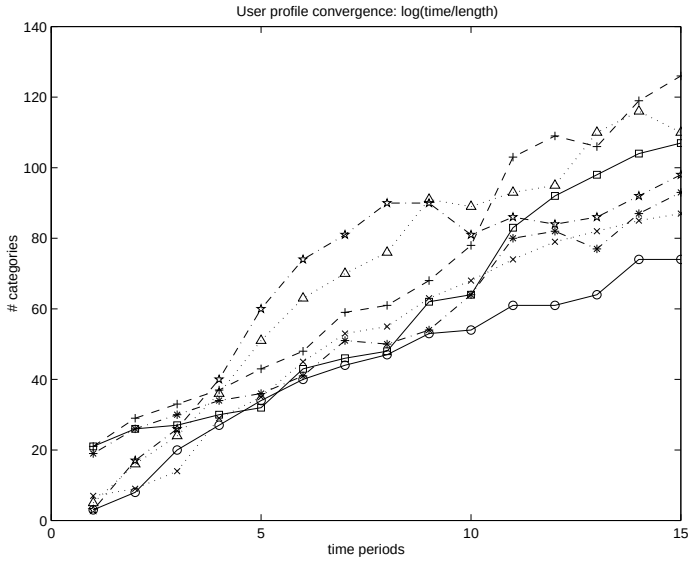


Figure A.5: Convergence of seven profiles with less than 100 categories; adjustment function $\log \frac{time}{length}$

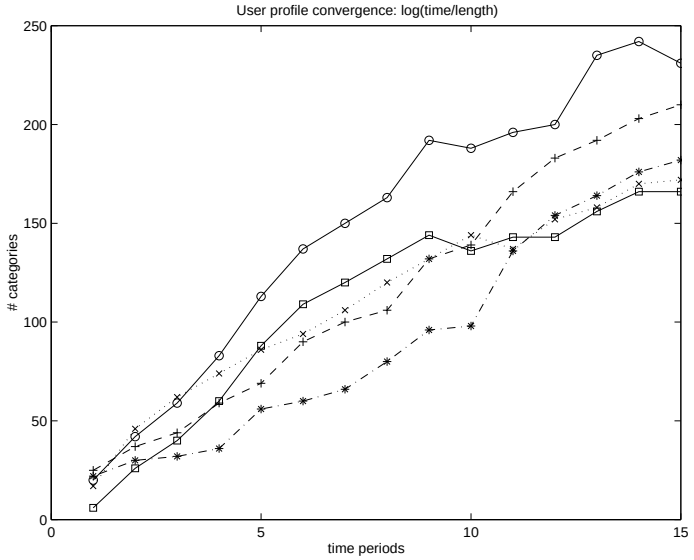


Figure A.6: Convergence of five profiles with less than 150 categories; adjustment function $\log \frac{time}{length}$

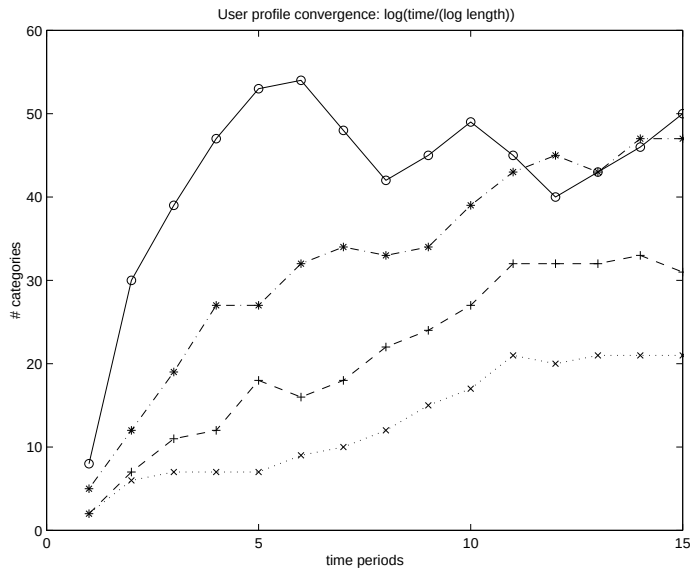


Figure A.7: Convergence of four profiles with less than 50 categories; adjustment function $\log \frac{time}{\log length}$

