Personalization by Partial Evaluation

Naren Ramakrishnan and Mary Beth Rosson
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061
Email: {naren,rosson}@cs.vt.edu

The central contribution of this paper is to model personalization by the programmatic notion of partial evaluation. Partial evaluation is a technique used to automatically specialize programs, given incomplete information about their input. The methodology presented here models a collection of information resources as a program (which abstracts the underlying schema of organization and flow of information), partially evaluates the program with respect to user input, and recreates a personalized site from the specialized program. This enables a customizable methodology called PIPE that supports the automatic specialization of resources, *without enumerating the interaction sequences beforehand*. Issues relating to the scalability of PIPE, information integration, sessionizing scenarios, and case studies are presented.

1 Introduction

The rapid growth of the World Wide Web (WWW) and the concomitant increase in online content has greatly exacerbated information overload. Personalization constitutes the mechanisms and technologies required to customize information access to the end-user. Elements of the Internet personalization landscape include search engines [45, 75], handcrafted content indices (e.g. `yahoo.com`) [49], and recommender systems [58, 71]. Even a cursory survey of the articles in the recent *Communications of the ACM* issue on the topic [51, 63] reveals that the scope of personalization extends to many different forms of information content and delivery [15, 23, 36, 48], not just web pages. The underlying algorithms and techniques, in turn, range from simple keyword matching of consumer profiles, collaborative filtering for E-commerce [4, 41, 74], to more sophisticated forms of data mining, such as clustering web server logs [50, 72]. Various dichotomies have been proposed that classify personalization research according to the philosophies of the underlying domains, the proposers, and their parent communities. Thus, categorizations such as Content-based vs. Collaborative [2, 3, 62], 'Customization vs. Transformation' [56], non-destructive vs. destructive, 'public transportation vs. hot-rods' [68] have become widely accepted. In addition, combinations of various approaches are also used [8, 21, 32, 55, 73]. Advances in data management [25] and web site management [24] have fueled many of these developments. Social aspects of personalization are either exploited or uncovered by research projects [38, 43, 66]. Theories from human-computer interaction [59] have provided insights into the design of interfaces [9, 37, 42] and 'information foraging' [33].

Limitations of Current Approaches

While impressive levels of functionality are being achieved in personalization, the lack of a sophisticated conceptual model for personalization constitutes a serious bottleneck from a designer's perspective (and indirectly, to users). Compare and contrast this with the availability of powerful models for other aspects of online information access, such as querying [25], mining [1, 61], and navigation [67, 69].

Example 1: We motivate our ideas with a simple example involving personalization using link labels. (In this paper, we specifically concentrate on web site personalization. Extension of our proposed methodologies to other domains of personalization are addressed toward the end of the project description.) Consider a congressional web site, organized in a hierarchical fashion, that provides information about US Senators, Representatives, their party, precinct, and state affiliations (Fig. 1). Notice that the site designer has made a somewhat arbitrary partition, with type of politician as the root level dichotomy, the party as the second level, and so on. Some users think of politicians primarily by party (and hence will wish to personalize w.r.t. party criteria), while others might be interested in politicians who serve their geographical area. For example, the circled region in Fig. 1 indicates the result of personalization for the input 'Democrat Senators.' A designer wishing to support this form of personalization will likely provide a 'combination query' interface that involves party and the type of politician. To cover all potential scenarios, however, the designer will have to anticipate every type of partial input (query) beforehand, and implement customization interfaces (algorithms) for all of them. In the absence of an adequate programming model, many assumptions and simplifications are embodied in the interfaces to personalization systems, e.g., a prior on the types and forms of information input that are supported. Such modeling assumptions can be either due to necessity ("this site is organized in this manner and I can't help it"), and/or lack of understanding/appreciation of users' needs ("I think this is the best way to provide the interface to my customers"). In either case, it causes serious cognitive and representational frustrations for the user, since the modes of interaction are hardwired (e.g. 'This facility will work only if you specify

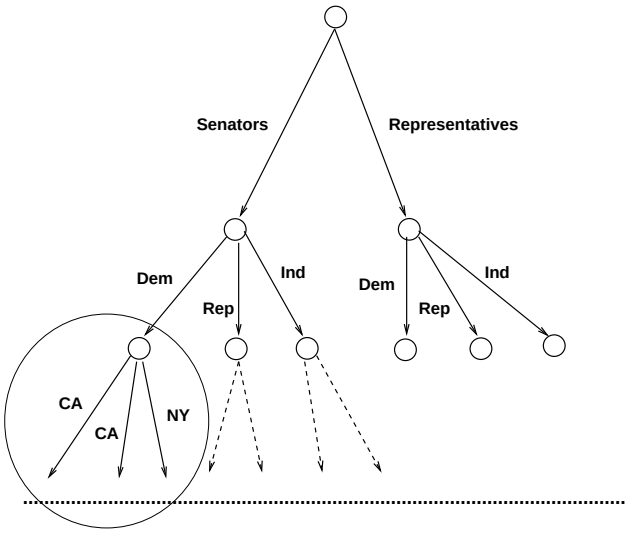

Figure 1: Hypothetical web site organization modeling information about US Senators and Representatives. Only the first few levels of the site are shown; the lower levels can be visualized as modeling individual precincts of politicians, the bills they sponsor, constituencies, their addresses, interests, education etc. The labels on edges represent choices and selections made by a navigator.

both the party and the type of politician'). While the example here involves only content-based techniques, the limitations apply (and are often compounded) in other personalization scenarios.

What is Required?

A customizable methodology for personalization should support the automatic specialization of web resources based on user input, **without enumerating the interaction sequences beforehand**. In other words, the designer should not have to anticipate and predefine mechanisms for every conceivable partial input. For example, in Fig. 1, if the user knows only the state of the politician, but not details for level(s) higher in the hierarchy, personalization should be able to simplify the lower levels of the hierarchy, thus being responsive to **differing levels of input by visitors**. The methodology should be able to exploit structure at varying levels of information access [68]. In addition, a skilled systems engineer should be able to use the methodology in conjunction with a specific collection of web sites to design a personalization system. The facility should scale, both with the size of the input sources (web sites being modeled) and with the complexity of the personalization scenario [64]. This requires that the design of the system be integrated with the task model(s) underlying the assumed interaction scenario. A functional metaphor for personalization, in addition, should allow the designer to view the system as a composition of individual subsystems, supporting information integration [29, 39]. Providing incrementality (when web sites change content and structure dynamically) without requiring the designer to restructure, supporting heterogeneity (to enable compelling scenarios), and enabling domain-specific techniques for making recommendations/selections will be a critical factor in the success of such a framework.

