Personalizing Interactions with Information Systems

Saverio Perugini and Naren Ramakrishnan
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061
Email: {sperugin,naren}@cs.vt.edu

August 21, 2002

Abstract

Personalization constitutes the mechanisms and technologies necessary to customize information access to the end-user. It can be defined as the automatic adjustment of information content, structure, and presentation tailored to the individual. In this chapter, we study personalization from the viewpoint of personalizing *interaction*. The survey covers mechanisms for information-finding on the web, advanced information retrieval systems, dialog-based applications, and mobile access paradigms. Specific emphasis is placed on studying how users interact with an information system and how the system can encourage and foster interaction. This helps bring out the role of the personalization system as a facilitator which reconciles the user's mental model with the underlying information system's organization. Three tiers of personalization systems are presented, paying careful attention to interaction considerations. These tiers show how progressive levels of sophistication in interaction can be achieved. The chapter also surveys systems support technologies and niche application domains.

Contents

1 Introduction

Personalization entails customizing information access, structure, and presentation to the end-user. While the roots of personalization can be traced back to information filtering [BC92] and recommender systems [RV97], the web has propelled personalization into a highly studied and legitimate research area. The explosion of online content and the advent of ubiquitous computing devices and information appliances [Ber00] have made personalization critical to the success of Internet applications. Personalization is achieved in information systems which afford complex, compelling, and user-adapted interactions. Studying how users interact with information systems and understanding the frustrations they experience provides ample motivation for personalization.

1.1 Why Personalize?

We begin with the quintessential information access paradigm on the web - browsing. Bush is regarded as the godfather of browsing as we know it today [Bus45]. In browsing, two distinct roles are seen. "The role of the author was to create the hypertext and the role of the user/reader was to browse through it. Thus, the reader was faced with the task of understanding the author's mental model of the hypertext documents in order to navigate the collection of linked nodes (hyperbase) effectively" [BC99].

Pre-defined, hardwired browsing interfaces in information systems have been succinctly characterized with phrases such as 'static hypertext' [BC99], 'strong authoring' [BC99], and 'one-size-fits-all' [Bru01, Chi97]. Such a rigid model assumes that the author's viewpoint is correct. The resulting mental mismatch problem has been identified as a legitimate research issue in [Bor86, Suc87]. The goal of personalization technologies is to help overcome this mismatch. Essentially the same issue arises in other information access paradigms, and a variety of delivery mechanisms.

1.2 Approaches to Personalized Interaction

Operationally, the word 'personalization' is broad and open to many interpretations. For instance, we can aim for a naturalness in interaction, interestingness of content, quality of web pages, or speed of access. Many surveys of personalization focus on technical distinctions of how information is tailored to end-users and the level at which it is targeted. Business schools have adopted terms such as 'real time,' 'one-to-one,' and 'check-box' personalization. Therefore, there are truly 'personalized views of personalization' [Rie00b]. For instance, the articles in the *Communications of the ACM* August 2000 special issue on personalization range from topics such as natural language dialogs, to web site restructuring, to manually customizable portals.

In this survey, we approach personalization from the viewpoint of personalizing *interaction*. Interaction with an information system is thus the common thread among all systems surveyed in this chapter. Distinctions are only made when they reveal differences among interaction paradigms. For instance, Amazon's recommender system might make better recommendations of books than another bookseller's but if they possess the same interaction paradigm, they are considered equivalent for our purposes. In fact, many personalization solutions do not even explicitly recognize the issue of interaction with an user; needless to say they are not surveyed here. Distinctions such as *content-based* and *collaborative*—very popular in the recommender systems community—thus do not find place in this chapter.

We posit that surveying personalization according to interactions of users [Mar97] is a more holistic approach to studying this subject. To the best of our knowledge, this survey is the first to employ this approach. The reader should keep in mind that we use the term *personalization* synonymously with personalized interaction.

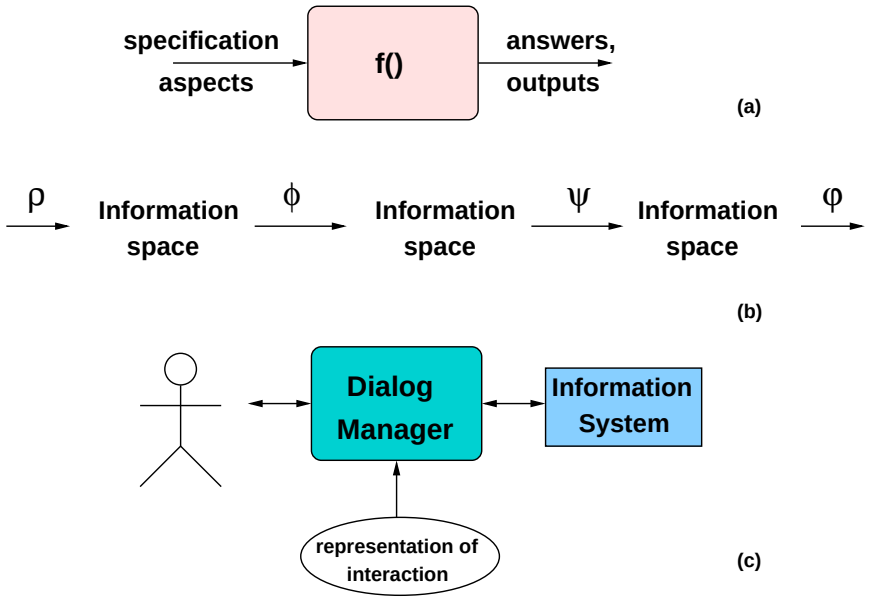

Figure 1: Three approaches to personalized interaction surveyed in this chapter: (a) templates for personalization, (b) operators for personalization, and (c) representing and reasoning about interaction.

1.3 Organization of this Survey

Three main approaches to personalizing interaction are outlined (see Fig. 1). The first approach is the terminal case where the system provides no support for maintaining interaction and the onus of personalization is shifted to the user. As shown in Fig. 1(a), the system effectively behaves as a functional engine mapping users' specification aspects into results. It doesn't recognize the fact that information access occurs in the context of an interaction.

Writing SQL queries in a database context is an example of a functional modeling. Although the user might interactively explore the database through a sequence of such queries, the system *per se* does not provide any support for interaction. We refer to these approaches as template-based and survey a sample of systems as they relate to information access on the web. We pay particular attention to systems that combine distinct modalities of information-seeking and which are especially relevant to the primary Internet access paradigms today.

Systems in the next tier provide a set of basic primitives for sustaining interaction. As shown in Fig. 1(b), these primitives are typically in the form of operators that successively transform an information space. The user is encouraged to apply these operators in a form that is suitable to his information-seeking activity. We say that such systems recognize and encourage interaction.

The third tier of systems are truly novel in that they explicitly represent and capture interaction. As Fig. 1(c) shows, interaction here resembles more a dialog between the user and the information system. A natural dialog is one where both parties interact to achieve the desired information-seeking goals. Systems in this tier are characterized by their representations, whose expressiveness and capabilities directly relate to the quality of personalized interaction. They are most capable of reconciling the mental mismatch issue introduced earlier.

Reader's Guide

The survey paints a picture of how personalization is conducted in each of the three tiers by showcasing a number of research projects. Section 2 describes templates for personalization. Section 3 introduces systems which afford expressive operators for personalization. Section 4 discusses representing and reasoning about interaction. A few

```
SELECT Name, Count
FROM States, WebCount
WHERE Name = T1
ORDER BY Count Desc
```

Figure 2: A WSQ query to rank states by how often they appear on the web (from [GW00]). This query has traditional SQL semantics. States and WebCount are relations. The schema of States is States(Name, Population, Capital). The WebCount relation, whose schema is WebCount(SearchExp, T1, T2, . . . , TN, Count), is populated by the results of a web search request. T1, T2, . . . , TN are values for parameters in SearchExp. Notice that all aspects of information-seeking necessary to determine an answer are provided in one stroke.

novel research projects are elaborated upon here. It should be remarked that the relative lengths of these sections do not reflect our view on their relative importance. They are more a reflection of the popularities of the template and operator-based approaches and the nascency of the representational approach. Project descriptions in each tier are also not meant to be exhaustive. In Section 5 we describe systems support tools and technologies that help achieve personalization. In Section 6 we describe a few niche domains that have witnessed significant investments in personalization. Section 7 concludes this chapter with some observations about the future.

2 Templates for Personalized Interaction

It can be argued that being able to set the background color for a desktop screen is a rudimentary form of personalization. Here, the goals of personalization have become so over-specified that the responsibility of achieving the personalization is shifted to the user, who must specify the settings.

A typical form of over-specification involves templates that are meant to be customized by the user. Another involves providing an expressive web query language, not unlike SQL. Their salient feature is a 'one-shot' [Bru01, MM00] style of personalization. This section surveys such approaches. Specifically, we start from a database perspective and describe the WSQ/DSQ project and probabilistic relational algebra. We next discuss web queries as templates and the use of templates for constructing personal information spaces.

2.1 WSQ/DSQ

The WSQ/DSQ (pronounced 'wisk-disk') project [GW00] at Stanford University attempts to bridge the gap between structured relational databases (DBs) and the unstructured web in support of an information retrieval request. WSQ (Web-Supported Queries) incorporates web search results into SQL queries over a database to enrich an answer. On the other hand, DSQ (Database-Supported (Web) Queries), its complement, leverages DB relations to enhance and explain web search results.

For the purposes of this chapter, it is sufficient to focus on the WSQ component. WSQ leverages web search results to provide a richer set of input parameters for a query against original relational data sources. In other words, the output of one query (the web search) is provided as input, along with the extant DB relations, to a master SQL query. In WSQ the primary mode of information-seeking is thus an SQL query and the secondary mode is web search.

The essential idea behind WSQ is to permit users to make references to web search requests within a traditional SQL query. A user writes a query that makes reference to a web search engine (WSQ/DSQ uses AltaVista and Google) and search terms, obtains an answer—a new relation, and proceeds to the next independent query that may (e.g., join the resulting relation with itself) or may not involve the resulting relation. A typical WSQ query is shown in Fig. 2. Interaction in WSQ is hence limited to issuing a query and obtaining an answer. This is referred

to as a one-shot interaction [Bru01] or a one-shot task [MM00]. Furthermore, traditional query processing cannot proceed until all attribute values initially populated with calls to a particular web search engine are replaced with corresponding URL answers.

Thus, interaction in WSQ is best modeled as a template for personalization. The order of the two information-seeking interactions in WSQ are determined *a priori* at query-creation time. This design of over-specification means that information-seeking parameters are provided in one stroke.

In fairness to the designers, the design in WSQ/DSQ is commensurate with the targeted applications (i.e., answering questions regarding comparisons or frequencies of items on the web, e.g., "Rank all countries in North America by how often they are mentioned by name on the web."). Nonetheless, WSQ is an interesting research project to study with respect to personalization and combining aspects of information-seeking. We view it as a limiting case of a personalization system.

2.2 Probabilistic Relational Algebra

In [FR97], Fuhr and Rölleke approach the problem of integrating aspects of information-seeking from a different angle. Specifically, the designers weave canonical IR parameters (e.g., weights, rankings, and probabilities) into a DBMS in order to enhance and improve retrieval.

Integration here is motivated by the fact that database management systems (DBMSs) lack a clean method to incorporate IR parameters. For instance, DBMSs do not adequately address vagueness, imprecision, and uncertainty which IR systems are designed for. DBMSs are however strongly grounded in theory and relations afford expressive query languages (QLs). IR systems, on the other hand, incorporate parameters well but have problems incorporating ground facts. In addition, there is limited expression in IR QLs that is currently addressed with ad hoc methods.

In order to weave standard IR parameters into a DBMS, Fuhr and Rölleke generalize traditional relational algebra, where probabilities are either 0 or 1, to the continuous range $[0 \ldots 1]$. Incorporating probabilities into tuples of DB relations is relatively straightforward. Ensuring that these probabilities are correctly propagated in an answer (after possibly complex joins or other operations) is difficult, due to uncertainty about the independence/dependence of tuples at query formulation time. Additional data could be modeled in relations to make such constraints explicit. With large DBs, however, such constraints and annotations embedded into relations may approach exponential levels.

Due to the explicit requirement to specify all information-seeking aspects at query formulation time, we classify Fuhr and Rölleke's work as a template for personalization. The system they developed does not allow users to specify parameters over time or leave parameters residual. Information-seeking sessions of this system resemble interaction with WSQ, with minor adaptations, e.g., instead of specifying which search engine to employ, the user indicates a probability threshold or an index weight.

2.3 Web Query Languages

Search Interfaces: Precursors to Web QLs

In order to combat cognitive frustrations experienced in browsing, many sites provide within-site search interfaces. We present three common user interface designs here. Fig. 3 (left) illustrates part of a book search tool available at Amazon.com. This type of search interface is typical and requires a user to associate search terms with categorical information (e.g., author, title, and publisher). The goal of such search interfaces is to avoid enumerating multiple browsing paths to terminal information (in this case, a book webpage). An alternative design, shown in Fig. 3 (right), is called a 'power-search' and has gained popularity in many e-commerce sites. A power-search more closely resembles a web query language. In other words, such tools include a small language for communicating inputs involving multiple query fields (possibly combined via `AND`s or `OR`s). The power-search of Amazon.com

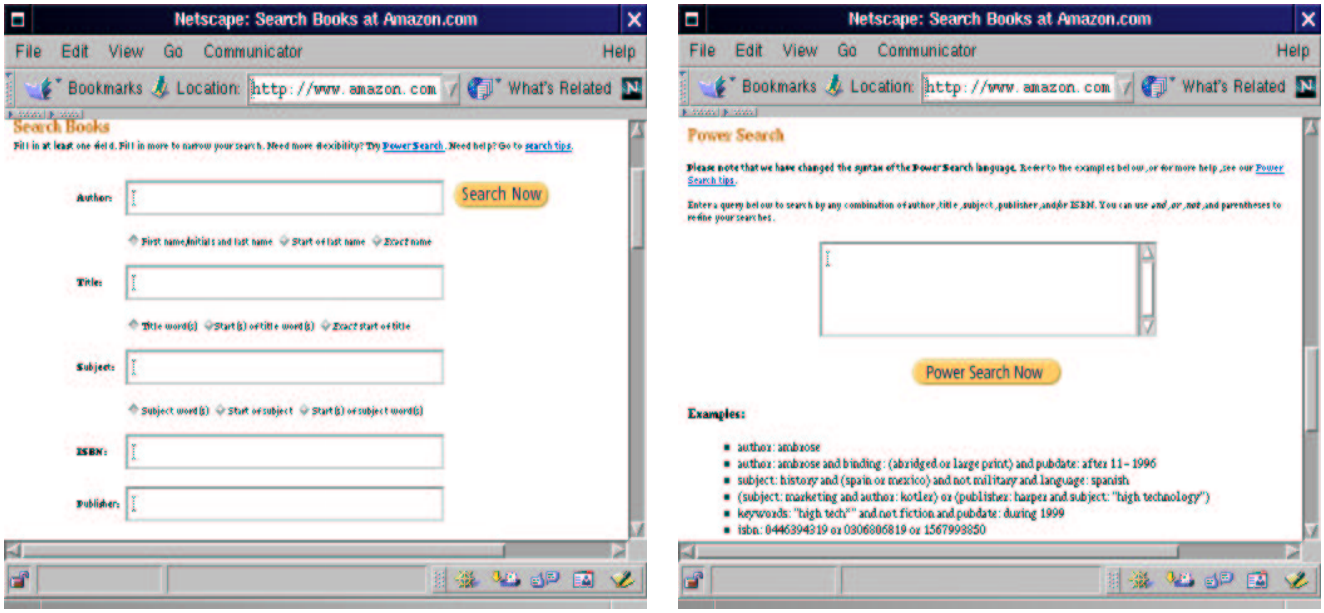

Figure 3: (left) A book search interface at Amazon.com. This interface contains multiple category-labeled text-fields, expecting input to belong to a category. Such a design attempts to hide hyperlink enumeration in web sites. (right) A power-search facility at Amazon.com that allows multiple query terms from different categories, but still requires categorical information.

Figure 4: A search facility of Amazon.com that allows the entry of free-form text.