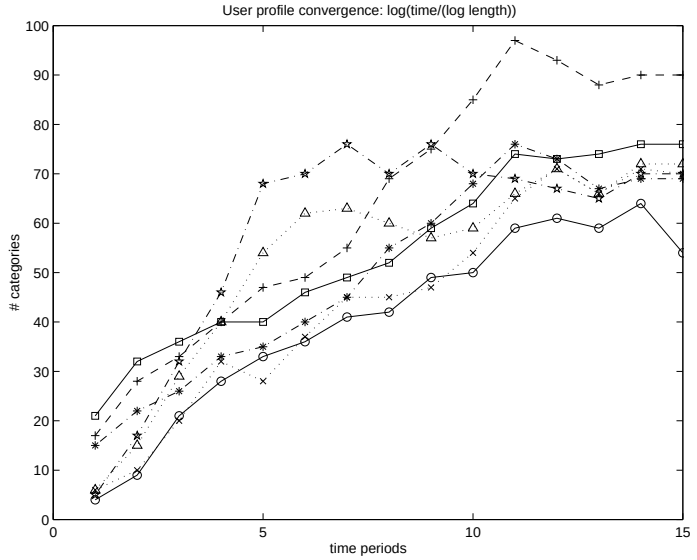


Figure A.8: Convergence of seven profiles with less than 100 categories; adjustment function $\log \frac{time}{\log length}$

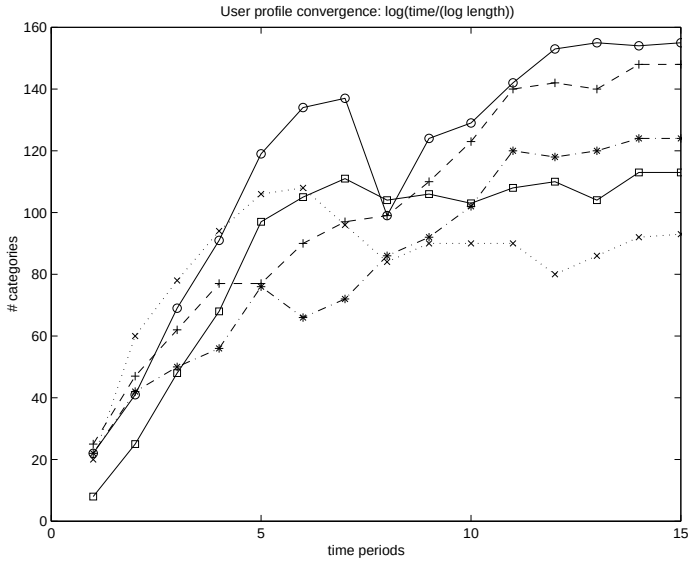


Figure A.9: Convergence of five profiles with less than 150 categories; adjustment function $\log \frac{time}{\log length}$

# Appendix B

# Implementation

## B.1 Architecture

Figure B.1 shows a data flow diagram for the system that has been developed for this thesis. The evaluation modules are not included.

The system can be divided into three parts: the observer, the user profiler, and the retriever. The observer extracts data from the caches and calculates the interest in the pages. The current cache and database is displayed in a window (figure B.2) and can be browsed. The two subprocesses, the BrowserSpy and the SpyObserver, extract the data (BrowserSpy) and calculate the interest factors (SpyObserver). The interest functions to be applied can interactively be chosen by the user.

The user profiler characterizes the surfed pages. This is done incrementally, so only new pages need to be characterized. The resulting categories and their weights are passed to the SpyObserver, which calculates the interests in the categories and passes them back to the profiler. The profiler maintains the personalized ontology, i.e., the ontology and the weights of its nodes.

This weighted ontology is then used in the retriever part. The retriever contacts ProFusion, retrieves the results and re-ranks them with respect to the personalized ontology.