The PIPE Methodology

The central idea here is to model personalization by the programmatic notion of *partial evaluation*. Partial evaluation is a technique used to specialize programs, given incomplete information about their input [34, 46]. The methodology presented here — PIPE — models a collection of web sites

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

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

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

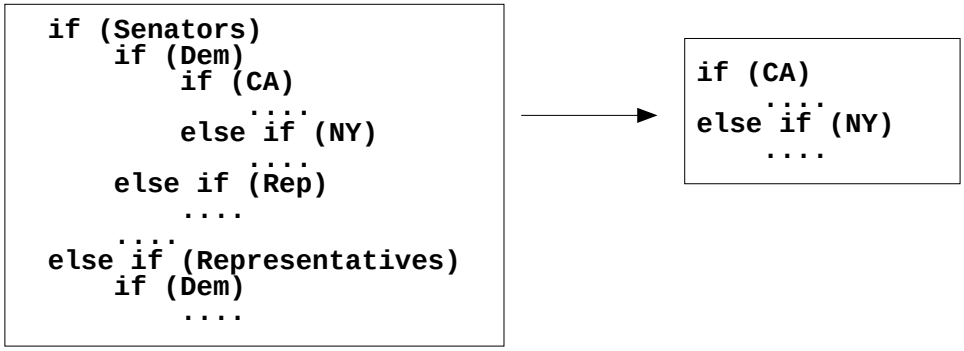

Figure 3: (left) Example input to a PIPE implementation, reflecting the organization of information at a web site devoted to political officials, and (right) output from partial evaluator, personalizing w.r.t. 'Democrat Senators.' The specialized program can be used to recreate web pages, where not only the pages but their structure is customized for the user.

as a program (which abstracts the underlying schema of organization and flow of information), partially evaluates the program with respect to user input, and recreates a personalized web site from the specialized program.

How does Partial Evaluation Work? 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 'pre-compile' 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 `power` function shown in the left part of Fig. 2 (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 [34] for a good introduction. While the traditional motivation for using partial evaluation is to achieve speedup and/or remove interpretation overhead [34], 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.

We illustrate the basic idea with the site in Fig. 1. The labels for the links in Fig. 1 can be obtained

from the text anchoring the hyperlinks via '`<a href>`s' in the web pages, or from XML tags [22]. A web crawler employing a depth-first search can then be used to obtain a program, that models the links such that the interpretation of the program refers to the organization of information in the web sources. For example, the data in Fig. 1 produces the program in the left part of Fig. 3, where the link labels are represented as program variables. The mutually-exclusive dichotomies of links (more on this issue later) at individual nodes (e.g. 'A Democrat is not a Republican') are modeled by `else if`s. Notice that while the program only models the organization of the web site, other textual information at each of the internal nodes can be stored/indexed alongside by associating augmented data structures with the program variables. Furthermore, at the 'leaves' (i.e., the innermost sections of the program), variable assignments corresponding to the individual URLs of the Senator/Representative home pages can be stored. To personalize this site, for say, 'Democrat Senators,' the above program is partially evaluated with respect to the variables `Senators` and `Dem` (setting them to 1). This produces the simplified program in the right part of Fig. 3 which can be used to recreate web pages, thus yielding personalized web content (shown by the circular region in Fig. 1). In addition, this approach allows personalization even when variable values for certain level(s) are available, in the absence of values for level(s) higher in the hierarchy. For example, if the user desires information about a NY politician (but is unsure whether he/she is a Senator or Representative or a Democrat/Republican/Independent), then a partially evaluated output (with respect to `NY` and setting other variables such as `CA` to zero) will simplify the lower levels of the hierarchy, thus being responsive to varying levels of user input.

Early Results: The PIPE methodology has been utilized in three diverse domains: (i) personalizing information about politics (specifically, the VoteSmart web site `http://www.vote-smart.org`), (ii) personalizing recommendations about mathematical software on the web (`http://gams.nist.gov` and `http://www.netlib.org`), and (iii) personalizing the Blacksburg Electronic Village (BEV) [20] (`http://www.bev.net`) for tourists. The reader is referred to the publication [60] for details of the approach and to the web site `http://pipe.cs.vt.edu` for demos of these and other applications. Evaluations against benchmark problem sets as well as user feedback are very encouraging. In addition, PIPE can accommodate individual recommender systems by modeling their invocation as program statements that abstract the control flow of the recommendation algorithm. It supports information integration by cascading individual web resources as functions pipelined in succession from a main subroutine/program.

Preliminary Analysis of PIPE

PIPE models and exploits structural cues to provide flexibility in personalization (in a manner similar to that proposed by Rus and Subramanian [68]). Currently, the above three implementations assume that (i) the link labels represent choices made by a navigator, (ii) it is possible to ascertain the values for such labels (program variables) from user input, and (iii) the web sites are predominantly hierarchical with the leaves containing most of the information sought. Such assumptions are often validated by the existence of ontologies (e.g. in the mathematical software and the BEV studies) that help guide the personalization process. In our implementations, we realized early that personalization will only be as effective as the ease with which the link labels could be determined or supplied by the user. For example, partially evaluating a beverages site with respect to 'Coke' will not yield any benefit if the link label says 'Coca-Cola.' Thus, information mediation (for handling synonymy and polysemy) becomes important in this setting. Currently, PIPE implementations also do not personalize within a web page; formatting and support for document structures, DTDs etc. are hence not addressed.

What PIPE is NOT

- **Query Optimization by Partial Evaluation.** While query processing/optimization can benefit from partial evaluation [16, 46], this is not the underlying model here. PIPE provides the notion of a 'single program' whose control models the flow of information. In a database setting (or in languages like WebSQL, WebOQL, and Florid [25]), the information is dispersed across multiple tables, and it is the responsibility of the programmer to write queries that do personalization. To achieve similar results by query processing, extra care has to be taken either while modeling the data in database tables and/or formulating the SQL queries. In addition, the results from a personalization run might require new links to be created that are not originally present in the schema which, in turn, have to be reformatted into a web structure. There exists no automatic way to design the SQL query from the user input, and intimate knowledge of the database schema (such as the levels at which the labels appear, via path expressions) is required. In contrast, partial evaluation can specialize segments of the program even if they fall on disjoint subtrees, does not require knowledge of the levels at which the labels appear, and can automatically provide an integrated view of the specialized content.