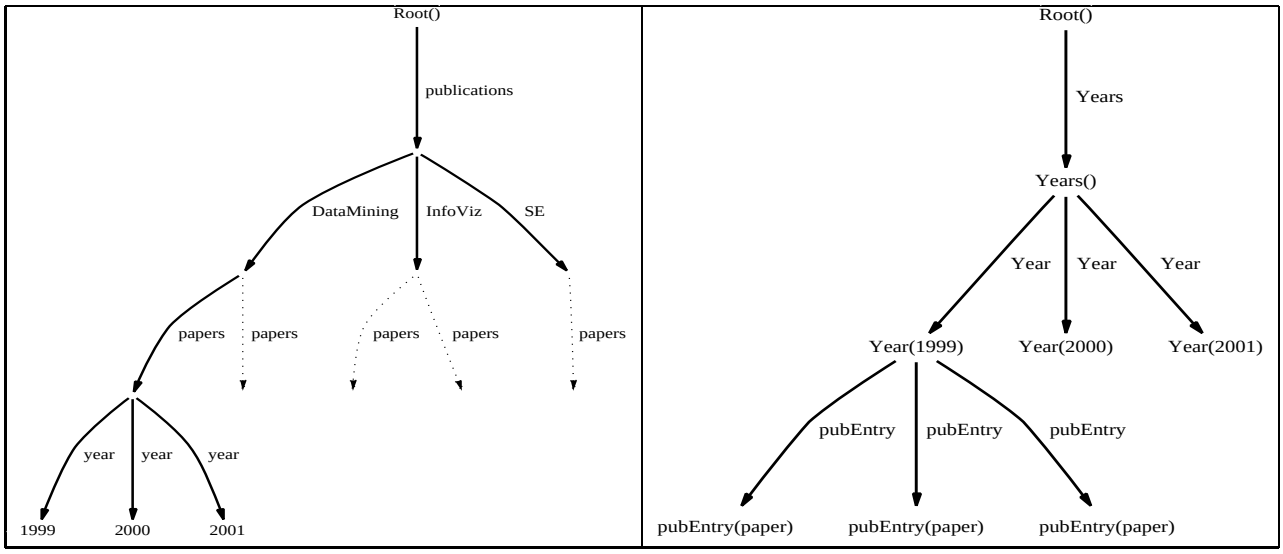

Figure 5: (left) Directed graph model of XML input to the StruQL query shown in Fig. 6. The publications are ordered by research area. (right) Directed graph model of XML output from the StruQL query shown in Fig. 6. Notice that publications are now ordered by year. Such XML data sources can be easily converted into a set of browsable webpages with tools such as XSL/XSLT [Cha99, Wid99].

shown in Fig. 3 (right) still requires a user to specify categorical information, however. From a user perspective, a less restrictive interface is a free-form text box (Fig. 4) that does not require categorical information. In such a design, users' query terms are matched against all attributes of an information nugget (e.g., webpage, book, or movie). The search facilities involved in the interfaces outlined above do not employ expressive QLs.

The systems presented below provide users with a sophisticated QL. Through the invocation of a query, such systems combine aspects of information-seeking with reconstruction properties of an information space.

Restructuring Semistructured Data

The semistructured data [ABS00, FLM98] and XML communities have embraced the idea of building, restructuring, and managing information spaces (e.g., web sites) with adaptations of traditional SELECT-FROM-WHERE queries. In this context, new and personalized information spaces may be constructed from DB relations, structured files, or semistructured data (e.g., XML). In addition, extant information spaces, such as web sites, may be restructured via a declarative query. This latter application is more interesting to study for our purposes.

For example, consider a researcher who disseminates his publications on his webpage via a research area browsing dichotomy. The original data may be stored in XML files (see left side of Fig. 5). If this researcher desires to restructure the hierarchical presentation with respect to year, he could write a semistructured data query (see Fig. 6). The output of the query is another XML file containing the publications of the researcher ordered by year of publication (see right side of Fig. 5). We use the StruQL query language [FFLS97, FFK$^+$98] to illustrate the query example in Fig. 6, but there exist a number of other semistructured and XML QLs such as Araneus, Florid, Lorel, WebOQL, WIRM, YAT, XSL/XSLT, and XML-QL [FLM98].

A query to restructure an information space actually mixes two distinct modalities of information-seeking. Typically, the data to restructure is a subset of an information space and retrieved via the WHERE clause of a semistructured data query. The WHERE clause thus serves as a match operator. Once data is bound to variables in a WHERE clause, manipulation of those variables within the CONSTRUCT clause of a query (the LINK clause in the case of StruQL) restructures the space. Thus, reconstruction activities take place following a retrieval or match

```
{ WHERE root in publications.xml,
        root -> "publications".
        ("DataMining" | "InfoViz" | "SE")."paper" -> paper,
        paper -> attribute -> attributeValue

  COLLECT Root(), pubEntry(paper)

  /* group by year */
  { WHERE attribute = "year",
          attributeValue -> "PCDATA" -> yearValue
    COLLECT Years(), Year(yearValue)
    LINK Root() -> "Years" -> Years(),
    Years() -> "Year" -> Year(yearValue),
    Year(yearValue) -> "pubEntry" -> pubEntry(paper),
    Year(yearValue) -> "year" -> yearValue
  }
}
```

Figure 6: An StruQL query [FFLS97]. Notice that enough parameters have been specified in order to produce a reconstructed answer.

operation. These operations are performed by the information system at query execution time. Analogous to WSQ [GW00], user intervention is unnecessary to realize the mixture of aspects of information-seeking. Systems supporting reconstruction via querying are best classified as templates for personalization, because users specify all aspects of information-seeking at query formulation time.

2.4 Personal Information Spaces

Yahoo! provides many tools, e.g., My Yahoo!, Yahoo! Companion, and Inside Yahoo! Search, for managing one's personal information space [MPR00]. My Yahoo! [MPR00], a manually customizable web portal, has been freely available since 1996. With My Yahoo! users may customize the content and layout of a personalized webpage. See Fig. 7 for the content template for personalization of My Yahoo!. Interaction here entails filling in pre-defined templates and is referred to as check-box personalization.

There are many such sites which provide templates for creating My sites. These types of templates are simply an abstraction of a personal webpage with infrastructure provided by a third party (e.g., Yahoo!). After exploring these tools for personal use, we conclude that their usefulness is limited by the absence of interactivity and interaction.

Nevertheless, one of the main attractions to My sites is the ease with which they permit users to manage centralized bookmarks. Personal user bookmarks provide fertile ground for collaborative filtering [THA+97] if bookmarks may be shared. The Siteseer system [RP97] mines overlap in bookmark folders to deliver personal recommendations of webpages to users. In addition to serendipitous webpage recommendation, users who interact with many computer systems and clients on a daily basis need central access to bookmarks.

Therefore, beyond providing templates for personalization, many of the My site providers, including Yahoo! and Google, implement web browser toolbars. The main goals of these toolbars are to provide ubiquitous access to bookmarks (stored in the My page), email, and web search. See Fig. 8 for examples of popular embedded toolbars for web browsers. Interacting with a toolbar template for personalization is useful, but again limited. These toolbars are static and only provide direct access to stored information. A toolbar that facilitates a dialogue between a user and browser is a vision for truly personal interaction.

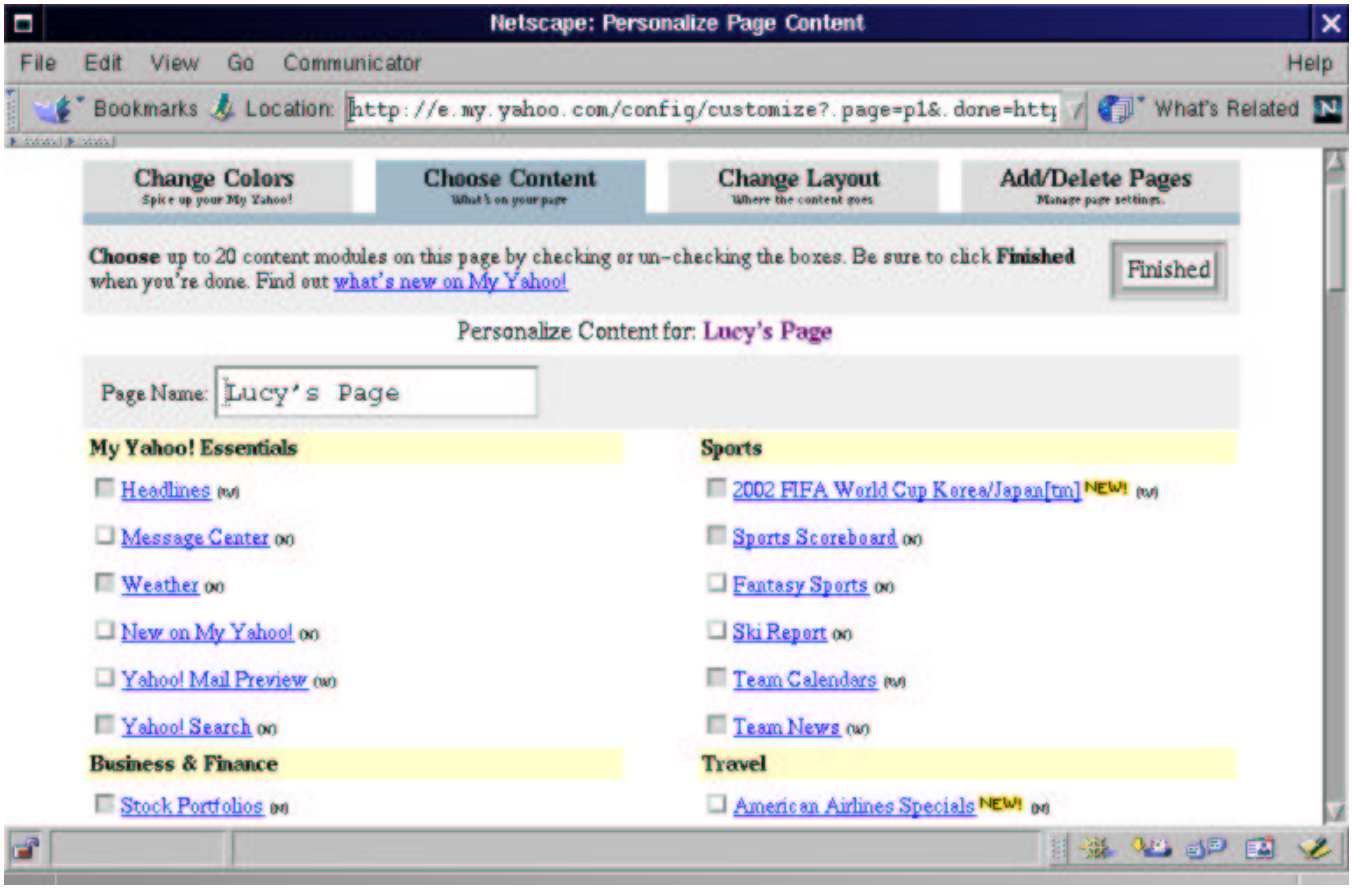

Figure 7: The content template for personalization of My Yahoo!. In this form webpage, users select desired content within categories to appear on a My Yahoo! personalized webpage. Users may similarly customize layout and color in a personalized webpage.

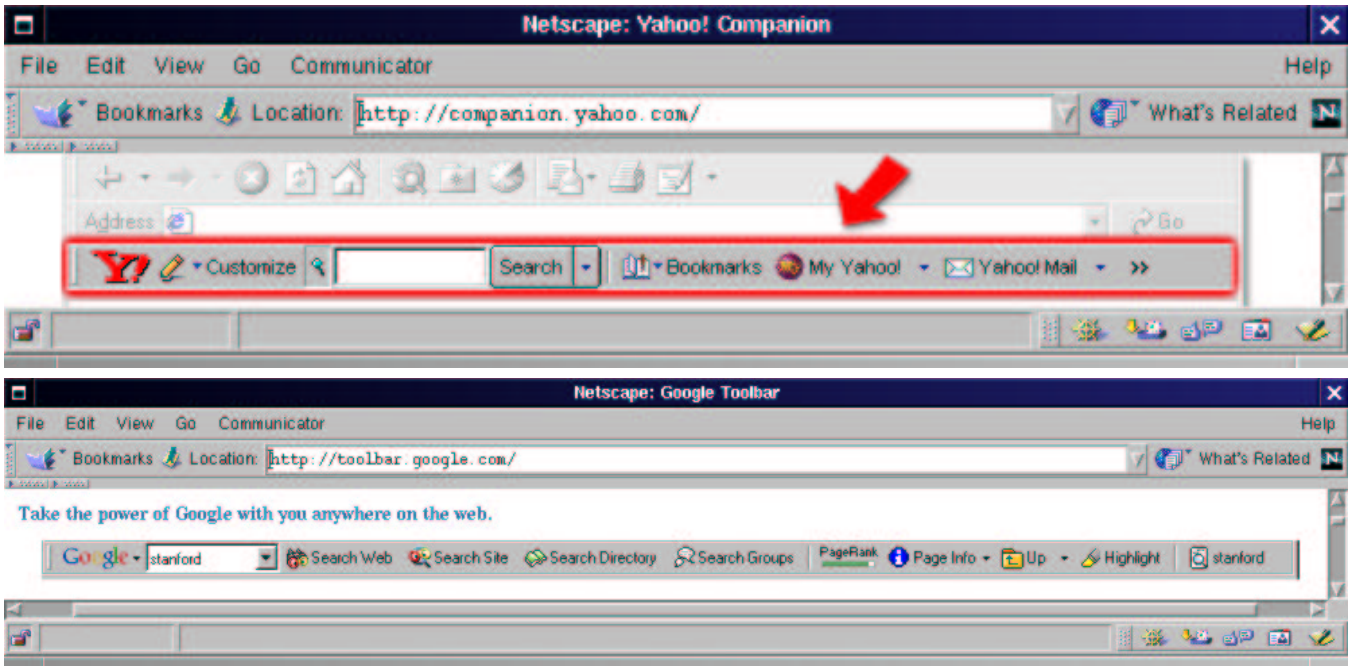

Figure 8: Static browser toolbar plugins: (top) The Yahoo! toolbar called Yahoo! Companion provides ubiquitous access to bookmarks, email, and web search. (bottom) The Google toolbar provides direct access to web search operators such as within site search, search term highlighting, and word-find.

3 Operators for Personalized Interaction

Recently, supporting the seamless integration, combination, and composition of many atomic operators by the end-user in compelling ways has become popular [Rie00a]. In this section we analyze a number of systems and projects which provide this functionality.

3.1 Search and Results Refinement

Many search systems provide users with operators to refine searches and improve search results. Typically, such operators are iteratively invoked during the course of an information-seeking session. Some operators such as relevance feedback are broad and directed toward helping users focus an initially imprecise query. Other operators, such as the 'search with results' functionality provided in many web search engines, are focused to reduce a results space. Some systems provide a hybrid of the two with a clustering operator. Users may cluster to prune results or cluster an original information space to facilitate query formulation. We expound on all three operators below.

Relevance Feedback

Relevance feedback is concerned with addressing the mental mismatch issue in query formulation. Namely, the vocabulary which a user employs to specify an information-seeking goal may not match the terms in the system representing the desired information. This should not come as a surprise, since information-seeking itself is ultimately concerned with resolving a problem for which existing knowledge is inadequate [Bel00]. This problem has been identified by many in the information systems community.

> "The major problem in interaction for naive users is therefore the large semantic gap between the user model (concepts) and the system model (words)" [Sac00].

Some systems provide static functionality supporting mnemonics to address this problem. Other researchers however contend that interaction is an ideal vehicle by which to formalize an information-seeking goal.

> "... the essence of 'interactive retrieval' lies in the constant adjustment between 'answer evaluation' and the 'command formulation' tasks to achieve user satisfaction" [Chi97].

In the mid-1960s, Rocchio developed an interactive technique for tackling this problem called 'relevance feedback' [Roc71]. Relevance feedback entails iteratively ranking search results by the user in order to correctly reformulate an information-seeking query. This helps to distinguish relevant results from irrelevant results and aids in query refinement. The process terminates when the user is satisfied that the query is ideal. Since the problem of finding the correct words for a successful search is still endemic to information systems today, much research has been conducted on interaction styles for relevance feedback. Belkin contends that information foragers would rather take a laissez-faire approach (i.e., uncontrolled term suggestion) toward query reformulation than explicit relevance feedback [Bel00]. We direct the interested reader to [CCTL01, HR01] for treatment of relevance feedback in the context of recommendation and personalization. Relevance feedback has also been employed as a technique to model user interests [MMLP97].

Web Search

While visions for future web search engines include dynamically directing users with computed links [Hea00], currently refinement operators are employed to provide aspects of personalization. Another results refinement operator, quite complementary to relevance feedback, is 'search-within.' While relevance feedback addresses a broader problem, a correctly formulated query is implicit in search-within. The operator simply reduces search to the scope of a particular information space, typically results. Search-within operators are predominantly seen in web search engines such as Google, HotBot, and Lycos. Fig. 9 (top) shows Google's interface design for searching within results. On the other hand, web taxonomies such as LookSmart and Yahoo! provide search capabilities at every step while drilling-down categories in a hierarchical fashion. The interface design of Yahoo!'s free form categorical search is shown in the bottom of Fig. 9.

Such search functionality integrates browsing and personalization; to support a truly interactive experience, however, search-within operators should be closed and applicable at any point in the information-seeking session. The search-within results operators available in Google, HotBot, and Lycos are closed. The search-within category operator, such as that seen in Yahoo! and LookSmart, is however not closed. For example, if a user initiates a search while browsing a category hierarchy in Yahoo!, interaction via hierarchical browsing is disrupted and the user is returned a flat list of results without further search or hierarchical browsing capabilities.