The JavaDoc documentation of the source code can be found at

`http://www.ittc.ukans.edu/~alexp/thesis/javadoc`.

## B.2 Netscape Cache Format

It has been deduced from the Mozilla sources that the database index file, `fat.db`, is organized as follows. It is clustered into blocks of size 16KB or 4KB. The four bytes at position 12 of the first block contain the block length. The four bytes starting at position 56 of the first block contain the number of blocks. Starting with the second block, data is encoded:

- the first two bytes contain the number of entries in this block, each entry representing one file in the cache
- following these two first bytes there are pairs of two-byte-blocks where the first component contains the (relative) offset of the actual URL, and the second component contains the offset of the structure containing the rest of the information on a particular item in the cache. The number of entries (first two bytes of a block) has to be divided by two since it does not represent the number of pairs but rather the total number of components.
- Each second component of the (url-offset, struct offset) pairs then points to a structure which is organized as follows:

- 4 bytes: length of structure
- 4 bytes: version of cache format
- 4 bytes: time of last modification, GMT in seconds
- 4 bytes: time of last access, GMT in seconds
- 4 bytes: expiry date, GMT in seconds
- 4 bytes: content length
- 1 byte: the corresponding page a web site?
- 4 bytes: lock date
- 4 bytes: length of the cached file's name (including terminating null) the filename

The content type should be in accordance with the above data, but this seems not always to be the case. Hence, the following approach was chosen: Depending on the suffix of the filename, it is decided if the corresponding page is a text file: if the suffix is .HTM or does not exist at all, then it probably is a text page.