- **A General Purpose System.** PIPE is a designer's tool for a particular domain and is not intended to substitute for search engines. It provides a programmatic framework to design personalization systems by a skilled systems engineer and requires a careful identification of 'starting points,' a tight methodology for information integration [27], and an adequate understanding of the problem domain to be effective. We do not posit that PIPE constitutes a silver bullet; many web resources are not hierarchical, many use links for purposes other than narrowing on an information source, and structure exists at other granularities that should be modeled. However, PIPE does provide a systematic conceptual methodology to study the design, implementation, and evaluation of personalization systems. For example, a facility to personalize travel information for visitors to the Grand Canyon can be constructed by modeling various web resources (pertaining to this domain), and using PIPE to customize content for visitors, based on travel preferences and attributes. The creation of a composite program and integration need only be performed once (*offline*); every user session will result in a customized partial evaluation (*online*). The underlying philosophy is that the investment in making the offline design would provide great advantages for personalization scenarios.

Reader's Guide

Section 2 proposes how this methodology can be extended to larger web sites by incorporating domain specific constraints and mining semistructured data. We will utilize the three prototype implementations of PIPE to illustrate the basic ideas. Section 3 outlines a fundamental approach to 'design for personalization,' discusses information integration in personalization systems, and draws parallels to scenario-based design. Broadening aspects of the PIPE methodology, including application to non-traditional domains are presented in Section 4. Some application domains that will benefit from this idea are introduced in Section 5.

2 Scaling up PIPE

2.1 Research Issues

The basic methodology presented thus far is content-based, works at the level of web site organization, and does not mine/model the textual content or formatting within individual pages, beyond

associating them with the appropriate nodes in a tree/graph-based framework. Moreover, for the purposes of personalization, the schema obtained by simple recursive depth-first crawling can get unwieldly for even medium sized sites. Our preliminary studies [57] have shown that even sites that are unequivocally judged by the average web surfer to be a highly structured site are best abstracted by semistructured models. The causes include lack of apriori schema, cross-references (that violate a tree taxonomy), duplication of common page sets, and constantly evolving standards (e.g., in bioinformatics resources). Such semistructure should be factored into the representation by compressions; e.g. factoring commonalities in tree building. The scalability factor depends on the ability to extract succinct schema from semistructured sources and page layouts (a problem studied in considerable depth [28, 31, 53]) and to incorporate domain-specific restrictions into the partial evaluation process. **Extra care should be taken however, in this process, since termination of the partial evaluator shouldn't be compromised.** Research in binding-time analysis, program transformation, and compiler generation show that one of the main issues in automatic specialization is ensuring that the evaluator doesn't run into an endless loop (perhaps by introducing infinitely many auxiliary code segments) and that the semantic interpretation of the program is not affected (e.g. in terms of the effects of code segments). Web sites that possess recursive links (for navigational and cross-indexing purposes) are especially difficult.

2.2 Solution Methodology

Considerable attention has been devoted to extracting schema from semistructured data. The primary motivation has been to obtain approximate representations in situations where there is no prior schema, to drive query processing, and/or for use in data mining. Our goal here is to obtain a good starting point for personalization. Research on *typing* semistructured data and inferring constraints involves characterizing the forms of typing/inference supported [1], choosing an appropriate representation for modeling schema [17], and designing efficient algorithms for extracting schema [53]. The first activity is typically driven by the goals of accuracy, approximation, compression, completeness, and storage efficiency. The representations vary from datalog rules [53] to description logics [17]. Algorithms appropriate for mining schema are based on simulation, constraints, and the minimum description length (MDL) principle. Nestorov et al. [53] provide one of the widely used approaches that employs fixed-point semantics to obtain compact representations for semistructured data (see Fig. 4). The result of mining is a set of Datalog rules whose interpretation refers to the schematic organization of information in the web resources. The cost of the mining algorithm is double-quadratic in the size of the web site (the original algorithm in [53] ignores pre-leaf nodes). For web sites that are purely hierarchical and that do not contain cycles, more simplifications are available that enable efficient implementations of the mining algorithm [53].

Characterizing Sources of Semistructure: Our proposed solution is to characterize specific sources of semistructure and model them to yield compressions that are especially appropriate for personalization. Arguably, the work in [53] is in part motivated by this consideration: the authors explore two basic forms of reductions typically arising from semistructure: (i) using the minimal number of internal nodes (types) to model the resource, and (ii) collapsing and using superimposed roles to arrive at size reductions (see the bottom two graphs in Fig. 4). These reductions, however, operate on a graph-based OEM-type schema supplied by crawlers, which frequently miss many important domain-specific sources of semistructure. A web crawler provides simple forms of URL indexing and pattern matching that can help remove purely navigational links that do not yield any benefit in modeling. However, many important types of links (and consequently, sources of semistructure) can only be uncovered by a careful examination of a web site (a one-time cost justifiable for the creation of a personalization system). While it is difficult to address this issue in all generality, we illustrate the basic idea using the GAMS (Guide to Available Mathematical