Lastly, integrating modes of information-seeking is seen at other levels in Yahoo!. Since Yahoo! provides a suite of specialized pages (e.g., travel pages at http://travel.yahoo.com, movie pages at http://movies.yahoo.com, and maps at http://maps.yahoo.com), designers envision personalizing searches according to the category of the request [MPR00]. For example, if one searches for 'Mission Impossible,' Inside Yahoo! Search can direct one to the appropriate page within http://movies.yahoo.com.

Clustering

Clustering elegantly reduces information overload and prevents users from sifting through many similar results. Results clustering can aid answer examination while initial clustering familiarizes a user with an information space.

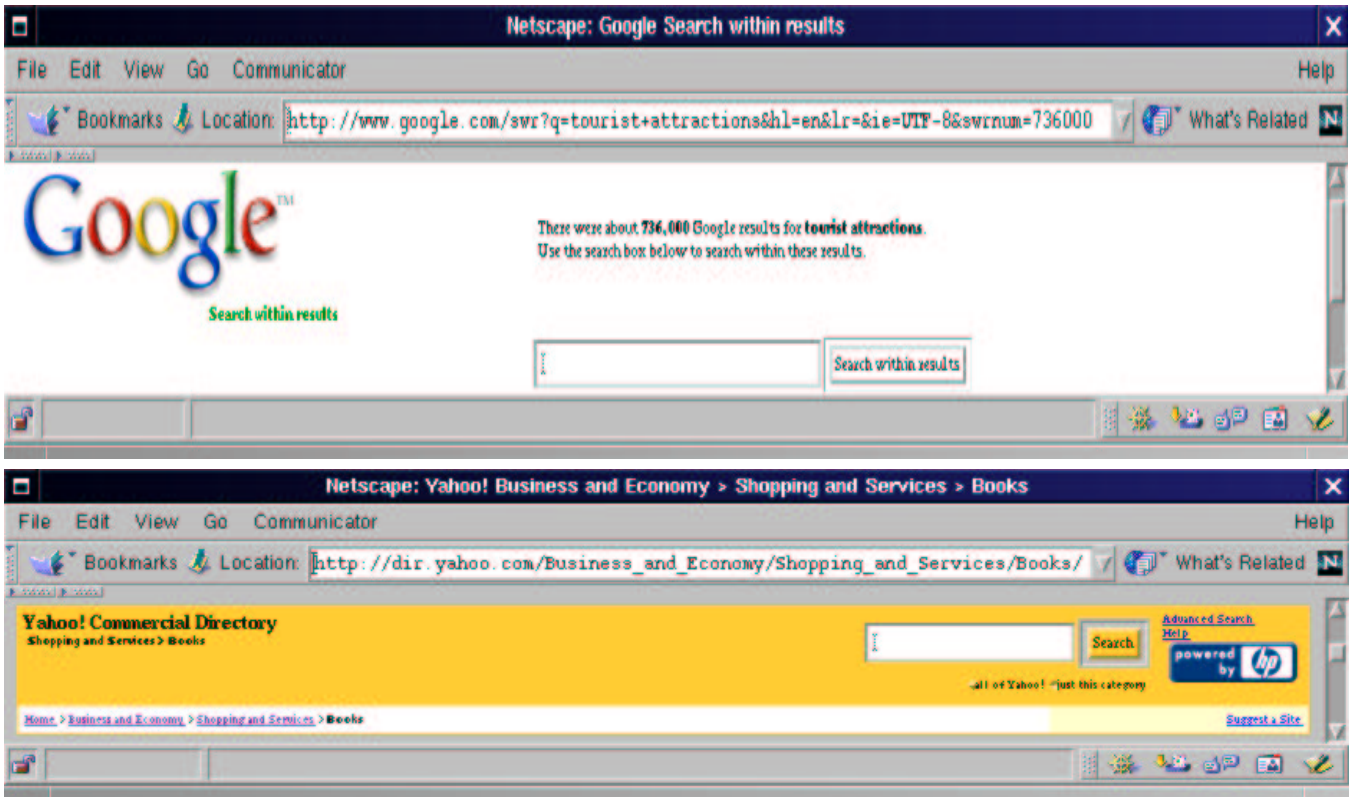

Figure 9: (top) Free form query interface for the search-within results operation in Google. (bottom) The interface to Yahoo!'s search-within category. Designs such as these provide a simple form of integrating personalization and browsing.

In many search engines, including AltaVista and Google, clustering of results is done by default so users do not see more than two pages from the same site.

Search-within functionality and clustering capabilities are just two of the many operators available in web search engines. Others include similarity and 'from links' searches. We omit discussion of these here and refer the interested reader to Search Engine Watch at http://www.seachenginewatch.com for details and comparisons. A cursory look at implementation details and structural differences in search engines is given in [Tho98].

3.2 Scatter/Gather

We present the Scatter/Gather project [CKPT92] as an example of a system that provides operators for personalized interaction. The two interactive information-seeking operations being integrated are scattering (clustering) and gathering (browsing). We begin our discussion with some motivation for the work in [CKPT92].

A large number of research projects have addressed the use of document clustering algorithms to improve information retrieval. Due to accuracy constraints however, such algorithms have poor, quadratic, run-time complexities. Therefore, these algorithms have not been widely accepted by the IR community. The Scatter/Gather project employs document clustering for different objectives. Instead of attempting to improve information retrieval via clustering, it aims to enrich browsing experiences via clustering. Clustering facilitates the formulation of an information-seeking goal by the user. Clustering in the context of Scatter/Gather is more sophisticated that the clustering for web search results described above. For instance, it entails more than collapsing webpages from the same site.

Interaction with the Scatter/Gather system is as follows. Essentially, a one-time, offline clustering of a document corpus is performed. This initial step is expensive. Afterwards, clustering is done in an online, iterative, and interactive fashion. Clustering is the scattering component of Scatter/Gather. Clusters are described to users via terms and succinct summaries. Thus, in addition to employing clustering algorithms, Scatter/Gather makes use of summarization algorithms. These algorithms essentially consider the central words of a cluster (i.e., those which appear most frequently in the group as a whole). After an initial scatter, a user selects clusters which she wishes to explore further. This step comprises the gather phase. After gathering clusters, the documents of those selected clusters are merged and re-clustered. Then, the scatter phase resumes. This interactive and iterative process continues until a user has honed in on a desired set of documents. The interleaving of scattering and gathering operations drives the information exploration process. During this process, themes of the corpus are extracted and presented to the user. One advantage of this approach is that no browsing hierarchy is hardwired *a priori*. Rather, a hierarchy is created quite naturally, on-the-fly, via clustering. Interaction with Scatter/Gather is illustrated in Fig. 10.

The interactive nature of the personalization operators available in Scatter/Gather leads us to categorize the project here. Modes of information-seeking in Scatter/Gather follow a strict, ordered sequence dictated by operation semantics. Interaction begins with a gather operation and proceeds in a scatter, gather, scatter, gather fashion. One cannot arbitrary intermix these operations. Two scatter operations in succession produce the same set of clusters. Furthermore, gathering does not make sense if it is not immediately followed by a scatter. While these two modes of information-seeking may be specified and performed over time, they are complementary and dependent on each other. Neither have semantics in isolation because no hardwired hierarchical schema is in place from the onset.

3.3 Dynamic Taxonomies

Another project closely related to Scatter/Gather is Dynamic Taxonomies [Sac00]. The motivation here is personalizing a taxonomy with set theoretic operations (e.g., union and intersection). In this context a dynamic taxonomy is a model of an information space which can be browsed and simplified by set theoretic operations. A user may drill-down a taxonomy to arrive at an interesting node. At this point in the interaction the user may continue to browse or perform a 'zoom' operation.

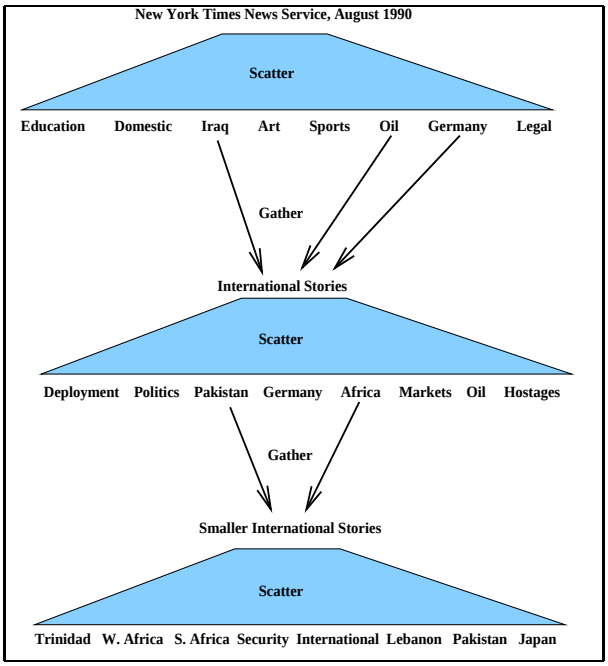

Figure 10: Interaction with Scatter/Gather (figure adapted from [CKPT92]).

The adaptation, reduction, and dynamic nature of a taxonomy via the zoom operation is computed by 'extensional inference.' The zoom can reveal relationships in the original taxonomy that even the designer may be unaware of. One caveat to this approach is that the original taxonomy must be multidimensional (i.e., an atomic data item may be classified under more than one concept).

Interaction with a dynamic taxonomy, and adaptation and reduction of it proceed as follows. Consider the multidimensional taxonomy shown on the left side of Fig. 11. The zoom operation begins with extensional inference. When a particular concept is selected (D in the case of Fig. 11), all the data atoms under this concept are computed. Performing a zoom on concept D of the taxonomy shown on the left side of Fig. 11 infers the intensional relationships illustrated with dotted arcs in the center of Fig. 11. As is shown, the zoom operation reduces the taxonomy to all the data items (i.e., concept nodes and atomic nodes) classified directly under the node that the zoom was performed on (in this case, node D). In addition, the taxonomy retains the other nodes and paths that lead to the atomic nodes classified under the zoomed node. All nodes which do not lead to those atomic data items are pruned from the taxonomy yielding a reduced taxonomy or a conceptual summary (see right side of Fig. 11). The zoom operator is closed.

With multidimensional taxonomies it is easy to see that a conjunction of the sets of ancestor nodes of the corresponding atomic objects under which a zoom is performed thins an information space. Furthermore, multidimensional taxonomies yield all set theoretic operations applicable and useful (of which intersection is the most powerful). If the information base is restricted to monodimensional taxonomies, conjunctions result in null sets yielding set union as the only applicable operator. Union operations do not simplify the taxonomy, but rather expand it and thus do not reduce information overload.

The two modes of information-seeking mixed in Dynamic Taxonomies are browsing and zooming (also referred to as taxonomic retrieval in [Sac00]). While decision points at which to browse or zoom are determined by the user, there is an ordering on such activities dictated by operation semantics. For instance, two zoom operations in succession yield the same taxonomy present prior to the second zoom operation. On the other hand, performing the second zoom operation on a different node transforms the taxonomy. The new node being zoomed upon must

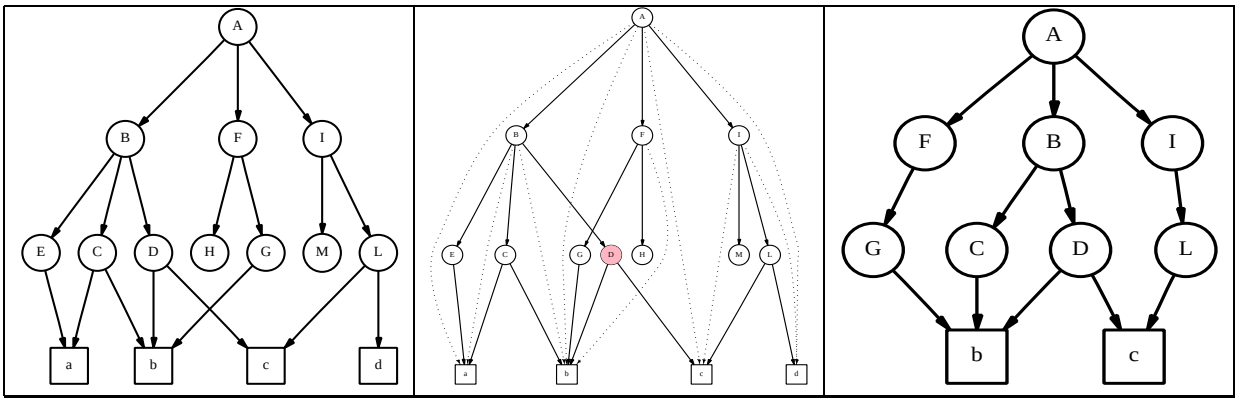

Figure 11: Illustration of the zoom operation in Dynamic Taxonomies. (left) A multidimensional dynamic taxonomy. (center) extensional inference of all concepts related to node D. (right) the reduced taxonomy after a zoom operation on concept D. Adapted from Figures 6 and 7 of [Sac00].

however be arrived at via browsing. It is clear that the zoom operation is subservient to browsing. Browsing operations, however, can be performed independently of zooming.

Since browsing and zooming are performed interactively and subject to constraints, the interaction model of Dynamic Taxonomies is similar to that of Scatter/Gather [CKPT92]. In other words, in both systems, the application of available operators for personalization is constrained. It is interesting to note that Sacco does not explicitly allude to this interaction constraint in [Sac00].

A byproduct of Sacco's approach is that dynamic taxonomies can be nicely integrated with other retrieval methods (e.g., IR and DB queries). For example, Sacco states that "extensional inference can be applied to any subset of the information base, no matter how generated, and thus guarantees a tight, symmetric coupling with other retrieval methods" [Sac00]. Such integration and associated distinctions do not alter our classification of Dynamic Taxonomies as affording operators for personalization. In conclusion, Dynamic Taxonomies is simply a set theoretic model to realize combinations of information-seeking activities. In the following systems we investigate, no constraints exist on the composition or application of available operators for personalization.

3.4 RABBIT

RABBIT is a novel information system that was well ahead of its time (circa 1984). Many of the ideas motivating RABBIT are related to a number of the papers and systems we analyze in this section. There seems to have been a gap in the literature addressing the pertinent issues (mental mismatch and combinations of interaction operators) from the time that RABBIT was published up until nearly 1995.

Essentially, RABBIT provides a unique interface to a DB. Browsing an information space is the main interaction motif. While affording compelling browsing experiences, the interface is based on the paradigm of 'retrieval by reformulation' [Wil84]. Retrieval by reformulation allows a user to incrementally specify and formalize an information-seeking goal. Specifically, a user may interleave six closed transformation operators (called critiques) with browsing. The idea is to iteratively refine a query following an operation based on how the system responds to the previous operation. A user query is implicit in the interaction with the RABBIT system. RABBIT distinguishes itself from other IR systems by exploiting partial information. Therefore RABBIT is useful to novices in a particular domain. Specifically, RABBIT assumes that a user knows more about the generic structure of the information space than RABBIT does. RABBIT however knows more about the particulars. The six critique operators available in RABBIT—require, prohibit, alternatives, describe, specialize, and predicate—are expounded in [Wil84].

The most interesting aspect of the RABBIT system is that its reformulation operators (i.e., the critiques) may be specified and invoked at arbitrary points in the interaction. Thus, in contrast to the personalization operators available in Scatter/Gather and Dynamic Taxonomies, RABBIT's operators may be applied in an unbiased fashion. An early interactive information retrieval system similar to RABBIT, which embraces the idea of integrating operators such as browsing and searching, is presented in [CT87].

The systems we present below also exhibit personalization operator independence. After a long absence from the information systems literature (over 10 years after RABBIT appeared), approaching mental model mismatches from an operation combination perspective resurfaced in [MTW95]. This work, which motivates the need for personalization operator integration, is discussed next.

3.5 DataWeb

In 1995 researchers from IBM Almaden and the Ohio State University wrote a visionary paper outlining the issues surrounding the mental mismatch problem between the designer and users of an information system [MTW95]. In addition to identifying and expounding on a legitimate cognitive problem, the authors identify approaches to solving the problem. Without using the phrase explicitly, the authors discuss aspects of mixed-initiative interaction [HM97], in the context of the interface and browsing taxonomies of Yahoo!, as chief among possible approaches.

Mixed-initiative interaction is a flexible dialogue strategy between participants where the parties can take turns at any time to change and steer the flow of interaction. It is easily observed in human conversations. For instance, the following conversation between a travel agent and a traveler illustrates a facet of mixed-initiative called unsolicited reporting [AGH99].