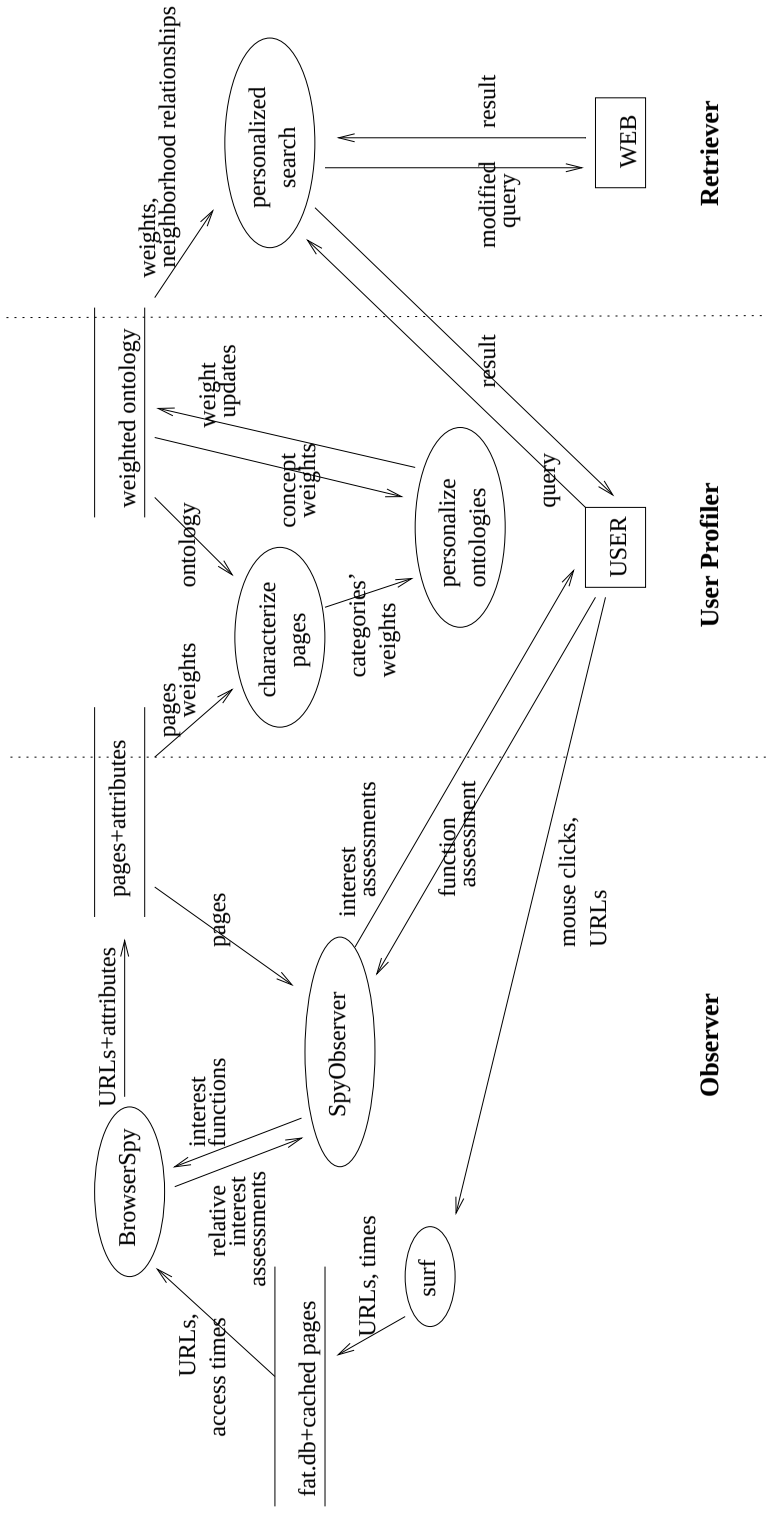


Figure B.1: Architecture

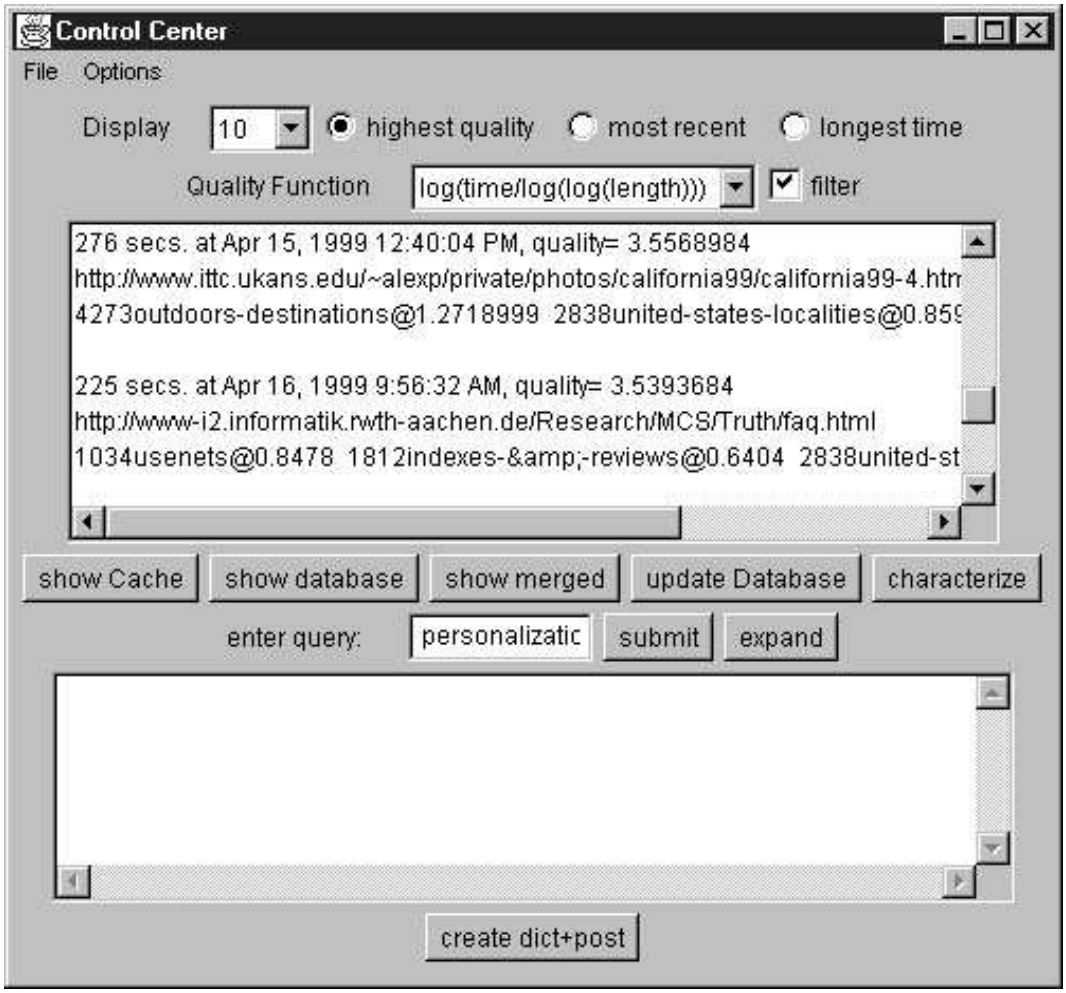


Figure B.2: Screenshot

# Bibliography

[1] R. Armstrong, D. Freitag, T. Joachims, and T. Mitchell. WebWatcher: A Learning Apprentice for the World Wide Web. In *Proc. AAAI Spring Symp. on Information Gathering from Heterogeneous, Distributed Environments*, March 1995.

[2] F. Asnicar and C. Tasso. ifWeb: a prototype of user model-based intelligent agent for documentation filtering and navigation in the World Wide Web. In *Proc. 6th Intl. Conf. on User Modeling*, Chia Laguna, Sardinia, June 1997.

[3] M. Balabanović. An adaptive web page recommendation service. In *Proc. 1st Intl. Conf. on Autonomous Agents*, Marina del Rey, CA, USA, February 1997.

[4] M. Balabanović and Y. Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *Proc. 1995 AAAI Spring Symposium on Information Gathering from Hetereogeneous, Distributed Resources*, 1995.

[5] R. Barrett, P. Maglio, and D. Kellem. How to personalize the Web. In *Proc. ACM CHI'97*, Atlanta, USA, 1997.

[6] D. Billsus and M. Pazzani. Learning probabilistic user models. In *Proc. 6th Intl. Conf. on User Modeling, Workshop on Machine Learning for User Modeling*, Chia Laguna, Sardinia, June 1997.

[7] Communications of the ACM 42:2, February 1999.