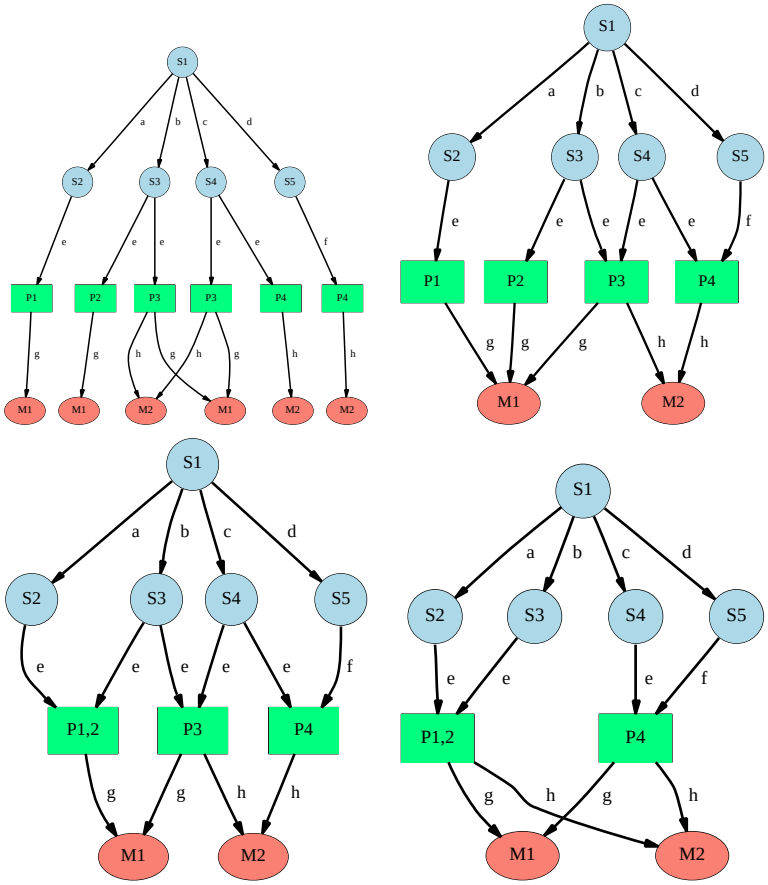

Figure 4: Four stages in mining schema from a semistructured data source, with slight adaptations of the algorithm presented in [53]. The input is assumed to be a graph with labeled and directed edges (top left). Commonalities encountered in tree-building are factored first (top right). At this stage, multiple internal nodes may possess the same input and output labels (for example, `P1` and `P2`) to the same nodes. The algorithm of Nestorov et al [53] then proceeds to type the data, thus collapsing `P1` and `P2` (bottom left). Finally, per [53], nodes are allowed to belong to multiple types, rendering `P3` to be redundant and expressable as a superposition of `P1,2` and `P4` (bottom right).

Software) taxonomy of mathematical software (`http://gams.nist.gov`).

Example 2 (Mining GAMS Schema): GAMS is a tree-structured taxonomy that indexes nearly 10,000 algorithms (from over 100 software packages at 4 repositories) for most areas of scientific software. It is used in an interactive manner, guiding the user from the top of a classification tree to specific software modules as the user describes the problem in increasing detail. During this process, many features of the problem are determined, indirectly from the user. Closer inspection of GAMS reveals at least four different sources of semistructure: (i) GAMS organizes its taxonomy into a hierarchy consisting of Classes, Subclasses, Packages, and Modules. However, not all levels of the hierarchy contain the same type of pages. For example, some Subclass pages link to not only other Subclasses, but also directly to Packages. (ii) For certain problem types, more than one category in GAMS is pertinent. For example, optimization software reside not only in the subtree rooted at G ('optimization'), but also K ('approximation'), and L8 ('regression'). Some of this information is provided via "Search also" labels on the corresponding links. Modeling this aspect

Subclass	Number of Types	Compression Ratio
A (Arithmetic, Error Analysis)	58	13 %
C (Elementary and Special Functions)	268	12 %
D (Linear Algebra)	763	14 %
H2a (1-D Quadrature)	69	14 %
I2b1a (Linear, 2nd Order, Elliptic PDEs)	24	11 %
L (Statistics, Probability)	1330	51 %
Entire GAMS Site	2793	60 %

Table 1: Summary results of mining semistructure from various GAMS (`http://gams.nist.gov`) subtrees. The number of types denotes the number of program entities (variables) necessary to model the subtree. The compression ratio denotes the savings obtained vis-a-vis a naive recursive depth first crawl of the web sites.

with graph-based schema introduces additional links that aid in the mining process. (iii) We also create a virtual page for every instance of a distinct set of modules associated with a package and pointed to by a non-leaf node. For example, different subsets of the CMLIB package are reproduced multiple times at various leaves. Mapping these into a virtual entry provides an additional source for commonalities at the lower levels of the taxonomy. And finally, (iv) web pages separated by two consecutive links, with no branching choices in between, can be collapsed as follows:

$$\text{if a if b ...} \Rightarrow \text{if a and b ...}$$

These rules are modeled declaratively and used in conjunction with the mining algorithm presented earlier. A tabulation of results for various GAMS subtrees is provided in Table. 1 and a detailed view of one of the subtrees is provided in Fig. 5. As can be seen, this methodology is very powerful and scales to the full scope of the GAMS site which houses tens of thousands of web pages. Notice further that the mining process only serves to reduce the schema (for partial evaluation), and individual web pages that get factored together will have to be indexed (by a hash, say) so that personalization could reproduce them if necessary. By this approach, we hope to gain insight into typical sources of semistructure and use experiences from one case study to leverage in newer application domains. The XTRACT system proposed in [28] addresses the issue of extracting DTDs and schema within individual pages and documents. Such techniques can be included in our methodology if the problem domain requires the modeling of information at this granularity.

Clustering and Topical Compression: A final variety of schema compression involves using clustering techniques and approximations [53, 61] to 'collapse' together types and labels that satisfy, say, a certain distance metric. Various suggestions based on geometrical considerations are provided in [53]. The BEV case study, discussed in [60], shows that for some domains, it is acceptable (and even desirable) to be less strict in variable assignments, thus yielding more false positives. Personalizing for 'galleries' might benefit from setting related variables such as 'museums' and 'showrooms.' Such assignments are facilitated by the availability of orthogonal decompositions (such as singular value decompositions of the term-document matrix [26], or Lanczos decompositions [14]) that geometrically reveal semantic relationships. These approximations identify hidden structures in word usage, thus enabling searches that go beyond simple keyword matching (see, for example [14]). Rank reduction techniques (SDD [40], SVD [11, 12, 47, 10]) for type clustering are applicable here as they have been shown to be especially appropriate for latent semantic indexing in information retrieval.

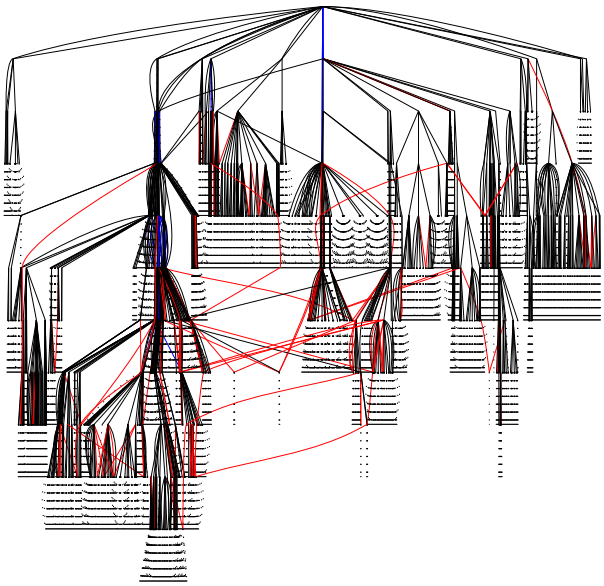

Figure 5: The L subtree of the GAMS site (software modules relating to statistics and probability) after modeling domain-specific sources of semi-structure. The leaves correspond to software modules and the internal nodes represent groupings of mathematical software according to a classification taxonomy. The schema depicted here is 40% of the size of the schema obtained by using naive web crawling. The blue links connect to subtrees that provide related software modules and the red links identify duplicated module sets; together, they form the primary source of compression.