Conversation

1 Agent: Where would you like to travel today, Sir?
2 Traveler: New York.
3 Agent: Do you have a particular airline in mind?
4 Traveler: Not really, but I want to sit in a first class, aisle seat.
5 Agent: Very well.
6 Traveler: I also need a vegetarian meal please.
7 Agent: Sure.
(conversation continues)

The above conversation begins by the agent having the initiative (line 1), and the traveler responding to this initiative (line 2). In line 4, however, the traveler specifies seat preferences out-of-turn and hence takes the initiative. Notice that even though the traveler does not answer the agent's question about airline, the conversation progresses smoothly. Such an interaction where the two parties can mix initiative in arbitrary ways is referred to as a mixed-initiative interaction.

Mixed-initiative interaction with the envisioned DataWeb system [MTW95] is as follows. A user may initially enter a keyword query. The ensuing navigation and summarization of an answer is used to refine the initial and possibly imprecise query. Thus, querying and navigation activities are weaved to facilitate query refinement. One can browse (drill-down or roll-up) or query to attain a different hierarchy at any point while interacting with the DataWeb system. Transition from one operation to another is seamless. While in this context queries induce hierarchies, there are also an initial set of pre-existing hierarchies available as exemplars for a user to browse prior to querying. Similar functionality exists in RABBIT where a user can browse pre-cached hierarchies to exploit 'find one' [Wil84] search techniques. Thus, a user may begin an information-seeking activity in the DataWeb system with a query or browse an extant hierarchy. As can be seen, DataWeb is a highly interactive system.

The authors make it clear that a user may invoke the available information-seeking operators on demand. There is no pre-determined ordering on the operations. For these reasons and the interactive nature of the outlined system, we view DataWeb as a system affording operators for personalization. The authors partially recognize that no constraints exist on the application of their information-seeking operators.

3.6 Web Browser Command Shells

The UNIX operating system typically comes bundled with many useful, focused, and atomic software development tools such as cat, grep, and sed. While these tools have merit in isolation, a large part of the success of the UNIX operating system can be attributed to the command shell which supports the composition and communication of such powerful tools via pipes. Such composition supports user interaction in creating a compelling and truly personal experience with the system while developing software. In other words, the design of tools in UNIX has been carved up at a comfortable and personable level of granularity. Furthermore the communication mechanism, which is provided by the shell, allows end-users to become programmers on-the-fly. A similar approach to personalization is advocated in [Smi00]. The ideas presented here are motivated in [Rie00a].

Interaction with a web browser also entails invoking atomic functions (e.g., clicking on a hyperlink). Furthermore, many popular web browsers integrate access to other tools through fancy user interfaces. For example, many web browsers today provide one-click access to an email application. What web browser vendors are yet to provide is a communication mechanism to support the composition of these atomic web tools. Consider the following scenario of interaction to motivate this idea.

> Lucy launches her favorite web browser. The browser opens to her startpage—the homepage of CNN.com. The headline highlights the summer heat wave on the west coast and reminds Lucy of her trip to the Grand Canyon next week. This reminder compels Lucy to open her mail utility from within the browser, to retrieve an email sent to her last week regarding heat precautions. Upon opening the mail client, Lucy uses the find command in the browser to retrieve the message. After locating it, she opens the mail message and immediately begins clicking on the URLs provided therein. These clicks spawn page loads in her browser. After a series of mouse clicks on URLs, page loads, and invocations of the find utility of the browser (to scan the page), Lucy realizes she has found a webpage of interest. She next prints the webpage so she can take it with her on the trip. Lucy closes her browser and terminates the information-seeking session.

From the above scenario of interaction it is clear that the browser has provided easy and central access to all the tools needed to complete the information-seeking interaction (i.e., email, http requests, find, and print). Interaction with the browser is however discrete and discontinuous in the information-seeking episode. Although Lucy knows what she is looking for from the start, she has to go through a series of individual painstaking tasks. Providing a mechanism within the browser to coordinate the communication between these autonomous tasks on demand would permit a user to create truly personal interactions. It is the interleaving of these autonomous commands that is currently done by manual invocation, and which would benefit from personalization.

LAPIS: Engaging Your Browser

Many of these ideas were first introduced in [MM00] and implemented in a browser shell called LAPIS (Lightweight Architecture for Processing Information Structure). The capabilities of LAPIS include a pattern language, a scripting language, and the ability to invoke external programs. Extensions to this research include providing support in an interface for a user to create a script ‘by example’ (also called ‘automation by demonstration’) and enriching captured context such as browsing history. With the advent of the XML suite of technologies, we expect such

```
SELECT y.URL
FROM   x in Fragment, y in Fragment
WHERE  x.URL = ‘‘http://www.yahoo.com/headlines/tech/’’
       x.HREF = y
       y.CONTENT = ‘‘*Microsoft*’’;;
```

Figure 12: An AKIRA query. The semantics of this query are to i) locate and fragment the specified webpage, ii) load each webpage that the specified webpage references, and iii) search all collected fragments for the text ‘Microsoft.’ This query is a modified version of one presented in [LSCS97].

approaches to become more feasible and subsequently gain widespread acceptance. While toolbars such as LAPIS are a step in the direction toward engaging a browser in a dialogue, high levels of sophistication are not seen. We surmise that more research on representing and reasoning about user interaction in information systems will aid future systems [Mar97]. The following system employs elaborate data modeling to facilitate combinations of information-seeking activities.

3.7 AKIRA

The AKIRA project [LSCS97] at the University of Pennsylvania has a theme similar to that of WSQ. The project attempts to incorporate data on the web into a canonical DB query. Instead of simply dealing with web search results as URLs and associated frequencies, the AKIRA project models webpage content. Modeling webpage content gives users the freedom to be expressive in queries. Within-webpage modeling can also affect the granularity of answers. The model employed to capture the webpage data in AKIRA is object-oriented. The information-seeking operators which are mixed in an interactive manner are browsing, querying, and output restructuring. While we classified WSQ and web query languages as templates for personalization, we view AKIRA as providing operators for personal interaction. Information-seeking sessions with AKIRA are interactive and no constraints exist on the order in which operators may be invoked.

A user interacts with the AKIRA system as follows. After he poses a query (see Fig. 12) and receives an answer, the user may browse the resulting pages or write another query to restructure the output. Furthermore, points at which these information-seeking activities are engaged may be mixed in any order. User interaction with AKIRA is similar to that with RABBIT. As opposed to RABBIT however, querying (including output restructuring) and browsing are the only two valid information-seeking operations available in AKIRA.

3.8 Complete Answer Aggregates

Meuss and Schulz’s complete answer aggregates [MS01] are tree based data structures used to facilitate the integration of browsing, querying and reformulation in an information-seeking session. Meuss and Schulz define a complete answer aggregate as “a complete and nonredundant view on all the possible target nodes, for each of the query variables, and on all links between these candidates that contribute to some answer” [MS01]. The approach of complete answer aggregates is based on sets, relations, and tree theory.

Interaction with the system proceeds as follows. A user writes a tree structured query, whose answers map tree query nodes to DB nodes. Since the number of answers to a tree query may be exponential and thus possibly lead to information overload, a method by which to summarize and compact the answer is required. The solution adopted is factorization, which not only compacts the answer, but also arranges relevant data elements of the answer in context. Such qualification was also the primary motivation for Dynamic Taxonomies [Sac00]. A terse but expandable answer is preferred over a long, flat, and monolithic list of hits.

Aspects of information-seeking are supported by information previews (e.g., counters) to facilitate decisions on whether to construct and issue another query, drill-down, or reformulate. Reformulation here is considered as a special case of querying. Meuss and Schulz provide two closed reformulation operations: node rank by counter values and compaction by attribute values [MS01]. Surprisingly, the two useful operations on answer aggregates do not directly correspond to any of the six critique operations in RABBIT [Wil84]. The main idea is that an initial tree query will present a useful starting point for active exploration of an answer space. Meuss and Schulz contend that such exploration facilitates 'interactive knowledge discovery and hypothesis testing' [MS01]. Subsequent browsing and reformulation is employed to refine/enhance an initial, possibly under specified query.

Connections from complete answer aggregates to Dynamic Taxonomies [Sac00] and RABBIT [Wil84] is seen in that all three projects model an information resource and provide canned, closed operations (including browsing) on that resource to transform, simplify, and personalize it. The zoom operation is available in Dynamic Taxonomies. In RABBIT, available operators are reformulations. Closure preservation in complete answer aggregates fosters both an exploratory style of browsing and seamless integration with further query type activities (reformulations). This browsing style is similar to that in OLAP systems [HAC+99].

It is clear that the operators here (i.e., the specification of attribute values to collapse by and the specification of counter values to rank by) are independent of each other and need not arrive in an ordered or predetermined fashion. Furthermore, although not explicitly mentioned by the authors, we believe that another tree query may be written against a complete answer aggregate (at a different time in the interaction). Such interaction exemplifies the interleaving of information foraging activities with browsing. While the authors do not explicitly address this aspect of their approach, they do stress the exploratory nature of complete answer aggregates. At different points in time, different aggregates may be viewed via certain attributes. Since the available operations may be specified in an unbiased fashion over time, interaction with complete answer aggregates is similar to that in the RABBIT system.

3.9 BBQ and MIX

XML as a data format provides excellent opportunities for mixing operations for personal interaction, especially browsing and querying. Typically XML data elements are nested, making XML documents conducive to browsing via drill-down and roll-up metaphors. In addition, most XML query languages such as XML-QL are closed [DFF+99]. Thus, interactively blending browsing and querying of XML is quite natural.

Blending Browsing and Querying (BBQ) [MLP00, MP00] is an information system which achieves precisely this objective. There are no system semantics dictating the order in which a user may apply the two information-seeking operations. Querying in BBQ [MLP00], as opposed to more traditional XML query languages [DFF+99], may be performed by example via a drag and drop interface. Thus, querying in the BBQ system is interactive, as opposed to the one-shot style of interaction seen in other systems [FR97, GW00]. After a query is answered, the system infers a document type definition (DTD). This DTD assists the processing of subsequent queries.

We view BBQ as an information system which affords operators for personalization since querying is interactive and combined independently and at any interaction point with browsing in BBQ. The designers of BBQ do not recognize this aspect of their system. BBQ is currently absorbed by a larger project called MIX [MP02]. MIX is a mediator-based approach to integrating querying and navigation. While BBQ incorporates visualization, there also exist systems, focused solely on visualization, which afford operators for personalization as a convenient by product.

3.10 Operators for Interactive Visualization

Interactive information visualization is the main thrust of the systems we discuss in this section. These systems provide operators to bring aspects of interactivity to bear upon a visualization. Ultimately and as a by product, such operators tackle the mental mismatch issue endemic to personalization research. Therefore, we showcase these

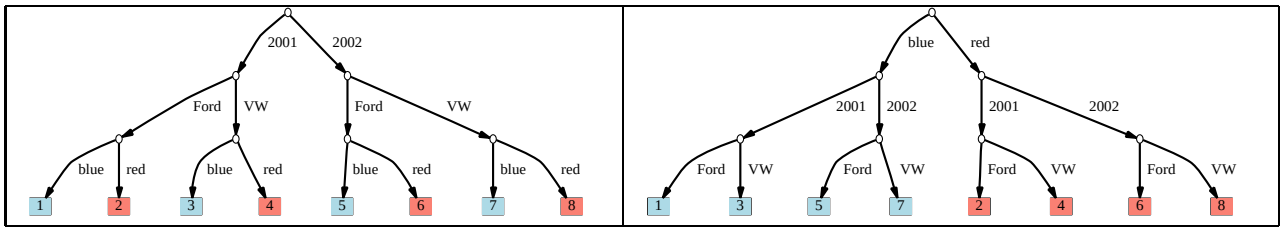

Figure 13: Illustration of a possible reconstruction operator on a UDH. The UDH description of the hierarchy on the left is modified to restructure the levels of the hierarchy (right).

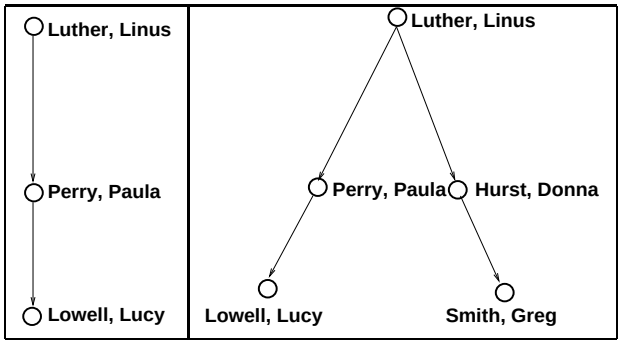

Figure 14: Adding a person to a management polyarchy (adapted from [RCCR02a]). (left) The path from 'Lowell, Lucy' to the root, 'Luther, Linus.' (right) The polyarchy resulting from adding 'Smith, Greg.' This figure illustrates how a user can incrementally add entities to a polyarchy which reveal resulting relationships. Such relationships are difficult to observe with a general overview of an extremely large hierarchy.

operations here in the context of the personalization they achieve. While there are a number of interactive information visualization systems, we focus on three which provide operators which affect user perception of an information base. Specifically, we analyze three data structures: user-defined hierarchies [WB99], polyarchies [RCCR02a, RCCR02b], and treemaps [Shn92, SW01]. A unifying theme among these systems is their ability to provide visualizations and views which expose semantic relationships in an information base.

User-Defined Hierarchies

User-defined hierarchies (UDHs) are dynamic hierarchies. Systems incorporating UDHs champion multiple visual layouts of a single hierarchy and therefore support dynamic hierarchy specification and visualization. Multiple layouts facilitate the discovery of semantic relationships in data. A number of different layout algorithms, each with support for discovering different types of properties (e.g., level of clustering) efficiently, are discussed in [WB99]. Such algorithms modify a hierarchy dynamically based on user interaction. Dynamic hierarchies are generated directly from data and not as a result of operations or transformations on an 'unpersonalized' hierarchy or representation. Modeling interaction is thus not stressed in [WB99]. Fig. 13 illustrates a possible reconstruction operator. A UDH, whose first, second, and third levels pertain to automobile year, model, and color respectively, is shown on the left side of Fig. 13. The right side of Fig. 13 might be the output of a goal-oriented reconstruction of the UDH description of Fig. 13 (left). This reconstruction reorganizes the hierarchy by making automobile color, year, and model the first, second, and third levels respectively.

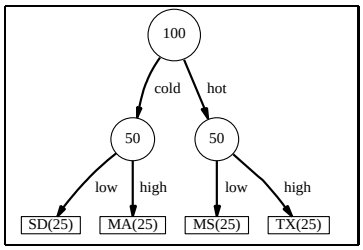

Figure 15: A tree containing data about states (adapted from [tre]).

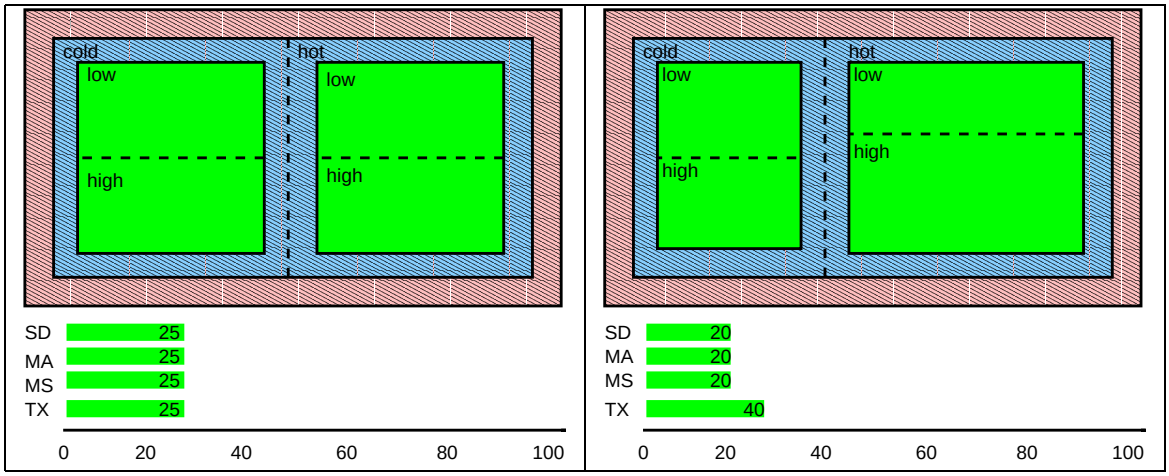

Figure 16: Illustration of the slide operator to adjust weights of attributes in treemaps (adapted from [tre]). (left) A possible treemap for the hierarchical data shown in Fig. 15. (right) Resulting treemap, which displays the recalculated state weights, from moving the sliders in (left).