[8] E. Casasola. ProFusion PersonalAssistant: an agent for personalized information filtering on the WWW. Master's thesis, The University of Kansas, Lawrence, KS, 1998.

[9] E. Casasola and S. Gauch. Intelligent information agents for the World Wide Web. Technical Report ITTC-FY97-11100-1, Information and Telecommunication Technology Center, The University of Kansas, 1997.

[10] L. Chen and K. Sycara. A personal agent for browsing and searching. In *Proc. 2nd Intl. Conf. on Autonomous Agents*, pages 132–139, New York, USA, 1998.

[11] P. Chesnais, M. Mucklo, and J. Sheena. The fishwrap personalized news system. In *Proc. IEEE 2nd Intl. Workshop on Community Networking Integrating Multimedia Services to the Home*, Princeton, New Jersey, USA, June 1995. http://fishwrap.mit.edu.

[12] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, 1992. ISBN 0-13-463837-9.

[13] E. Gabber, P. Gibbons, D. Kristol, Y. Matias, and A. Mayer. Consistent, yet anonymous web access with LPWA. *Communications of the ACM*, 42(2):42–47, February 1999.

[14] S. Gauch, A. Pretschner, L. Gerhard, and X. Zhu. Obiwan - ontology based informing web agent navigation, 1999. http://www.ittc.ukans.edu/obiwan/agents/poster/obiwan-entry.html.

[15] S. Gauch, G. Wang, and M. Gomez. ProFusion: Intelligent fusion from multiple distributed search engines. *J. of Universal Computing*, 2(9), September 1996.