Controlling the Complexity: While reducing the schema for partial evaluation, the computational complexity of partial evaluation should not be allowed to render the overall process ineffective. Partial evaluation is, in general, a costly operation because of the need to unroll loops and complex control structures in programs. However, such features are almost always absent in the kinds of structures considered here (web sites). Even links that point back to higher levels of the hierarchy do not cause code blowups since they typically connect otherwise disjoint subtrees and can be factored out by partial evaluation (using program point specialization, a technique that embeds smaller-sized functions for particular arguments). In addition, it can be easily shown by simple binding time analysis that the above compressions all guarantee termination of the partial evaluator (with respect to hierarchical sites).

Other Forms of Partial Evaluation: Recall from Fig. 3 that the mutually exclusive dichotomies of link labels are modeled by using an `else if` construct. While this has the advantage of supporting disjunctions and conjunctions in personalization scenarios, the automatic determination of such constructs is an important open question. Automating this aspect in the crawling process requires either apriori meta-data, explicit user direction, and/or mining browsing patterns (to see that user sessions that retrieved documents pertaining to Republicans did not retrieve documents pertaining to Democrats, for instance). We believe that this leads to the more important issue of personalizing (partially evaluating) w.r.t. 'collaborative features,' as opposed to simple content-based labels. Navigation-based personalization, such as 'footprints' [78] and XML-based description formats [23] are promising frameworks in which to conduct such studies.

Systems, Algorithms, and Tools: The techniques outlined so far do not require any sophisticated facilities for implementation. The framework proposed to carry out the experiments include

Perl (Perl provides hashes indexed by strings, useful when the same web page is encountered multiple times), the C-Mix partial evaluator, and the text processing capabilities of the `lynx` browser (used to populate individual hash table entries). We can customize wrappers for individual application domains. Various solutions have been proposed, notably induction of wrappers [44] and automatic generation by specification of formatting and conversion routines [7]. PIPE, by design and intention, is most appropriate when 'multiple views' of a site are requested by visitors (ref. the politicians example). Techniques such as those presented in [76] are also motivated by the same goal. To restructure web sites based on results of partial evaluation, we intend to explore the use of approaches such as WebStrudel [5, 24] that use declarative specifications to automatically generate pages. With the above implementation choices, the online time taken for partial evaluation is within design limits: O(2 seconds) for a typical personalization scenario in the politics case study (which involves thousands of pages). This is crucial from both the usability and efficiency points of view. The ability to rapidly transform web sites is increasingly gaining attention in industry — several commercial web sites already use XML to specify site constraints, structure and quickly reorganize their layouts to retain customer interest. Related ideas are covered in [6].

3 Design for Personalization

The discussion thus far has concentrated on hierarchical sites where designers and users of web sites find it intuitive to simplify information presentation by specifying information selection attributes. In addition, a taxonomy is typically available that supports a primary browsing paradigm. It can be argued that the PIPE methodology is a *natural* in this case, where functional modeling of the information resource corresponds to the evaluation of a program. This provokes various interesting questions — What types of information resources are particularly amenable to the PIPE methodology of personalization? Can more *amorphous* domains such as social networks within an organization fit this framework? Under what conditions will PIPE not work? Are there web sites and scenarios that are more *personable* than others? Can the design of an information resource be architected for personalization [70] (in contrast to querying or navigation)? If so, does PIPE provide the appropriate metaphor for reasoning about this aspect? In order to answer such more general questions of usefulness, we must develop a more complete understanding of the processes by which an online information resource is created, expressed, validated, and communicated.

3.1 Research Issues

There are two critical perspectives to be considered — that of the designer of a web site and of the subsequent users of the resource [42, 54]. A key issue for the designer is the inherent uncertainty about what individual users will be seeking when they begin interacting with the resource. As Example 1 reveals, the design process must be flexible and extensible, as the web site designer learns about user needs from the many paths of information access followed. From the user's perspective, the personalization process must map well to the pre-existing mental models that people bring to the information retrieval task. Again this points to a strong requirement for flexibility, because different users will have different needs. This means that a single set of personalization parameters (and corresponding procedures for applying them) must satisfy a wide range of information retrieval goals.

3.2 Solution Methodology

Our idea is to use scenario-based design (SBD) methods to analyze, design, and refine personalization design and usage tasks in three application domains (described in Section 4) [18, 65]. SBD

Figure 6: (left) Using a scenario editor to define and write narratives for a personalization scenario. (right) Programmatic representation of Bill's scenario in a PIPE implementation.

emphasizes the central role of work context in designing useful and usable interactive systems. The methods involve a combination of analytic and empirical activities — descriptions of use are observed, analyzed, transformed, refined, and evaluated in a continuing cycle of design and analysis. The final result will be three case studies of information personalization, including design rationale that explains how designers and users think about complex information resources, and how the new PIPE personalization services can be used to respond to their needs. **This rationale is a key research result, because it will serve as an initial basis for understanding what it means to design and use partial evaluation techniques for information personalization.**

Example 3 (Problem Scenario Development): This constitutes the beginning of SBD where narratives of use are designed to synthesize and highlight the difficulties and opportunities of current practice. Such problem scenarios are developed through field work that includes observation of work sessions, interviews or surveys of domain experts, and collection and analysis of work artifacts. Consider a finance domain of personalization. Users typically obtain recommendations of stocks from a brokerage firm, navigate to, say, the Yahoo! finance cross-index at [`quote.yahoo.com`] to conduct a ticker symbol lookup, and finally access the web pages of companies by using the profiles, stock charts, and financial statistics (see left part of Fig. 6). Such a recurring scenario could be modeled by observing users' analysis activities for several hours, followed by an interview and subsequent analysis of the design documents or other artifacts generated. The automatic *sessionizing* of user access patterns to demarcate individual sessions is an area of active research [52]. Methodologies for data preparation and gathering for this activity are covered in [50]. Traditionally, the integration of software design with such interaction scenarios is achieved by symbolic modeling techniques, such as motivated by OO and languages such as UML. PIPE addresses this requirement for SBD rather elegantly, since (i) the personalization scenario can be modeled programmatically, thus serving as input to the partial evaluator, and (ii) modes of information integration exploited by the user are identified early in the process and can be modeled as assertions in the control program. For instance, an online brokerage might refer to its recommendations by company name ('Microsoft'), whereas the Yahoo! cross-index uses the ticker symbol ('MSFT'). The control flow of the scenario (modeling such transforms) is then partially evaluated, where the final program is structured as shown in the right of Fig. 6. Analogously, information search logs would be collected for a set of

representative users, on some occasions complemented by 'think aloud' protocols as information retrieval goals are pursued. Relevant artifacts such as notes, printed pages, or other online documents would also be collected.