Polyarchies

Polyarchies [RCCR02a, RCCR02b] deal with predetermined (static) hierarchical structures. A polyarchy groups multiple intersecting hierarchies which share at least one node into a single hierarchical structure. Again, the main focus here is on visualization. A polyarchy helps to visualize both a single hierarchy and understand the relationships between multiple entities within that single hierarchy. In addition, a user may visualize more than one hierarchy simultaneously for a clear understanding of the relationships between multiple hierarchies. To facilitate these goals, manipulations such as sliding and pivot points are provided. Fig. 14 illustrates adding a new object to a polyarchy. The left side of Fig. 14 shows one path in a management hierarchy—the path from 'Lowell, Lucy' to the root, 'Luther, Linus.' The right side of Fig. 14 shows the result of adding 'Smith, Greg' to the polyarchy—an additional path to the root. The use of this addition operator in this example illustrates the discovery of relationships between the selected entities. This approach distinguishes relationship discovery in polyarchies versus that from a general overview of an extremely large hierarchy.

Treemaps

Treemaps are yet another data structure approach to inferring relationships in an information base. Similar to polyarchies, treemaps deal with predetermined structures—trees in this case. The traditional two dimensional treemap approach is discussed in [Shn92]. The treemap3 system (see http://www.cs.umd.edu/hcil/treemap3/) extends this by allowing users to choose the aggregation order to form a tree of their choice. Layout difficulties in visualizing

treemaps as opposed to supporting multiple aggregation orders are discussed in a newer article [SW01].

Fig. 16 illustrates direct manipulation of treemap attribute weights to recompute the value of objects (e.g., the weight of states). The tree under consideration (Fig. 15) models attributes of states. The first level of the tree involves values for climate while the second level contains values for population. The number in a node represents the weight of the node which is equal to the sum of the weights of all the descendents of the node. The left of Fig. 16 displays a possible treemap and state weights for this data. Users may adjust the weight of attributes in this treemap by manipulating the dotted sliders. A user may move the sliders to explore the cumulative effect that different attribute weight values have on the objects (states, in this case). In Fig. 16, moving the sliders corresponds to adjusting the relative importance of preferences. Such an interface helps the user decide on, for example, relocation options. After manipulation the value of each object (e.g., state) is automatically recalculated as illustrated in the right side of Fig. 16. Such manipulations are critical to decision support systems as seen below. We now turn to interaction as a vehicle to analyze massive data sets.

3.11 Interactive Data Mining and Analysis

Close examination of data analysis in DBMSs, decision-support systems, and data mining packages from a user perspective reveals that analysis calls for iteration, intuition, and exploration. We have established that a query in a DBMS is a one-shot activity. Such an approach is effective when a user knows what he is seeking, but is not conducive to exploration. Thus, a user experiences frustration when using a query information-seeking strategy to search for information that the user does not know [Bel00]. This problem is endemic when DBMSs and IR systems are used as interactive systems. As discussed above, results refinement techniques such as relevance feedback are typically employed to combat this problem.

This issue is exasperated in decision-support systems and data mining applications for the following reasons. Typically batch analysis of large data sets is costly and time consuming. Often the success of algorithms which discover patterns in data is predicated on and highly sensitive to algorithm-specific parameters (e.g., support and confidence) tuned by users. A poor choice of parameters may lead to useless results. The results of the first few runs on massive data sets may be correct, but undesirable or difficult to interpret. Furthermore, knowledge that a choice of parameters is poor often unknown until results are returned. In summary, in traditional analysis systems, not only is querying, computation, and analysis one-shot, it also takes place in a 'black-box' [HAC+99]. Computation is conducted as efficiently as possible, but users have no control over it once begun.

A classic chicken and egg problem ensues. The difficulty is that users can neither precisely formulate their analysis goals nor tweak algorithmic-sensitive parameters until implicit properties of a dataset (e.g., dimensionality) are progressively revealed to them. While much research has been conducted on improving the efficiency of data analysis and mining algorithms in decision-support systems, little research has addressed improving usability and personalized interaction in such systems.

Applying techniques from human-computer interaction to data analysis is an approach. The goal of the Control project [HAC+99] is to afford users direct interaction with computation in order to refine results and control processing 'just-in-time.' Interaction tightens the data analysis process loop. Analogous to the nature of the operations of personalization of the information systems discussed in this section, Hellerstein et al. provide users of analysis tools with canned operations to facilitate interactive exploration. Rather than computations assuming a black-box model, operators for personal interaction in the Control project afford users direct insight into the ongoing analysis. Such operators trade quality and accuracy of results for direct control. A data mining user is typically willing to accept approximate and partial results in return for a handle into the computation.

The Control project supports many interactive algorithms for data analysis. The supported operations include online aggregation or drill-down online enumeration through user interface widgets to support 'eyeballing' and 'panning,' online data visualization through a technique called 'clouds,' and online data mining. Control employs random

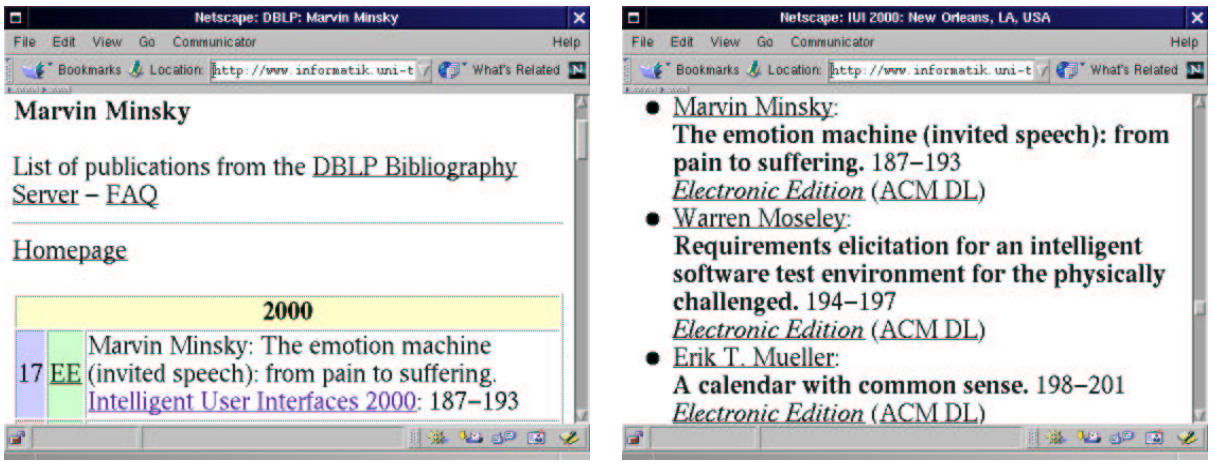

Figure 17: An association in the social network at DBLP. Jumping from a author webpage (left) to a conference webpage (right).

sampling and reordering to achieve online interactivity. In addition, Control implements ripple join algorithms to tackle online query-processing problems entailing multiple inputs.

Projects such as Control lie within the scope of data warehousing and OLAP. Data warehousing and OLAP technologies are critical to the success of decision-support systems which currently constitute a large segment of the database industry [CD97]. As OLAP technologies and resulting systems gain widespread acceptance, we expect the need for personalized interaction with them to increase. We view the design of systems such as Control as initial steps in this direction.

3.12 Social Network Navigation

While many sites on the web are organized along a hierarchical browsing motif, sites in certain domains are more effectively based on a social network navigation metaphor. A social network is a graph in which nodes represent entities (e.g., people, books, or movies) and edges represent relationships between entities (e.g., is-a-friend-of or have-co-authored-a-paper). Social networks are characterized by heterogeneous nodes and homogeneous edges. A simple example of a social network is one's network of family and friends. Examples of web sites based on a social network navigation metaphor are the Internet Movie Database at http://www.imdb.com, Barnes and Noble at http://www.bn.com, and the online computer science bibliography DBLP at http://www.informatik.uni-trier.de/∼ley/db/ (see Fig. 17).

Social networks can be induced from an existing information base for later exploration and exploitation. An early project on social network analysis induced a communication network from email logs in order to discover shared interests [SW93]. For purposes here, we are interested in operators to explore and exploit already-induced social networks, in order to discover products of interest, serendipitous collaborations, or network resources [SYV01]. Such operators enhance personal interaction and expedite the personalization process.

ReferralWeb [KSS97] is a collaborative filtering recommender system which provides users with operators for exploration and exploitation in a person-person social network. Associations between people nodes are mined from close proximity of names in web documents subject to a set of heuristics. An induced network facilitates the search for experts, communities, or documents. ReferralWeb contains operators for several types of searches including a referral chain search (e.g., a user may be interested in finding the relationship chain between herself and a colleague and thus ask, "What is my relationship to John Doe?"), an expert search (e.g., by specifying a topic and a social radius a user may ask "What friends of mine or friends of friends of mine know about tourist attractions in Italy?"), and an

expert controlled search (e.g., “List documents on the topic ‘human factors and user interface components’ close to Don Norman.”). The examples of searches given here have been adapted from those in [KSS97].

Consider how an editor of a journal may exploit a social network of authors in computer science to find an unbiased committee of reviewers for a communicated article. The editor surely does not desire individuals within close proximity of the author under review. The editor does however seek effective reviewers who must be close enough to the reviewees’ research area to be qualified. The editor may therefore apply the available operators on a social network induced from a corpus to find all individuals within three degrees of separation from the author subject to review. Systems consisting of a social network and a suite of expressive operators foster relationship discovery and are thus classified here.

4 Representing and Reasoning about Interaction

Thus far this chapter has echoed the theme that personalization is advantageously approached by studying and understanding interaction [Mar97]. In the previous two sections, the onus of personalization was on users. Templates are so over-specified that interaction is limited to filling out a form or writing a query to communicate an exact level of customization. While operators for personalization afford more freedom, interaction remains stifled by constraints on the applicability and composition of the available operators. If interaction is to guide the design of personalization systems, then beyond understanding and studying it, interaction must also be explicitly modeled and exploited. In other words, personalization should be approached from a user-centered design perspective [KNV00]. In our opinion, representing and reasoning about interaction is the holy grail of personalization. The main premise of this section and chapter echoes that of Marchetti et al. [MVPB93], namely that ‘. . . information retrieval is an inherently interactive process, and that support of users should be support of their interaction, with all of the system resources.’

4.1 Why Model Interaction?

Ultimately, models of interaction serve as a representational basis to design an interactive system. They are more expressive than templates and operators and are thus at a finer level of granularity. Care must be taken however to ensure that interaction is not modeled too tightly. In other words, over-representation and excessive modeling can lead to bulky designs. Systems which fall victim to this trap run contrary to the goals of personalization. Representations which are too general should be avoided for obvious reasons as well. This problem suggests the need for structures of interaction at a personable level of granularity. Pednault motivates this issue as:

> “The representation should be as rich and fluid as the interaction itself, but at a level of abstraction that allows the relationships among stimuli and responses to be readily observed in the data collected” [Ped00].

4.2 Information Seeking Strategies

Prior to designing an interactive system, we must first study, understand, and characterize the interactions which users desire of their information systems. Eventually designers shift from such understandings to system design representations which structure, support, and enhance interaction [BCST95]. We begin by characterizing information-seeking behavior.

Belkin et al. [BMC93] describe an information-seeking strategy (ISS) as a behavior a user engages in while interacting with a system. They have contributed a binary, four-dimensional ISS space (see Table 1) containing

	Dimensions							
	Method		Goal		Mode		Resource	
ISSs	Scan	Search	Learn	Select	Recognize	Specify	Information	Meta-information
1	√		√		√		√	
2	√		√		√			√
3	√		√			√	√	
4	√		√			√		√
5	√			√	√		√	
6	√			√	√			√
7	√			√		√	√	
8	√			√		√		√
9		√	√		√		√	
10		√	√		√			√
11		√	√			√	√	
12		√	√			√		√
13		√		√	√		√	
14		√		√	√			√
15		√		√		√	√	
16		√		√		√		√

Table 1: Four-dimensional information-seeking strategy space of Belkin et al. [BMC93].

16 (i.e., 2^4) strategies. Each dimension can be considered as a factor of information-seeking and describes a dichotomy. The ISS space factors are method of interaction (scan or search), goal of interaction (learn or select), mode of retrieval (recognize or specify), and resource (information or meta-information).

For instance, ISS15 is indicative of a highly specified search [BMC93]. A user is searching through an information base with the goal of selecting relevant items which match specification aspect input. ISS2, its complement, represents a prototypical example of a fuzzy and loose strategy. Here a user scans meta-information such as an index in order to learn to recognize where topics are situated. Depending on specific strategy instances, the information-seeking strategies (ISSs) in this space may overlap. More importantly, users typically shift between ISSs in the course of an information-seeking session, called an episode in [BMC93].

The following example illustrates such a shift. Consider a student who interacts with a university library information system to check out a reversed book for a course. If the student does not know the title of the book, he may interact with a directory indexed by course number to learn the title of the book (ISS12). After the student knows the title, he can use a search tool to find the book in the title-alphabetized reserve pages (ISS15).

Capturing and modeling such shifts is a way to support truly compelling experiences in information systems. The classification the space provides can be used to describe movement from one ISS to another. Design techniques to support combination through seamless movement from ISS to ISS are faithful to our vision of personalization through mixture of information-seeking activities as advocated throughout this chapter.

The single most striking aspect of this work is that Belkin et al. [BMC93] view an ISS as an interaction with an information system. In other words, an interaction with an IR system is a dialogue between a user and a system. Others projects divorce the two and view each ISS as a query or functional requirement of a system. Therefore, such systems do not take advantage of the interaction inherent in use. Rather than supporting interaction, such systems constrain, tolerate [BMC93], or react to it. This distinction goes to the heart of the difference between a one-way, reactive, interaction and a two-way, cooperative, dialogue.

Most designers make provisions for personalization in systems from the onset rather than supporting it through

interaction. This trend is most salient in templates, but is also seen in operators designed to implement personalization. Due to these reasons, Belkin [Bel97] feels that intelligent, agent-based approaches which circumvent the need for personal interaction with information resources are unlikely to be embraced by users.

These arguments have significant implications for the design of a system. Belkin et al. [BMC93] prefer the design of a system to explicitly support such interaction, both at the individual ISS and inter-ISS level. The work of Belkin et al. is thus truly visionary in making these novel observations and contributions.

Details of the transition from high level ISSs and interaction models to concrete implementation details need to be pinned down. Through the construction of a prototypical interface to an IR system, Belkin et al. [BMC93] explored this transition. The resulting system, called BRAQUE (BRowsing And QUEry formulation), is a two-level hypertext model of IR system DBs [MVPB93]. The system supports and validates the feasibility of the implementation of interaction as described here. In addition, and commensurate with systems presented above, BRAQUE blends query formulation and reformulation with browsing.

4.3 Structures of Interaction: Scripts, Cases, and Goal Trees

There are a number of formalisms applicable to modeling interaction. The goals, operators, methods and selection rules (GOMS) model of human-computer interaction, introduced by Card, Moran, and Newell [CMN80a, CMN80b, CMN83] in the early 1980s, is accepted as the most mature formalism. Since then, three variations of the original GOMS formulation have been developed—the keystroke-level model (KLM), natural GOMS language (NGOMSL), and cognitive-perceptual-motor GOMS (CPM-GOMS)—and are surveyed in [JK96].

Belkin et. al. [BMC93] however use a formal model called COR (conversational roles model), adept at representing dialogue structures for information-seeking. The model defines types of dialogue structures between two actors: the information provider and the information seeker. These structures capture turn taking, jumping out of dialogues, termination, and error recovery. COR models high-level dialogue structures while omitting details at the domain, task, and strategic levels. Therefore, a prescriptive interaction model in addition to the descriptive COR dialogue model is needed. Cases and scripts fill this void.

A dialogue is a specific instance of communication between a user and an information system. Dialogues may consist of many moves within a single ISS. For example, while employing an ISS, one user may decide to terminate interaction prematurely, while another may see the information-seeking goal to fruition. In either case, neither user has deviated from the particular ISS. The following is an example of a dialogue related to ISS12 of the course textbook example above.

Dialogue

1 System: May I have the course number please?
2 User: Yes. CS4604.
3 System: The title of the reserved book is “A First Course in Database Systems.”
4 User: Thanks.

Intra-strategy shifts however make dialogues a poor model of interaction for system design.

An interaction script, which is better suited, is a pattern in a two-party interaction or dialogue. Belkin et al. [BCST95] describe a script as a plan for dialogue between a user and an information system. Scripts are prototypes which model a class of concrete dialogues. Therefore, an actual dialogue is a specific instance of a script. A script is prototypical in that it implements an ISS. Scripts structure user interaction for the design of a system similar to how an interpreter structures the interaction of a program. The level of expressivity in scripts is correct for design. Scripts are written in plain English and intended to be easily understood by the layman as opposed to COR models.

1	**System:**	Here's what we can do (offers choice).
2	**User:**	Let's do this (chooses one).
3	**System:**	OK, here's how we'll do it (presents plan and means for accomplishing script).
4	**User:**	a. OK $\rightarrow$ 5 b. No, I don't like this. $\rightarrow$ 1

Figure 18: Preamble sequence for interaction scripts from [BMC93].