[16] L. Gerhard. Visualizing multi-dimensional conceptual hierarchies. Master's thesis, University of Kansas, Lawrence, Kansas, April 1999.

[17] R. Glass. Inspections - some surprising findings. *Communications of the ACM*, 42(4):17–19, April 1999.

[18] D. Harman. Evaluation Techniques and Measures. In *Proc. 4th Text Retrieval Conference (TREC-4)*, pages A–6–A–14, October 1996.

[19] T. Joachims, D. Freitag, and T. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proc. IJCAI'97*, August 1997.

[20] T. Kamba, K. Bharat, and M. Albers. The Krakatoa Chronicle - an interactive, personalized newspaper on the Web. In *Proc. 4th Intl. WWW Conf.*, pages 159–170, 1995.

[21] H. Kautz, B. Selman, and A. Milewski. Agent amplified communication. In *Proc. 8th Annual Conference on Innovative Applications of AI*, Portland, Oregon, USA, August 1996. As cited in [55].

[22] T. Kelley. Surfing in circles and loving it. *The New York Times*, 1999. January 21st, 1999.

[23] F. Kilander. A brief comparison of news filtering software, 1996. http://www.dsv.su.se/~fk.

[24] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. GroupLens: Applying collaborative filtering to Usenet news. *Communciations of the ACM*, 40(3):77–87, March 1997.

[25] T. Kurki, S. Jokela, R. Sulonen, and M. Turpeinen. Agents in delivering personalized content based on semantic metadata. In *Proc. 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace*, pages 84–93, Stanford, USA, 1999.

[26] W. Lam, S. Mukhopadhyay, J. Mostafa, and M. Palakal. Detection of shifts in user interests for personalized information filtering. In *Proc. ACM SIGIR'96*, Zurich, Switzerland, 1996.

[27] H. Lieberman, N. van Dyke, and A. Vivacqua. Let's browse: a collaborative Web browsing agent. In *Proc. Intl. Conf. on Intelligent User Interfaces*, January 1999.

[28] H. Liebermann. Letizia: An agent that assists Web browsing. In *Proc. Intl. Conf. on AI*, Montréal, Canada, August 1995.

[29] H. Liebermann. Autonomous interface agents. In *Proc. ACM Conf. on Computers and Human Intraction (CHI'97)*, Atlanta, USA, May 1997.

[30] T. Malone, K. Grant, F. Turbak, S. Brobst, and M. Cohen. Intelligent information sharing systems. *Communciations of the ACM*, (30):390–402, May 1987.

[31] X. Meng and Z. Chen. Improve Web search accuracy using personalized profiles, January 1999. http://www.cs.panam.edu/~meng/unix-home/Research/DataMine/Writing/spects99.ps.

[32] M. Minsky. *The Society of Mind.* Simon and Schuster, New York, 1987.

[33] M. Mitchell. *An Introduction to Genetic Algorithms.* "A Bradford Book". The MIT Press, 1996. ISBN 0-262-63185-7.

[34] D. Mladenić. Personal WebWatcher: design and implementation. Technical Report IJS-DP-7472, J. Stefan Institute, Department for Intelligent Systems, Ljubljana, 1998.

[35] D. Mladenić. Text-learning and intelligent agents. Technical Report IJS-DP-7948, J. Stefan Institute, Department for Intelligent Systems, Ljubljana, 1998.

[36] M. Montebello, W. Gray, and S. Hurley. A personable evolvable advisor for WWW knowledge-based systems. In *Proc. 1998 Intl. Database Engineering and Application Symposium (IDEAS'98)*, pages 224–233, Cardiff, Wales, UK, July 1998.

[37] M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *Proc. 17th Annual Intl. ACM-SIGIR Conf. on Research and Development in Information Retrieval*, pages 272–281, 1994.

[38] A. Moukas. Amalthaea: Information discovery and filtering using a multiagent evolving ecosystem. In *Proc. 1st Intl. Conf. on the Practical Application of Intelligent Agents and Multi Agent Technology*, London, 1996.