In SBD, field data such as these are summarized to characterize the users and their tasks. Claims analysis is used to identify tradeoffs that implicit in the current situation — features that have both positive and negative consequences for the web site designers or the site's users [19]. The implicit assumption is that these are the features that will provide the most leverage in a new design. The tradeoffs provide the motivation for scenarios that convey the concerns and opportunities of current practice. Thus it is the identification and discussion of tradeoffs that is accumulated as design rationale in scenario development.

Iterative Refinement of Scenarios: During scenario re-design, the scenarios of current practice are transformed by introducing new functions and interaction techniques. These transformations are grounded in the original tradeoff analysis and the specific techniques are inspired by relevant metaphors or technology [64]. Here, we can explore a range of personalization metaphors (e.g., the auditorium metaphor of Terveen et al. [73] that partitions the problem domain into spheres of interest, the footprints idea of Wexelblat and Maes [78] that gathers 'collaborative' features implicitly) with web site designers and users, and demonstrate how PIPE can be used to simplify a complex information model. Focus will be first on basic functionality — the kinds of tasks that a personalization system should support rather than the details of user interaction. The conceptual task scenarios that result are again analyzed for the tradeoffs they imply, identifying the features of the new situations that are most likely to have a mixture of positive and negative effects.

As the new design is refined, specific proposals for interaction techniques are introduced and considered with respect to any new tradeoffs implied. Prototypes will be developed to gather empirical data about these tradeoffs; the scenario context (e.g., user characteristics, work setting, task goals assumed) serves as a specification for usability data collection. As these data are collected, they are used to argue for refinements or transformations of the design scenarios and become part of the cumulating rationale for design decisions. Thus at the end of the process, the features of the final scenarios can be traced back to issues that were highlighted in the scenarios of current practice, and then addressed via the iterative refinement that takes place during design and usability evaluation.

4 Broadening Aspects

The usefulness of methodologies such as PIPE relies on the expressiveness of their representation(s) and their ability to reason efficiently with such representations [55]. In this section, we show how PIPE achieves these objectives with respect to incrementality (for dynamic web resources) and non-traditional applications. Ethical and privacy implications of the PIPE methodology are also summarized here.

Incrementality in PIPE: When web sites change form and content dynamically, PIPE must be able to seamlessly adapt to variations in structure without requiring the designer to redo the entire offline aspect of program creation. We intend to explore the use of program slicing techniques [13] and other transformation-based approaches that identify program segments that have to be restructured. For example, if a stock value is updated in a financial web site, program slicing (a forward analysis) can help identify program line segments (and web pages, correspondingly) that will be affected by the change and/or pages that must be remodeled to ensure consistency. Slicing is more powerful than partial evaluation (for personalization) since it doesn't constrain the form of partial

information and is also useful for mining dynamically generated web pages (by modeling the code segments generating the pages). The complementary technique of backward slicing enables 'inverse personalization scenarios' such as 'Under what criteria will this site recommend this apartment for me?' — a feature very useful for cost-benefit analysis.

Non-Traditional Applications of PIPE: Personalization is increasingly recognized as not merely a value-added facility, but a fundamental driver of acceptance in domains such as wireless and handheld computing [35]. In such resource-bounded environments, personalization serves to identify the most appropriate subsets of information that should be presented to the user, taking into account bandwidth and time considerations. Typically personalization in such domains is achieved by the use of proxies (and 'agents') that transcode web access by a form of functional indirection. PIPE can be used to compose individual proxy modules programmatically to customize information access. A first step in this direction has been the evolution of standards for specifying properties and attributes, such as the UIML (`http://www.uiml.org`) programmatic markup language for user interface design and resource description formats, such as RDF [23]. Other examples of applications that involve fairly extensive modeling, but are nevertheless amenable to PIPE-like approaches are personalizing information streams (e.g. for news-on-demand), and personalizing TV schedules for users. Symbolic modeling of constraints ('Certain shows have to be watched in sequence,' 'It is okay to skip this program since repeats are aired every Tuesday' etc.) within an online TV web guide can serve as a starting point for partial evaluation.

PIPE also reinforces connections with other aspects of information presentation, such as web site re-organization (using, say, browsing patterns). If the partially evaluated control flows for typical user sessions correspond to disjoint subtrees of a web site, this is a potential indicator for web site restructuring. Dead code identification is another technique that can help identify bad site design choices [72] and help improve site organization. PIPE also emphasizes the functional approach to designing information systems, a feature that is highly desirable when composing individual subsystems to yield large-scope and large-scale solutions. Traditionally functional approaches to IR [30] have emphasized the use of methods in OO models, rule processing, and deductive databases. PIPE constitutes a novel paradigm for functional modeling of personalization systems.

Ethical and Privacy Aspects: PIPE can be construed as a destructive form of personalization, since it can effectively circumvent any original navigation flow intended by the web site designer. However, by emphasizing a transformation-based approach to web site organization, it also serves as a valuable tool in identifying 'copy cats.' In addition, various rules underlying privacy could be embedded in a PIPE implementation (for example, 'never divulge the address of a person unless the personalization criteria partially evaluates code block C34'). Thus, increasing levels of security could be built in programmatically within an implementation of PIPE.

5 Application Case Studies

We now outline three case studies that can validate and provide insights into the design of personalization systems using PIPE. They are chosen to illustrate different aspects of information access that exercise various important features of the proposed methodology.

Rapidly Evolving Hierarchies: Several prominent web resources are hierarchical with rapidly evolving subtrees and dynamic links. Interaction aspects of users with hierarchies is well-understood from the navigation [69] and visualization viewpoints [77]. However, personalization entails that the organization of the resource mirror the user's understanding of the hierarchy (see Section 1).

We anticipate that SBD in this context will help identify the multiple views of the site typically requested by users and ensure that control segments are in place that model the flow of information in these scenarios. Information integration among subtrees of a hierarchy are especially crucial, and it is under these circumstances that the advantages of PIPE become evident. Possible candidates in the commercial sector include the MedlinePlus health information site (`http://www.nlm.nih.gov/medlineplus`), music indices, encyclopedias, and business directories (e.g. `http://www.hoovers.com`). Sites with a scientific flavor include bioinformatics resources, digital libraries of mathematical functions (e.g. `http://dlmf.nist.gov/Contents`), and the ACM Computing Classification System.