An alternate representation of interaction is a goal tree. A goal tree is arranged as a hierarchy of goals which organize the set of necessary moves in an ISS. Goal trees are represented in a Prolog-style notation with goals corresponding to predicates. There are goal trees associated with each ISS. Furthermore, in order to model rich interaction, some goal trees may contain sub-goals (predicates) which represent jumps to other regions of the ISS space.

In a simple system design, scripts may be stored in a dialogue manager. Upon entrance, a user and the system execute a preamble script in order to determine and retrieve the desired or appropriate script. Such introduction, which is not specific to any ISS from [BCST95] is given in Fig. 18. Combinations of scripts can also be used to achieve more expressive dialogues.

In practice, knowledge of how the dimensions of the ISS space affect each other is invaluable to reduce the number of script combinations which a system must support. Knowledge of these dimensional relationships also makes the prediction of moves between the ISSs at decision points easier. Thus, these relationships help stir user interaction and form complex scripts. For instance, Xie [Xie02] addresses how interaction intentions relate to ISSs. She identifies patterns of interaction revealing the circumstances under which certain ISSs are employed.

Another approach to interaction shifts is to mine patterns of usage in systems to anticipate which subset of the remaining 15 possible ISSs users will most desire to follow. For instance, if the leaf node in a goal tree cannot be simplified, it can be expanded and replaced with the goal tree of an alternate ISS. This leads to the broader question of where scripts come from.

Belkin et al. outline two approaches for deriving scripts in [BCST95]. The first entails 'a general characterization of information-seeking goals and a related cognitive task analysis.' The second is driven by empirical observation of interaction patterns. This approach involves inducing patterns in system use akin to web log mining. Belkin et al. use case-based reasoning (CBR) for this purpose. The end-goal is to collect, analyze, and characterize cases in the ISS space.

In such an approach, the system and cases bootstrap each other. After collecting an initial set, the ISS space induces a partition on the gathered cases. Designers then attempt to select a prototypical case from each partition which leads to a script. Due to the iterative nature of CBR, it is acceptable to start the system with a prototype. MERIT [BCST95] is an interactive IR system that embodies these ideas.

4.4 PIPE: Personalization by Partial Evaluation

PIPE [Ram00] is a research project that employs representations similar to scripts for capturing information-seeking interactions. It is aimed as a modeling methodology for information personalization. However, PIPE makes no commitments to a particular algorithm, format for information resources, type of information-seeking activities or, more basically, the nature of personalization delivered. Instead, it emphasizes the modeling of an information space in a way where descriptions of information-seeking activities can be represented as partial information. Such partial

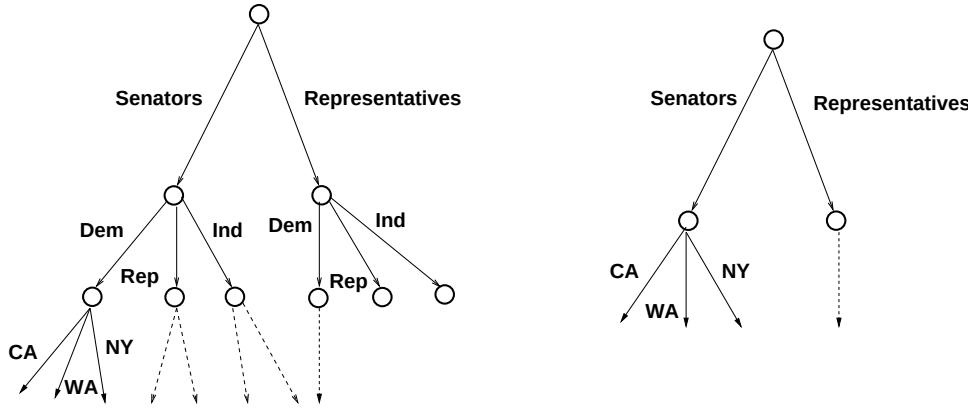

Figure 19: Personalizing a browsing hierarchy. (left) Original information resource, depicting information about members of the US Congress. The labels on edges represent choices and selections made by a navigator. (right) Personalized hierarchy with respect to the criterion 'Democrats.' Notice that not only the pages, but also their structure is customized for (further browsing by) the user.

```
int pow (int base, int exponent) {
  int product = 1;
  for (int i = 0; i < exponent; i++)
    product = product * base;
  return product;
}
```

```
int pow2 (int base) {
  return (base * base);
}
```

Figure 20: Illustration of the partial evaluation technique. A general purpose `pow`er function written in C (left) and its specialized version (with `exponent` statically set to 2) to handle squares (right). Such specializations are performed automatically by partial evaluators such as C-Mix.

information is then exploited (in the model) by *partial evaluation*, a technique popular in the programming languages community [Jon96].

It is easy to illustrate the basic concepts of PIPE by describing its application to personalizing a browsing hierarchy. Consider a congressional web site, organized in a hierarchical fashion, that provides information about US Senators, Representatives, their party and state affiliations (Fig. 19—left). Assume further that we wish to personalize the site so that a reduced or restructured hierarchy is made available for each user. The first step to modeling in PIPE involves thinking of information as being organized along a motif of interaction sequences. We can identify two such organizations—the site's layout and design that influences how a user interacts with it, and the user's mental model that indicates how best her information-seeking goals are specified and realized. In Fig. 19 (left), the designer has made a somewhat arbitrary partition, with type of politician as the root level dichotomy, the party as the second level, and state at the third. However the user might think of politicians by party first, a viewpoint that is not supported by the current site design. Site designs that are hardwired to disable some interaction sequences can be called 'unpersonalized' with respect to the user's mental model.

Example: Personalizing a Browsing Hierarchy

One typical personalization solution involves anticipating every type of interaction sequence beforehand, and implementing customized interfaces (algorithms) for all of them [Hea00]. For independent levels of classification (such as in Fig. 19—left), this usually implies creating and storing separate trees of information hierarchies. Sometimes,

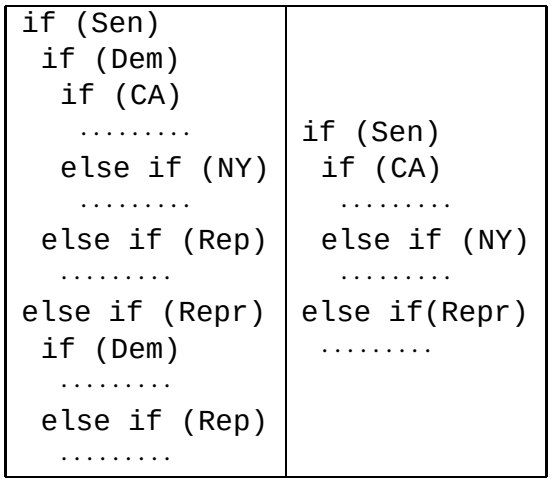

Figure 21: Using partial evaluation for personalization. (left) Programmatic input to partial evaluator, reflecting the organization of information in Fig. 19 (left). (right) Specialized program from the partial evaluator, used to create the personalized information space shown in Fig. 19 (right).

the site designer chooses an intermediate solution that places a prior constraint on the types and forms of interaction sequences supported. This is frequently implemented by directing the user to one of several predefined categories (e.g., 'to search by State, click here.'). It is clear that such solutions can involve an exponential space of possibilities and lead to correspondingly cumbersome site designs.

The approach in PIPE is to create a programmatic representation of the space of possible interaction sequences, and then to use the technique of partial evaluation to realize individual interaction sequences. PIPE models the information space as a program, partially evaluates the program with respect to (any) user input, and recreates a personalized information space from the specialized program.

The input to a partial evaluator is a program and (some) static information about its arguments. Its output is a specialized version of this program (typically in the same language), that uses the static information to 'precompile' as many operations as possible. A simple example is how the C function `pow` can be specialized to create a new function, say `pow2`, that computes the square of an integer. Consider for example, the definition of a `pow`er function shown in the left part of Fig. 20 (grossly simplified for presentation purposes). If we knew that a particular user will utilize it only for computing squares of integers, we could specialize it (for that user) to produce the `pow2` function. Thus, `pow2` is obtained automatically (not by a human programmer) from `pow` by precomputing all expressions that involve `exponent`, unfolding the for-loop, and by various other compiler transformations such as *copy propagation* and *forward substitution*. Automatic program specializers are available for C, FORTRAN, PROLOG, LISP, and several other important languages. The interested reader is referred to [Jon96] for a good introduction. While the traditional motivation for using partial evaluation is to achieve speedup and/or remove interpretation overhead [Jon96], it can also be viewed as a technique for simplifying program presentation, by removing inapplicable, unnecessary, and 'uninteresting' information (based on user criteria) from a program.

Thus we can abstract the situation in Fig. 19 (left) by the program of Fig. 21 (left) whose structure models the information resource (in this case, a hierarchy of web pages) and whose control-flow models the information-seeking activity within it (in this case, browsing through the hierarchy by making individual selections). The link labels are represented as program variables and semantic dependencies between links are captured by the mutually-exclusive `if..else` dichotomies. To personalize this site, for say, 'Democrats,' this program is partially evaluated with respect to the variable `Dem` (setting it to one and all conflicting variables such as `Rep` to zero). This produces the simplified program in the right part of Fig. 21 which can be used to recreate web pages with personalized web

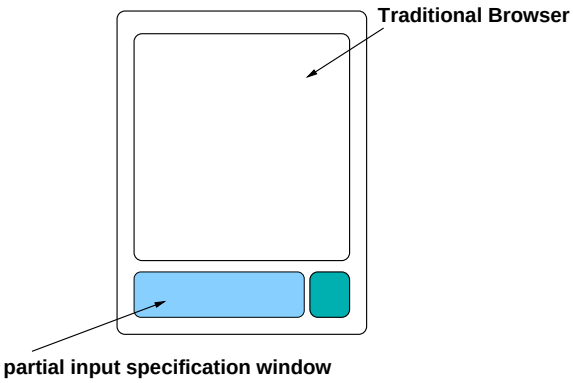

Figure 22: Sketch of a PIPE interface to a traditional browser. The interface retains the existing browsing functionality at all times. At any point in the interaction, in addition, the user has the option of supplying personalization parameters and conducting personalization (bottom two windows). Such an interface can be implemented as a toolbar option in existing systems.

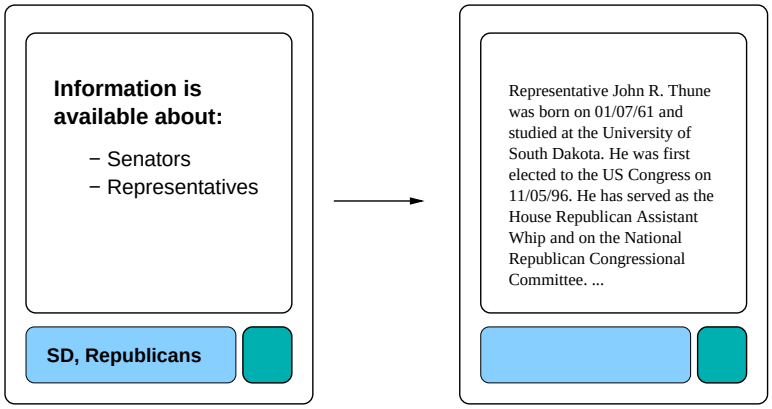

Figure 23: Example of using a PIPE interface to browse a web site about US Congressional Officials.

content (shown in Fig. 19—right). For hierarchies such as in Fig. 19, the representation afforded by PIPE (notice the nesting of conditionals in Fig. 21, left) is typically much smaller than expressing the same as a union of all possible interaction sequences.

Since the partial evaluation of a program results in another program, the PIPE personalization operator is closed. In terms of interaction, this means that any modes of information-seeking (such as browsing, in Fig. 21) originally modeled in the program are preserved. In the above example, personalizing a browsable hierarchy returns another browsable hierarchy. The closure property also means that the original information-seeking activity (browsing) and personalization can be interleaved in any order. Executing the program in the form and order in which it was modeled amounts to the system-initiated mode of browsing. 'Jumping ahead' to nested program segments by partially evaluating the program amounts to the user-directed mode of personalization. In Fig. 21 (right), the simplified program can be rendered and browsed in the traditional sense, or partially evaluated further with additional user inputs. PIPE's use of partial evaluation is thus central to realizing a mixed-initiative mode of information-seeking [RCPQ02], without explicitly hardwiring all possible interaction sequences. With this approach, it is also possible to encode miscellaneous application logic (about the interaction) and use it to drive the personalization.

An interface design for such interaction is shown in Fig. 22. Fig. 23 describes a sample scenario with the hypothetical web site of Fig. 19 (left). At the beginning of the session, the user is presented with the homepage that has

options for choosing a branch of Congress. The user prefers, instead, to provide information about party ('Republican') and state ('South Dakota'). She uses the PIPE toolbar to specify this information out-of-turn (Fig. 23, left). Partial evaluation of the PIPE model with these details actually results in also setting the Representative variable to true (since the only Republican in South Dakota is the lone Representative from that state). This results in Fig. 23 (right) where information about the Representative is displayed. This simple example shows the importance of forming a representation of interaction as a basis for personalization.

Modeling in PIPE

Modeling an information space as a program that encapsulates the underlying information-seeking activity is key to the successful application of PIPE. For browsing hierarchies, a programmatic model can be trivially built by a depth-first crawl of the site. In addition, a variety of other information spaces and corresponding information-seeking activities can be modeled in PIPE. Modeling options for representing information integration, abstracting within a web page, interacting with recommender systems, modeling clickable maps, representing computed information, and capturing syntactic and semantic constraints pertaining to browsing hierarchies are described in [Ram00, RP01]. Opportunities to curtail the cost of partial evaluation for large sites are also described in [RP01]. We will not address such modeling aspects here except to say that the effectiveness of a PIPE implementation depends on the particular modeling choices made *within* the programmatic representation (akin to [Wil84]). We cannot overemphasize this aspect—an example such as Fig. 21 can be made 'more personalized' by conducting a more sophisticated modeling of the underlying domain. For example, individual politicians' web pages at the leaves of Fig. 19 could be modeled by a deeper nesting of conditionals involving address, education, precinct, and other attributes of the individual. In other words, a single page could be further modeled as a browsable hierarchy and 'attached' (functionally invoked) at various places in the program of Fig. 21 (left). Conversely, the example in Fig. 19 can be made 'less personalized' by requiring categorical information along with user input. For instance, replacing `if (Dem)` in Fig. 21 with `if (Party==Dem)` implies that the specification of the type of input (namely that 'Democrat' refers to the 'name of the party') is required in order for the statement to be partially evaluated. Personalization systems built with PIPE can thus be distinguished by what they model and the forms of customization enabled by applying partial evaluation to such a modeling.

5 Making It Work: Systems Support and Enabling Technologies

We now briefly mention some systems support technologies to bring personalization solutions into mainstream adoption and use.

5.1 Data Modeling

Researchers have identified data modeling as critical to the degree of personalization delivered [Chi97, RS97]. For personalization purposes, data modeling often involves databases techniques for the web [ABS00, FLM98]. We focus here on content modeling and information integration techniques, such as web crawling and wrapping.

Web Wrappers and Information Integration

The main motivation for wrappers is bridging the gap between the abundance of data on the web and applications which have no direct access to the web [ABS00, HGMC+97]. WSQ [GW00] is an example of a system which can benefit from such modeling. The type of information extraction techniques employed are dependent on the type of personalization intended.

In template-based systems, a query typically drives the modeling process [AK97, KMA+98]. Manually designing a web wrapper and subsequently maintaining it is a painstaking process due to dependence on the source format. Therefore, research has been conducted on automatically generating wrappers. Such programs exploit structural cues in data. Ashish and Knoblock take a regular expression, grammar-based, approach to wrapper generation [AK97]. An alternate approach [SA99] is to exploit intermediate mappings between system-defined formats and standard formats, such as XML and DOM. The project culminated in the world wide web wrapper factory (W4F) toolkit [SA99].

All of these projects are focused on answering queries and thus approach wrapper generation from a within-page modeling standpoint. Others take a broader approach and model site structure or mediate inter-site differences [KMA+98]. Central to this approach is the flow of information within a site and across sites. In other words, information is integrated through data flow. The output of the first source is fed into the second source as input and so on. Such an approach can be contrasted to formalisms for information integration that use shared schemas and mediated queries [GMPQ+97, GBMS99, KLSS95].

These approaches suffer from a pitfall endemic to all wrappers, whether automatically generated or not. Crawling or wrapping a third party web site is error prone due to page irregularities, extensive use of stylish page formatting, and an abundance of semistructured data [ABS00]. While many wrapper and crawling packages are freely available on the web, such tools are difficult to use out-of-the-box and typically require a level of manual customization for a particular site. It is often useful to conduct a preliminary inspection of page design and site layout before implementing such systems. In addition, a variety of semantic issues exist for effective information integration which are currently handled with heuristics.