[39] D. Nichols. Implicit rating and filtering. In *Proc. 5th DELOS Workshop on Filtering and Collaborative Filtering*, Budapest, Hungary, November 1997. ISBN 2-912335-04-3.

[40] D. Oard and J. Kim. Freely available information filtering systems, 1998. http://www.clis.umd.edu/dlrg/filter/software.html.

[41] D. Oard and G. Marchionini. A conceptual framework for text filtering. Technical Report EE-TR-96-25 CAR-TR-830 CLIS-TR-9602 CS-TR-3643, University of Maryland, May 1996.

[42] K. Oostendorp, W. Punch, and R. Wiggins. A tool for individualizing the Web. In *Proc. 2nd WWW Conference: Mosaic and the Web*, 1994.

[43] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill&Webert: identifying interesting web sites. In *Proc. 13th Natl. Conf. on Artificial Intelligence*, 1996.

[44] E. Rich. User modeling via stereotypes. *Cognitive Science*, 3:329–354, 1979. As cited in [41].

[45] J. Rucker and M.J. Polanco. Siteseer: Personalized navigation for the web. *Communications of the ACM*, 40(3):73–75, 1997.

[46] H. Sakagami and T. Kamba. Learning personal preferences on online newspaper articles from user behaviors. In *Proc. 6th Intl. World Wide Web Conf.*, pages 291–300, 1997.

[47] H. Sakagami, T. Kamba, A. Sugiura, and Y. Koseki. Effective personalization on push-type systems - visualizing information freshness. In *Proc. 7th Intl. WWW Conf.*, Brisbane, Australia, April 1998.

[48] G. Salton. *Automatic Text Processing.* Addison-Wesley, 1989. ISBN 0-201-12227-8.

[49] E. Savia, T. Kurki, and S. Jokela. Metadata based matching of documents and user profiles. In *Proc. 8th Finnish Artificial Intelligence Conference, Human and Artificial Information Processing*, pages 61–69, 1998. As cited by [25].

[50] B. Sheth. A learning approach to personalized information filtering. Master's thesis, Massachusetts Institute of Technology, February 1994.

[51] B. Sheth and P. Maes. Evolving agents for personalized information filtering. In *Proc. IEEE Conf. on AI for applications*, 1993.

[52] H. Sorensen and M. McElligott. PSUN: a profiling system for Usenet news. In *Proc. CIKM'95 workshop on Intelligent Information Agents Workshop*, Baltimore, USA, December 1995.

[53] A. Stefani and C. Strappavara. Personalizing access to web sites: The SiteIF project. In *Proc. 2nd Workshop on Adaptive Hypertext and Hypermedia HYPERTEXT'98*, Pittsburgh, USA, June 1998.

[54] C. Thomas and G. Fischer. Using agents to personalize the web. In *Proc. ACM IUI'97*, pages 53–60, Orlando, Florida, USA, 1997.

[55] A. Vivacqua. Agents for expertise location. In *Proc. 1999 AAAI Spring Symposium Workshop on Intelligent Agents in Cyberspace*, pages 9–13, Stanford, USA, 1999.

[56] D. Widyantoro, J. Yin, M. El Nasr, L. Yang, A. Zacchi, and J. Yen. Alipes: A swift messenger in cyberspace. In *Proc. 1999 AAAI Spring Symposium*

*Workshop on Intelligent Agents in Cyberspace*, pages 62–67, Stanford, USA, 1999.

[57] T. Yan and H. Garcia-Molina. SIFT - a tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conf.*, pages 177–186, 1995.

[58] Y. Yao. Measuring retrieval effectiveness based on user preference of documents. *J. of the American Society for Information Science*, 46(2):133–145, 1995.

[59] X. Zhu, S. Gauch, L. Gerhard, N. Kral, and A. Pretschner. Ontology based web site mapping for information exploration, 1999. Submitted to CIKM'99; http://www.ittc.ukans.edu/obiwan/publications/SIGIR99.html.

[60] H. Zimmermann. *Methoden und Modelle des Operations Research.* Vieweg, 1992. ISBN 3-528-18917-7.