Scenarios with Multiple Modes of Information Flow: In this category, we explore situations where effective personalization requires the integration of information from recommender systems, cross-indices, resource lists, and individual web pages. We have entered into preliminary discussions with the designers of a site that uses pigment analysis catalogs to identify and reveal the palettes of painters in different eras and genres. The web site (`http://www.webexhibits.org/pigments/`) currently provides a power search facility where the interaction scenario is hardwired. Users can search for paintings by artist, style, period, or by membership in a particular pigment group. However, even a simple query such as 'What are the neo-classic styles of paintings that used colors similar to those in the baroque area?' cannot be accommodated. With PIPE, this amounts to partially evaluating with respect to the variables `neo-classic` and `baroque`, using a similarity function to model the information flow. We intend to abstract four different sources of information in PIPE: (i) the catalog contents (which contains paintings from 950 to 1981), (ii) a palette similarity table, (iii) citations of paintings, and (iv) auxiliary information such as images, histories, where the painting is housed, and other legends. The specific sources of semistructure in this domain arise from overlaps of painting styles. SBD in this case will involve users who are attempting to understand artistic influences and expose color families. Each of these will likely produce a different scenario that will be modeled by PIPE. Such scenarios with multiple modes of information flow are also pertinent in community-wide efforts, such as the Blacksburg Electronic Village (BEV; `http://www.bev.net`). The BEV provides a resource for the New River Valley in Southwest Virginia where nearly 70% of the population use the Internet actively. In its seventh year, BEV offers a wide array of services — information pertaining to arts, religion, sports, education, tourism, travel, museums, health etc. The lack of 'global controls' in such a setting will likely provide opportunities for compression by grouping topically related links (as identified in Section 2).

Sites Based on Social Network Navigation: A surprising number of web sites base their design on a social metaphor of navigating links through a multi-mode network to identify information. The Internet Movie Database at `http://www.imdb.com` while providing basic search facilities, models the network connecting actors, actress, movies, directors, songs etc. Users are able to systematically 'jump connections' to find answers to queries such as 'Which was the movie that first introduced the lead actor in Titanic?' Another well known example is the DBLP bibliography web site that models the network of authors, papers, journals, and conferences. The naive programmatic rendition of such sites will yield spaghetti code, but we anticipate that SBD will reveal recurring scenarios which can help unroll the program to just the right level of indirection, so as to enable partial evaluation. For example, if queries involve not more than two jumps of the 'actor-movie' chain, then the offline program creation can be simplified to model only these extensions, and not a more generic navigational structure. The domains considered here are ones where SBD techniques are most appropriate and together with partial evaluation will help illustrate the benefit of PIPE.

References

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

[2] G. Adomavicius and A. Tuzhilin. User Profiling in Personalization Applications Through Rule Discovery And Validation. In *Proceedings of KDD-99*, 1999.

[3] G. Adomavicius and A. Tuzhilin. Expert-Driven Validation of Rule-Based User Models in Personalization Applications. *Data Mining and Knowledge Discovery*, Vol. 5, 2001.

[4] C.C. Aggarwal, J.L. Wolf, K.-L. Wu, and P.S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. *Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'99)*, 1999.

[5] C. Anderson, A. Levy, and D. Weld. Web-Site Management with Tiramisu. In *Proceedings of the Web/DB Workshop, SIGMOD 1999*, 1999.

[6] V. Anupam, Y. Breitbart, J. Freire, and B. Kumar. Personalizing the Web using Site Descriptions. In *DEXA – Workshop on Internet Data Management (IDM)*, pages 732–738, 1999.

[7] N. Ashish and C. Knoblock. Wrapper Generation for Semi-Structured Internet Sources. *ACM SIGMOD Record*, December 1997.

[8] M. Balabanović and Y. Shoham. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, Vol. 40(3):pp. 66–72, 1997.

[9] N.J. Belkin. Helping People Find What They Don't Know. *Communications of the ACM*, Vol. 43(8):pp. 59–61, August 2000.

[10] M.W. Berry, Z. Drmac, and E.R. Jessup. Matrices, Vector Spaces, and Information Retrieval. *SIAM Review*, Vol. 41(2):pp. 335–362, 1999.

[11] M.W. Berry, S.T. Dumais, and G.W. O'Brien. Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, Vol. 37(4):pp. 573–595, 1995.

[12] M.W. Berry and R.D. Fierro. Low-Rank Orthogonal Decompositions for Information Retrieval Applications. *Numerical Linear Algebra with Applications*, Vol. 3(4):pp. 301–328, 1996.

[13] D. Binkley and K. Gallagher. Program Slicing. *Advances in Computers*, Vol. 43, 1996.

[14] A. Booker, M. Condliff, M. Greaves, F.B. Holt, A. Kao, D.J. Pierce, S. Poteet, and Y.-J.J. Wu. Visualizing Text Data Sets. *IEEE Computing in Science and Engineering*, Vol. 1(4):pp. 26–34, July/August 1999.

[15] S. Boykin and A. Merlino. Machine Learning of Event Segmentation for News On Demand. *Comminications of the ACM*, Vol. 43(2):pp. 35–41, 2000.

[16] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A Query Language and Optimization Techniques for Unstructured Data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages pages 506–516, 1996.

[17] D. Calvanese, G. Giacomo, and M. Lenzerini. What can Knowledge Representation do for Semi-Structured Data? *Proc. 15th National Conference on Artificial Intelligence (AAAI'98)*, 1998.

[18] J.M. Carroll. *Making Use: Scenario-Based Design of Human-Computer Interactions*. MIT Press, Cambride, MA, 2000.

[19] 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):pp. 181–212, 1992.

[20] J.M. Carroll and M.B. Rosson. Developing the Blacksburg Electronic Village. *Communications of the ACM*, Vol. 39(12):pp. 69–74, 1996.

[21] S. Chakrabarti, B.E. Dom, D. Gibson, J. Kleinberg, R. Kumar, P. Raghavan, S. Rajagoplan, and A. Tomkins. Mining the Link Structure of the World Wide Web. *IEEE Computer*, Vol. 32(8):pp. 60–67, August 1999.

[22] S.S. Chawathe. Describing and Manipulating XML Data. *Bulletin of the IEEE Technical Committee on Data Engineering*, Vol. 22(3):pp. 3–9, 1999.

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

[24] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experience with a Web-Site Management System. *Proc. ACM International Conf. on Management of Data (SIGMOD'98)*, 1998.