Several distinct solutions to this problem have emerged. One idea is to focus modeling to specific document structures. Rus and Subramanian concentrate on capturing and modeling tabular structures and thus employ document segmentation and structural detection algorithms [RS97]. XTRACT [GGR+00], a system similar to [AK97], uses grammars and AI techniques to infer DTDs (document type definitions) for XML data. The endurance of such approaches are tested by richer standards for document types such as XSchema [Fal01]. It is widely believed that XSchema may render DTDs obsolete. An alternate approach to webpage modeling, which also employs AI techniques, is wrapper induction [KWD97]. Systems such as [GGR+00, KWD97] scale well with regard to frequently changing sites due to the exploitation of machine learning techniques. Yet another solution uses program compaction techniques to infer schemas in semistructured data [NAM97, NAM98].

5.2 Requirements Gathering

Techniques discussed in this section address requirements gathering for personalization systems. This problem can be approached from two distinct angles. The first involves empirical and explicit requirements analysis techniques such as scenario-based design. An alternate approach involves web log mining to implicitly capture requirements. Ultimately these techniques are directed toward closing the gap between the goals of a system designer and the task model of a user [MAB00].

Scenario-Based Methods

The techniques presented here are especially important with regard to representing and reasoning about interaction. Carroll and Rosson make an explicit science out of scenario-based design and claims analysis in [CR92] where they describe the 'task-artifact' methodology. The end-goal of this research, which lies at the intersection of human-computer interaction (HCI) and software engineering, is to develop an action science approach to HCI.

The first step in the methodology is to collect scenarios. Scenarios are narrative accounts of users performing tasks and can be generated empirically or analytically. Carroll and Rosson develop a classification of scenarios, or typology, which aids in analytical and empirical approaches to scenario collection.

The next step in the methodology is claims analysis. A claim is 'a specific psychological consequence of a system feature' [CR92]. While a scenario provides a narrative account, a claim provides a causal account. Claims analysis attempts to explain scenarios and must proceed in parallel with scenario generation. Scenarios and claims thus developed can be utilized by CBR as applied to script-directed information systems. For instance, they can be processed to yield cases and identify prototypical scripts.

This work also has important connections to requirements engineering in PIPE [RP01]. In particular, scenario-based design and claims analysis can be used to generate interaction sequences in a domain which lacks precise, explicit, and clear semantics. In existing systems, the task-artifact cycle can be used to characterize interaction sequences. This is particularly interesting in sites based on the metaphor of social network navigation. In such non-traditional information spaces, scenario-based design can be employed to either unroll unbounded interaction sequences to a manageable level or define personalization. The resulting scenarios (i.e., interaction sequences) would be invaluable to finding an appropriate programmatic design representation of interaction.

Rosson has researched the integration of task and object models [Ros99] in software design. To facilitate this goal, she proposes using object-oriented analysis and design of scenarios. Scenarios are helpful in identifying an initial set of software objects. Claims analysis of the scenarios identifies constraints and opportunities. This work has ties to PIPE as well. In PIPE, a user's personalized experience, analogous to the task model, closely resembles the system's programmatic model of interaction, analogous to the object model.

These connections between scenario-based design and PIPE are explored in [RRC01]. In addition, the authors discuss how explanation-based generalization (EBG) can be used to explain scenarios to provide a starting point for a personalization system. EBG is a machine learning technique which has strong ties to partial evaluation [Jon96]. Proof trees in EBG used in explaining scenarios resemble the goal trees used by Belkin et al. [BCST95].

An alternative approach to requirements gathering is metaphorical design [Mad94]. It is well accepted by now that metaphors provide intuitive ways to think about interaction with information systems (e.g., the desktop). For instance, Wexelblat and Maes [WM99] explore the use of *footprints* as a navigation design metaphor.

Web Mining

The web is becoming fertile ground for what O'Leary calls 'AI Renaissance' [O'L97]. The use of collaborative filtering in recommender systems was one of the first attempts at conducting personalization. Collaborative filtering is difficult, since the the majority of web users are privacy conscious and dislike providing explicit feedback. When applying these techniques care must be taken to ensure that privacy is not compromised.

Web log mining is an alternate approach to capturing user interest and has been referred to as 'observational personalization' [MAB00]. Web log mining is implicit, unobtrusive, and entails chartering the footprints left by visitors. One can analyze web logs to mine navigational patterns.

For instance, IndexFinder [PE00] mines patterns to guide a non-destructive and transformation approach to web site adaptation. Non-destructive adaptations are those that add structure, pages to sites, or links to pages, but do not destroy structure or otherwise remove information from a site. IndexFinder identifies co-occurring page visits and recommends candidate index pages to the web master. Thus, IndexFinder is a semi-automatic approach.

Web navigation patterns are sought to evaluate web site usability as well [Spi00]. The focus here is on avoiding costly and error prone formative usability evaluations. The miner is looking for sequences of frequently visited pages and routes connecting pages frequently accessed together. Two popular web log mining software systems are MiDas and Web Utilization Miner (WUM) [Spi00]. Another project which has user modeling goals is discussed in [MCS00]. Here, weblog mining helps form associations which are used in a collaborative filtering style to aid a recommendation engine.

While web mining is data driven and therefore heuristic at best, it is inexpensive and can be applied more frequently than its manual counterparts discussed above. The projects described here show that mining access logs is a feasible approach to gathering requirements for personalization. This approach however suffers from problems

of coordination and ethics. Therefore, social and operational issues need to be addressed to make such techniques practical and more appealing.

5.3 Transformation Algorithms

XML has matured from simple text markup for data interchange to a mature technology with a rich suite of associated tools. The eXtensible Stylesheet Language Transformation (XSLT) performs transformations from XML to XML and various other formats including plain text, HTML, and VoiceXML. The transformation capabilities of XSLT can be used to implement partial evaluation, among other operations, and to create a robust and easily maintainable personalization application.

Specifically, if an XML file models interaction with a web site, then an XSLT stylesheet representing partial input (i.e., a user request) can be matched against XML tag labels to simplify and personalize interaction. Since XSLT can transform XML into multiple output formats, transforming the model of interaction into a browsable web site (i.e., HTML) is also easy. XSLT thus unifies the processes required to conduct personalization into a single mature and well-accepted technology. Details regarding the use of these new and emerging W3C standards are presented in [CDA00].

5.4 Delivery Mechanisms and Intermediaries

Intermediaries, which are 'programs or agents that meaningfully transform information as it flows for one computer to another,' [MB00] are critical to the success of personalization applications on the web. Examples of intermediaries are portals, proxies, and transcoders. IBM's WBI [MB00] provides a programming model for intermediaries akin to PIPE's contribution of a programming model for personalized interaction.

6 Niche Domains

6.1 Adaptive Hypermedia

In the past 15 years, hypermedia has been extended to support personalization capabilities (e.g., the adaptive web [BM02]). Adaptive hypermedia lies at the intersection of hypermedia and user modeling [Bru01]. Hypermedia services such as educational and online-help systems have been most impacted by personalization research.

Links in adaptive hypermedia systems are dynamic, leading to different destinations for different users. The techniques employed include direct guidance, adaptive link sorting, hiding, annotation, generation, and map adaptation. In addition to navigational adaptations, such applications modify the aesthetics of presentation to direct a user. We refer the interested reader to [Bru96] and its sequel [Bru01] for a comprehensive survey of methods and techniques of adaptive hypermedia systems and to [BBH99] for a succinct introduction. Examples of browsing-oriented adaptive hypermedia systems are Syskill & Webert [PMB96] and WebWatcher [JFM97].

6.2 Mobile Environments

Mobile arenas, which host the fastest growing segment of web users, are plagued with low bandwidth networks, thin clients, and information appliances [Ber00]. Furthermore, ubiquity is enriched and propelled by wireless portals, avatars [AR02, LFW01], and information kiosks [MBG$^+$01]. As these devices become commonplace, transcoding the information they present will not only become a necessity, but also vital to their widespread use and success [BBE$^+$02, Pan01]. Therefore, the use of personalization technology here extends past aesthetics. It is a requirement and no longer expendable. To introduce this application domain, we present the following two representative projects.

Proteus

Proteus [ADW01] is a mobile personalization system developed at the University of Washington. The goal of the system is to both transcode and personalize web content based on mobile devices. To achieve the first goal, the designers segment webpages into screens using a probabilistic model. To achieve the goal of personalization, the designers collect training data from desktop computer usage to build user models.

Proteus supports both destructive and constructive within-page adaptations and implements three transformation operators—elide-content, swap-siblings, and add-shortcut. Creating a new webpage or adding new links between existing pages is not supported. The system can be contrasted to other adaptive systems such as IndexFinder [PE00]. While IndexFinder provides only non-destructive adaptation targeted by topic to all site visitors, Proteus is destructive as well and provides customization per individual. In addition, Proteus's user models are richer than those that result from web log mining, which are essentially limited to navigational usage patterns.

W^{3}IQ

W^{3}IQ [JPK98] aims to provide asynchronous mobile access to the web. The designers explore collaborative information retrieval techniques to minimize resource use and information overload. W^{3}IQ employs intermediaries, such as proxy filters and cache servers, to facilitate disconnected browsing. In addition, it supports three types of transaction-like operations, which save state and are thus tolerant to disconnection.

6.3 Voice Interfaces and Multimodal Interaction

Speech and dialogue-based systems, which afford mixed-initiative interaction [HM97], provide ripe domains for personalization. Zadrozny et al. state that natural language is 'a compelling enabling technology for personalization' [ZBC$^+$00] and that mixed initiative dialogue is a form of personalization. Voice applications (e.g., voice portals[1]) and associated tools (e.g., VoiceXML) have collectively spawned the voice web [SB02]. Furthermore, this domain demonstrates how researchers in qualitatively different areas can work unconsciously on the same problem. We survey such connections below.

Sisl (several interfaces, single logic) [BCD$^+$00], a primarily speech-based system, aims to minimize dialogue constraints to provide extensive flexibility to users. Thus, the motivations of Sisl are commensurate with those of PIPE. Furthermore, the authors of Sisl recognize the idea of engaging a system in a two-way dialogue as a means to provide personalization.

Sisl however takes a broader approach to personalization and supports multiple interfaces, error recovery, reversion, partial input, and partial orderings on specification aspects in dialogue. In contrast to PIPE, Sisl takes an event-based approach. The designers model application logic by event handling (reactive) mechanisms. The specification aspects of PIPE are called events in Sisl.

Sisl makes a distinction between partial orderings and partial information. Partial information is incomplete in that all specification aspects required to complete a dialogue or information-seeking activity are communicated incrementally. Partial orders, on the other hand, permit aspects to arrive in different orders. Furthermore, Sisl makes a distinction between *out-of-turn* aspects and *unsolicited* aspects. PIPE traditionally clubs these two together, because its support mechanism, partial evaluation, handles both uniformly.

Both Sisl and PIPE rely on the assumption that a representation of default order execution exists (e.g., a script). This representation involves anticipation in both approaches. PIPE eagerly (partially) evaluates that representation with respect to specification aspects, which may arrive in any order, to implement partial orderings in dialogues. Sisl, on the other hand, lazily evaluates aspects. In other words, when Sisl receives an aspect out-of-turn, which violates its representation, it logs that aspect in a queue. The system retrieves that aspect when the default order of

[1] Examples are Tellme (http://www.tellme.com) and BeVocal (http://www.bevocal.com).

execution solicits it later. At that time, the event is enabled and added to the activated set. The designers refer to this process, which handles unsolicited events and thus minimizes anticipation, as lookahead.

A closer connection between PIPE and speech-based systems is made in [RCPQ02] where the form interpretation algorithm of VoiceXML [MBD+01] is shown to be a partial evaluator in disguise.

7 Conclusions

We have presented an overview of personalization systems according to the interaction they afford. The reader will have gathered that our personal preferences fall in the third tier of systems which explicitly represent and reason about interaction. As personalization systems become prevalent, the need to engage the user in compelling interactions will become more important.

Several factors lead us to be optimistic about the future of personalization as an academic discipline. For instance, the widespread use of physical computing devices, location-aware systems, and embedded Internet appliances means that personalization will transcend current delivery mechanisms. Such domains pose interesting problems that will continue to challenge our assumptions about personalization. Users create context in physical situations that can be stored and retrieved for use in electronic access paradigms. Thinking about how information access works in such multimodal settings will lead to a theory of human-information interaction, as espoused in [TMK+02].

We would like to end this chapter on a cautionary note. The contents of this chapter show that relevant work is becoming increasingly fragmented across many venues and sub-disciplines. Pertinent research is now published among the artificial intelligence, database systems, knowledge management, information retrieval, world wide web, user interfaces, and human-computer interaction conferences. We advocate periodic reconciliation and a back-to-basics approach to unify methodologies, when possible. For instance, in [Ram02] we have highlighted the role of *partial* information in achieving various forms of personalization. Such models and modeling methodologies will help systematize the study of personalized interaction.

References

[ABS00] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.

[ADW01] C. R. Anderson, P. Domingos, and D. S. Weld. Personalizing Web Sites for Mobile Users. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, pages 565–575, Hong Kong, May 2001. ACM Press.

[AGH99] J. F. Allen, C. I. Guinn, and E. Horvitz. Mixed-Initiative Interaction. *IEEE Intelligent Systems*, Vol. 14(5):pages 14–23, September–October 1999.

[AK97] N. Ashish and C. Knoblock. Wrapper Generation for Semi-Structured Internet Sources. *SIGMOD Record*, Vol. 26(4):pages 8–15, December 1997.

[AR02] E. André and T. Rist. From Adaptive Hypertext to Personalized Web Companions. *Communications of the ACM*, Vol. 45(5):pages 43–46, May 2002.

[BBE+02] D. Billsus, C. A. Brunk, C. Evans, B. Gladish, and M. Pazzani. Adaptive Interfaces for Ubiquitous Web Access. *Communications of the ACM*, Vol. 45(5):pages 34–38, May 2002.

[BBH99] P. De Bra, P. Brusilovsky, and G.-J. Houben. Adaptive Hypermedia: From Systems to Framework. *ACM Computing Surveys*, Vol. 31(4es), December 1999. Article No. 12.

[BC92] N. J. Belkin and W. B. Croft. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, Vol. 35(12):pages 29–38, December 1992.

[BC99] R. Bodner and M. Chignell. Dynamic Hypertext: Querying and Linking. *ACM Computing Surveys*, Vol. 31(4es), December 1999. Article No. 15.

[BCD+00] T. Ball, C. Colby, P. Danielsen, L. J. Jagadeesan, R. Jagadeesan, K. Läufer, P. Mataga, and K. Rehor. Sisl: Several Interfaces, Single Logic. *International Journal of Speech Technology*, Vol. 3(2):pages 93–108, June 2000.

[BCST95] N. J. Belkin, C. Cool, A Stein, and U. Thiel. Cases, Scripts, and Information Seeking Strategies: On the Design of Interactive Information Retrieval Systems. *Expert Systems with Applications*, Vol. 9(3):pages 379–395, 1995.

[Bel97] N. J. Belkin. An Overview of Results from Rutgers' Investigations of Interactive Information Retrieval. In *Proceedings of the Thirty-Fourth Annual Clinic on Library Applications of Data Processing: Visualizing Subject Access for Twenty-First Century Information Resources*, 1997.

[Bel00] N. J. Belkin. Helping People Find What They Don't Know. *Communications of the ACM*, Vol. 43(8):pages 58–61, August 2000.

[Ber00] E. Bergman, editor. *Information Appliances and Beyond*. The Morgan Kaufmann Series on Interactive Technologies. Morgan Kaufmann, 2000.

[BM02] P. Brusilovsky and M. T. Maybury. From Adaptive Hypermedia to the Adaptive Web. *Communications of the ACM*, Vol. 45(5):pages 31–33, May 2002.

[BMC93] N. J. Belkin, P. G. Marchetti, and C. Cool. BRAQUE: Design of an Interface to Support User Interaction in Information Retrieval. *Information Processing and Management*, Vol. 29(3):pages 325–344, May–June 1993.

[Bor86] C. Borgman. The User's Mental Model of an Information Retrieval System: An Experiment on a Prototype On-line Catalogue. *International Journal of Man-Machine Studies*, Vol. 24(1):pages 47–64, 1986.

[Bru96] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 6(2–3):pages 87–129, 1996.

[Bru01] P. Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 11(1–2):pages 87–110, 2001.

[Bus45] V. Bush. As We May Think. *The Atlantic Monthly*, Vol. 176(1):pages 101–108, July 1945.

[CCTL01] W. B. Croft, S. Cronen-Townsend, and V. Larvrenko. Relevance Feedback and Personalization: A Language Modeling Perspective. In *Proceedings of the Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, pages 49–54, Dublin, Ireland, June 2001. Dublin City University.

[CD97] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technologies. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, pages 65–74, Tucson, AZ, May 1997. ACM Press.

[CDA00] I. Cingil, A. Dogac, and A. Azgin. A Broader Approach to Personalization. *Communications of the ACM*, Vol. 43(8):pages 136–141, August 2000.

[Cha99] S. S. Chawathe. Describing and Manipulating XML Data. *IEEE Data Engineering Bulletin*, Vol. 22(3):pages 3–9, September 1999.

[Chi97] Y. Chiaramella. Browsing and Querying: Two Complementary Approaches for Multimedia Information Retrieval. In N. Fuhr, G. Dittrich, and K. Tochtermann, editors, *Proceedings of Hypertext, Information Retrieval, Multimedia (HIM'97)*, pages 9–26, Dortmund, Germany, September–October 1997.

[CKPT92] D. Cutting, D. Karger, J. Pedersen, and J. W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, Copenhagen, Denmark, June 1992. ACM Press.

[CMN80a] S. K. Card, T. P. Moran, and A. Newell. Computer Text-Editing: An Information-Processing Analysis of a Routine Cognitive Skill. *Cognitive Psychology*, Vol. 12:pages 32–74, 1980.

[CMN80b] S. K. Card, T. P. Moran, and A. Newell. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM*, Vol. 23(7):pages 396–410, July 1980.

[CMN83] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum, Hillsdale, N.J., 1983.

[CR92] J. M. Carroll and M. B. Rosson. Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario. *ACM Transactions on Information Systems*, Vol. 10(2):pages 181–212, April 1992.

[CT87] W. B. Croft and R. H. Thompson. I^3R: A New Approach to the Design of Document Retrieval Systems. *Journal of the American Society for Information Science*, Vol. 38(6):pages 389–404, 1987.

[DFF+99] A. Deutsch, M. Fernández, D. Florescu, A. Levy, D. Maier, and D. Suciu. Querying XML Data. *IEEE Data Engineering Bulletin*, Vol. 22(3):pages 10–18, 1999.

[Fal01] D. C. Fallside, ed. XML Schema W3C Recommendation Document. Technical report, World Wide Web Consortium, May 2001.

[FFK+98] M. Fernández, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 414–425, Seattle, WA, June 1998. ACM Press.

[FFLS97] M. Fernández, D. Florescu, A. Levy, and D. Suciu. A Query Language for a Web-Site Management System. *SIGMOD Record*, Vol. 26(3):pages 4–11, September 1997.

[FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3):pages 59–74, September 1998.

[FR97] N. Fuhr and T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Transactions on Information Systems*, Vol. 15(1):pages 32–66, January 1997.

[GBMS99] C. H. Goh, S. Bressan, S. Madnick, and M. Siegel. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems*, Vol. 17(3):pages 270–293, July 1999.

[GGR+00] M. N. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A System for Extracting Document Type Descriptors from XML Documents. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pages 165–176, Dallas, TX, May 2000. ACM Press.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, Vol. 8(2):pages 117–132, March–April 1997.

[GW00] R. Goldman and J. Widom. WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pages 285–296, Dallas, TX, May 2000. ACM Press.

[HAC+99] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive Data Analysis: The Control Project. *IEEE Computer*, Vol. 32(8):pages 51–59, August 1999.

[Hea00] M. A. Hearst. Next Generation Web Search: Setting Our Sites. *IEEE Data Engineering Bulletin*, Vol. 23(3):pages 38–48, September 2000.

[HGMC+97] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha. Extracting Semistructured Information from the Web. In *Proceedings of the NSF–ESPRIT Workshop on Management of Semistructured Data*, pages 18–25, Tucson, AZ, May 1997.

[HM97] S. Haller and S. McRoy. Computational Models for Mixed Initiative Interaction (Papers from the 1997 AAAI Spring Symposium). Technical Report SS-97-04, AAAI/MIT Press, 1997.

[HR01] D. Hiemstra and S. Robertson. Relevance Feedback for Best Match Term Weighting Algorithms in Information Retrieval. In A. F. Smeaton and J. Callan, editors, *Proceedings of the Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, pages 37–42, Dublin, Ireland, June 2001. Dublin City University.

[JFM97] T. Joachims, D. Freitag, and T. M. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 770–777, Nagoya, Aichi, Japan, August 1997. Morgan Kaufmann.

[JK96] B. E. John and D. E. Kieras. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Transactions on Computer-Human Interaction*, Vol. 3(4):pages 320–351, December 1996.

[Jon96] N. D. Jones. An Introduction to Partial Evaluation. *ACM Computing Surveys*, Vol. 28(3):pages 480–503, September 1996.

[JPK98] A. Joshi, C. Punyapu, and P. Karnam. Personalization and Asynchronicity to Support Mobile Web Access. In *Proceedings of the CIKM'98 Workshop on Web Information and Data Management*, Bethesda, MD, November 1998. ACM Press.

[KLSS95] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In C. Knoblock and A. Levy, editors, *Information Gathering from Heterogeneous, Distributed Environments*, pages 85–91. AAAI Press, Stanford, CA, 1995. AAAI Spring Symposium Series Technical Report.

[KMA+98] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. G. Philpot, and S. Tejada. Modeling Web Sources for Information Integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 211–218, Madison, WI, July 1998. AAAI Press.

[KNV00] J. Kramer, S. Noronha, and J. Vergo. A User-Centered Design Approach to Personalization. *Communications of the ACM*, Vol. 43(8):pages 45–48, August 2000.

[KSS97] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, Vol. 40(3):pages 63–65, March 1997.

[KWD97] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 729–737, Nagoya, Aichi, Japan, August 1997. Morgan Kaufmann.

[LFW01] H. Lieberman, C. Fry, and L. Weitzman. Exploring the Web with Reconnaissance Agents. *Communications of the ACM*, Vol. 44(8):pages 69–75, August 2001.

[LSCS97] Z. Lacroix, A. Sahuguet, R. Chandrasekar, and B. Srinivas. A Novel Approach to Querying the Web: Integrating Retrieval and Browsing. In D. W. Embley and R. C. Goldstein, editors, *Proceedings of the ER'97 Workshop on Conceptual Modeling of Multimedia Information Seeking*, Los Angeles, CA, November 1997. Springer.

[MAB00] M. D. Mulvenna, S. S. Anand, and A. G. Büchner. Personalization on the Net using Web Mining. *Communications of the ACM*, Vol. 43(8):pages 122–125, August 2000.

[Mad94] K. H. Madsen. A Guide to Metaphorical Design. *Communications of the ACM*, Vol. 37(12):pages 57–62, December 1994.

[Mar97] G. Marchionni. *Information Seeking in Electronic Environments*. Cambridge Series on Human-Computer Interaction. Cambridge University Press, 1997.

[MB00] P. Maglio and R. Barrett. Intermediaries Personalize Information Streams. *Communications of the ACM*, Vol. 43(8):pages 96–101, August 2000.

[MBD+01] S. McGlashan, D. Burnett, P. Danielsen, J. Ferrans, A. Hunt, G. Karam, D. Ladd, B. Lucas, B. Porter, K. Rehor, and S. Tryphonas. Voice eXtensible Markup Language: VoiceXML. Technical report, VoiceXML Forum, October 2001. Version 2.0.

[MBG+01] F. Mintzer, G. W. Braudaway, F. P. Giordano, J. C. Lee, Karen A. Magerlein, S. D'Auria, A. Ribak, G. Shapir, F. Schiattarella, J. Tolva, and A. Zelenkov. Populating the Hermitage Museum's New Web Site. *Communications of the ACM*, Vol. 44(8):pages 52–60, August 2001.

[MCS00] B. Mobashier, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. *Communications of the ACM*, Vol. 43(8):pages 142–151, August 2000.

[MLP00] K. D. Munroe, B. Ludäscher, and Y. Papakonstantinou. Blending Browsing and Querying of XML in a Lazy Mediator System. In C. Zaniolo, P. C. Lockemann, M. H. Scholl, and T. Grust, editors, *Proceedings of Seventh International Conference on Extending Database Technology (EDBT'00)*, Konstanz, Germany, March 2000. Springer. In Exhibitions section.

[MM00] R. C. Miller and B. A. Myers. Integrating a Command Shell Into a Web Browser. In *Proceedings of the 2000 USENIX Annual Technical Conference*, pages 158–166, San Diego, CA, June 2000. The USENIX Association.

[MMLP97] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. Palakal. A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation. *ACM Transactions on Information Systems*, Vol. 15(4):pages 368–399, October 1997.

[MP00] K. Munroe and Y. Papakonstantinou. BBQ: A Visual Interface for Browsing and Querying XML. In H. Arisawa and T. Catarci, editors, *Proceedings of Fifth Working Conference on Visual Database Systems (VDB5)*, Fukuoka, Japan, May 2000. Kluwer Academic Publishers.

[MP02] P. Mukhopadhay and Y. Papakonstantinou. Mixing Querying and Navigation in MIX. In *Proceedings of the Eighteenth International Conference on Data Engineering (ICDE'02)*, San Jose, CA, February–March 2002.

[MPR00] U. Manber, A. Patel, and J. Robinson. Experience with Personalization on Yahoo! *Communications of the ACM*, Vol. 43(8):pages 35–39, August 2000.

[MS01] H. Meuss and K. U. Schulz. Complete Answer Aggregates for Treelike Databases: A Novel Approach to Combine Querying and Navigation. *ACM Transactions on Information Systems*, Vol. 19(2):pages 161–215, April 2001.

[MTW95] R. J. Miller, O. G. Tsatalos, and J. H. Williams. Integrating Hierarchical Navigation and Querying: A User Customizable Solution. In I. F. Cruz, J. Marks, and K. Wittenburg, editors, *Proceedings of ACM Workshop on Effective Abstractions in Multimedia Layout, Presentation, and Interaction*, San Francisco, CA, November 1995. ACM Press.

[MVPB93] P. G. Marchetti, S. Vazzana, R. Panero, and N. J. Belkin. BRAQUE (abstract): An Interface to Support Browsing and Interactive Query Formulation in Information Retrieval Systems. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 358, Pittsburg, PA, June–July 1993. ACM Press.

[NAM97] S. Nestorov, S. Abiteboul, and R. Motwani. Inferring Structure in Semistructured Data. *SIGMOD Record*, Vol. 26(4):pages 39–43, December 1997.

[NAM98] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting Schema From Semistructured Data. In L. M. Haas and A. Tiwary, editors, *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 295–306, Seattle, WA, June 1998. ACM Press.

[O'L97] D. O'Leary. The Internet, Intranets, and the AI Renaissance. *IEEE Computer*, Vol. 30(1):pages 71–78, January 1997.

[Pan01] C. Pancake. The Ubiquitous Beauty of User-Aware Software. *Communications of the ACM*, Vol. 44(3):page 130, March 2001.

[PE00] M. Perkowitz and O. Etzioni. Adaptive Web Sites. *Communications of the ACM*, Vol. 43(8):pages 152–158, August 2000.

[Ped00] E. P. D. Pednault. Representation is Everything. *Communications of the ACM*, Vol. 43(8):pages 80–83, August 2000.

[PMB96] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill and Webert: Identifying Interesting Web Sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 54–61, Portland, OR, August 1996. AAAI Press.

[Ram00] N. Ramakrishnan. PIPE: Web Personalization by Partial Evaluation. *IEEE Internet Computing*, Vol. 42(9):pages 21–31, November–December 2000.

[Ram02] N. Ramakrishnan. The Traits of the Personable. Technical Report cs.AI/0205022, Computing Research Repository (CoRR), May 2002.

[RCCR02a] G. G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Animated Visualization of Multiple Intersecting Hierarchies. *Information Visualization*, Vol. 1:pages 50–65, April 2002.

[RCCR02b] G. G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Polyarchy Visualization: Visualizing Multiple Intersecting Hierarchies. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'02)*, pages 423–430, Minneapolis, MN, April 2002. ACM Press.

[RCPQ02] N. Ramakrishnan, R. Capra, and M. A. Pérez-Quiñones. Mixed-Initiative Interaction = Mixed Computation. In P. Thiemann, editor, *Proceedings of the ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'02)*, pages 119–130, Portland, OR, January 2002. ACM Press. Also appears in *ACM SIGPLAN Notices*, Vol. 37, No. 3, March 2002.

[Rie00a] D. Riecken. Personal End-User Tools. *Communications of the ACM*, Vol. 43(8):pages 89–91, August 2000.

[Rie00b] D. Riecken. Personalized Views of Personalization. *Communications of the ACM*, Vol. 43(8):pages 27–28, August 2000.

[Roc71] J. J. Rocchio. Relevance Feedback in Information Retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Ros99] M. B. Rosson. Integrating Development of Task and Object Models. *Communications of the ACM*, Vol. 42(1):pages 49–56, January 1999.

[RP97] J. Rucker and M. J. Polano. Siteseer: Personalized Navigation for the Web. *Communications of the ACM*, Vol. 40(3):pages 73–75, March 1997.

[RP01] N. Ramakrishnan and S. Perugini. The Partial Evaluation Approach to Information Personalization. Technical Report cs.IR/0108003, Computing Research Repository (CoRR), August 2001.

[RRC01] N. Ramakrishnan, M. B. Rosson, and J. M. Carroll. Explaining Scenarios for Information Personalization. Technical Report cs.HC/0111007, Computing Research Repository (CoRR), November 2001.

[RS97] D. Rus and D. Subramanian. Customizing Information Capture and Access. *ACM Transactions on Information Systems*, Vol. 15(1):pages 67–101, January 1997.

[RV97] P. Resnick and H. R. Varian. Recommender Systems. *Communications of the ACM*, Vol. 40(3):pages 56–58, March 1997.

[SA99] A. Sahuguet and F. Azavant. Looking at the Web through XML Glasses. In *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIs'99)*, pages 148–159, Edinburgh, Scotland, September 1999. IEEE Computer Society.

[Sac00] G. M. Sacco. Dynamic Taxonomies: A Model for Large Information Bases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12(3):pages 468–479, May–June 2000.

[SB02] S. Srinivasan and E. Brown. Is Speech Recognition Becoming Mainstream? *IEEE Computer*, Vol. 35(4):pages 38–41, April 2002.

[Shn92] B. Shneiderman. Tree Visualization with Tree-Maps: 2-D Space-Filling Approach. *ACM Transactions on Graphics*, Vol. 11(1):pages 92–99, January 1992.

[Smi00] D. C. Smith. Building Personal Tools by Programming. *Communications of the ACM*, Vol. 43(8):pages 92–95, August 2000.

[Spi00] M. Spiliopoulou. Web Usage Mining for Web Site Evaluation. *Communications of the ACM*, Vol. 43(8):pages 127–134, August 2000.

[Suc87] L. A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication*. Cambridge University Press, 1987.

[SW93] M. F. Schwartz and D. C. M. Wood. Discovering Shared Interests Using Graph Analysis. *Communications of the ACM*, Vol. 36(8):pages 78–89, August 1993.

[SW01] B. Shneiderman and M. Wattenberg. Ordered Treemap Layouts. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'01)*, pages 73–78, San Diego, CA, October 2001. IEEE Computer Society.

[SYV01] M. P. Singh, B. Yu, and M. Venkatraman. Community-Based Service Location. *Communications of the ACM*, Vol. 44(4):pages 49–54, April 2001.

[THA+97] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: A System for Sharing Recommendations. *Communications of the ACM*, Vol. 40(3):pages 59–62, March 1997.

[Tho98] B. Thomas. URL Diving. *IEEE Internet Computing*, Vol. 2(3):pages 92–93, May–June 1998.

[TMK+02] J. J. Thomas, D. R. McGee, O. A. Kuchar, J. W. Graybeal, D. L. McQuerry, and P. L. Novak. What is your Relationship with your Information Space? In *Proceedings of Computer Graphics International*, Bradford, UK, July 2002.

[tre] Interactive Transactions of OR/MS (ITORMS): Visualization and Optimization. URL: http://catt.bus.okstate.edu/jones98/treemaps.htm.

[WB99] R. M. Wilson and R. D. Bergeron. Dynamic Hierarchy Specification and Visualization. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'99)*, pages 65–72, San Francisco, CA, October 1999. IEEE Computer Society.

[Wid99] J. Widom. Data Management for XML: Research Directions. *IEEE Data Engineering Bulletin*, Vol. 22(3):pages 44–52, September 1999.

[Wil84] M. D. Williams. What makes RABBIT run? *International Journal of Man-Machine Studies*, Vol. 21:pages 333–352, 1984.

[WM99] A. Wexelblat and P. Maes. Footprints: History-Rich Tools for Information Foraging. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI'99)*, pages 270–277, Pittsburgh, PA, May 1999. ACM Press.

[Xie02] H. Xie. Patterns between Interactive Intentions and Information-Seeking Strategies. *Information Processing and Management*, Vol. 38(1):pages 55–77, January–February 2002.

[ZBC+00] W. Zadrozny, M. Budzikowski, J. Chai, N. Kambhatla, S. Levesque, and N. Nicolov. Natural Language Dialogue for Personalized Interaction. *Communications of the ACM*, Vol. 43(8):pages 116–120, August 2000.