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

[26] P.W. Foltz and S.T. Dumais. Personalized Information Delivery: An Analysis of Information Filtering Methods. *Communications of the ACM*, Vol. 35(12):pp. 51–60, 1992.

[27] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The Tsimmis Approach to Mediation: Data Models and Languages. In *Proceedings of Second International Workshop on Next Generation Information Technologies and Systems*, pages 185–193, 1995.

[28] M. Garoflakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A System for Extracting Document Type Descriptors from XML Documents. *Proc. ACM International Conf. on Management of Data (SIGMOD'2000)*, 2000.

[29] 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):pp. 270–293, 1999.

[30] P. Gray, P. King, and L. Kerschberg. Functional Approach to Intelligent Information Systems. *Journal of Intelligent Information Systems*, 2000. to appear.

[31] J. Hammer, H. Garcia-Molina, J. Cho, A. Crespo, and R. Aranha. Extracting Semistructured Information from the Web. In *Proceedings of the Workshop on Management for Semistructured Data*, 1997.

[32] H. Hirsh, C. Basu, and B.D. Davison. Learning to Personalize. *Communications of the ACM*, Vol. 43(8):pp. 102–106, August 2000.

[33] B.A. Huberman, P. Pirolli, J. Pitkow, and R.J. Lukose. Strong Regularities in World Wide Web Surfing. *Science*, Vol. 280:pp. 95–97, 1998.

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

[35] A. Joshi. On Proxy Agents, Mobility, and Web Access. *ACM Baltzer Journal of Mobile Networks and Applications (MONET)*, 2000. to appear in the Special Issue on Software Architectures for Mobile Applications.

[36] P.B. Kantor, E. Boros, B. Melamed, V. Menkov, B. Shapira, and D.J. Neu. Capturing Human Intelligence in the Net. *Communications of the ACM*, Vol. 43(8):pp. 112–115, August 2000.

[37] J. Karat, C.-M. Karat, and J. Ukelson. Affordances, Motivation, and the Design of User Interfaces. *Communications of the ACM*, Vol. 43(8):pp. 49–51, August 2000.

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

[39] 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 *AAAI 1998, The Fifteenth National Conference on Artificial Intelligence*, 1998. Madison WI.

[40] T.G. Kolda and D.P. O'Leary. A Semidiscrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval. *ACM Transactions on Information Systems*, Vol. 16(4):pp. 322–346, 1998.

[41] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, Vol. 40(3):pp. 77–87, March 1997.

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

[43] S.R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for Emerging Cyber-Communities. In *Proceedings of the Eighth World Wide Web Conference*. 1999. Toronto, Canada.

[44] N. Kushmerick, D.S. Weld, and R.B. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of IJCAI'97*, pages 729–737. 1997.

[45] S. Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, Vol. 280(5360):pp. 98–100, 1998.

[46] L. Lei, G.-H. Moll, and J. Kouloumdjian. A Deductive Database Architecture Based on Partial Evaluation. *SIGMOD Record*, Vol. 19(3):pp. 24–29, 1990.

[47] T.A. Letsche and M.W. Berry. Large-Scale Information Retrieval with Latent Semantic Indexing. *Information Sciences - Applications*, Vol. 100:pp. 105–137, 1997.

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

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

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

[51] M.D. Mulvenna, S.S. Anand, and A.G. Buchner. Personalization on the Net Using Web Mining. *Communications of the ACM*, Vol. 43(8):pp. 123–125, August 2000.

[52] O. Nasroui, H. Frigui, A. Joshi, and R. Krishnapuram. Mining Web Access Logs Using Relational Competitive Fuzzy Clustering. *Proc. of the Eighth International Fuzzy Systems Association World Congress*, August 1999.

[53] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting Schema from Semistructured Data. *Proc. ACM International Conf. on Management of Data (SIGMOD'98)*, 1998.

[54] Jakob Nielsen. User Interface Directions for the Web. *Comminications of the ACM*, Vol. 42(1):pp. 65–72, 1999.

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

[56] M. Perkowitz and O. Etzioni. Adaptive Web Sites. *Communications of the ACM*, Vol. 42(8):pp. 152–158, 2000.

[57] S. Perugini, P. Lakshminarayanan, and N. Ramakrishnan. Web Mechanics: Personalizing the GAMS Cross-Index. *IEEE/AIP Computing in Science and Engineering*, 2000. Invited Paper. To appear.

[58] B.J. Pine, S. Davis, and B.J. Pine II. *Mass Customization*. Harvard Business School Press, Boston, MA, April 1999.

[59] P. Pirolli, J. Pitkow, and R. Rao. Silk from a Sow's Ear: Extracting Usable Structures from the Web. In *Proceedings of CHI'96*. 1996. Vancouver, Canada.

[60] N. Ramakrishnan. PIPE: Personalization is Partial Evaluation. *IEEE Internet Computing*, Vol. 4(6), Nov-Dec 2000. Accepted for Publication. to appear.

[61] N. Ramakrishnan and A.Y. Grama. Data Mining: From Serendipity to Science. *IEEE Computer*, Vol. 32(8):pp. 34–37, August 1999.

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

[63] D. Riecken. Personalized Views of Personalization. *Communications of the ACM*, Vol. 43(8):pp. 26–28, 2000.

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

[65] M.B. Rosson and J.M. Carroll. *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. Morgan Kaufmann, Redwood City, CA, 2001. in press.

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

[67] D. Rus and J. Allan. Does Navigation Require More than One Compass? In *Proceedings of the 1996 Pacific Rim Conference on Artificial Intelligence*, 1996.

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

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

[70] B. Schneiderman. Designing Information-Abundant Web Sites: Issues and Recommendations. *International Journal of Human-Computer Studies*, Vol. 47(1), 1997.

[71] C. Shapiro and H. Varian. *Information Rules: A Strategic Guide to the Network Economy*. Harvard Business School Press, Boston, MA, November 1998.

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

[73] L. Terveen, W. Hill, and B. Amento. Constructing, Organizing, and Visualizing Collections of Topically Related Web Resources. *ACM Transactions on Computer-Human Interaction*, Vol. 6(1):pp. 67–94, March 1999.

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

[75] B. Thomas. URL Diving. *IEEE Internet Computing*, Vol. 2(3):pp. 92–93, 1998.

[76] J.D. Ullman. Information Integration Using Logical Views. *Proc. Int. Conf. Database Theory (ICDT'97)*, 1997.

[77] L.C. Vroomen. Graphical User Interfaces For Hierarchies: Workshop Summary. *SIGCHI Bulletin*, Vol. 30(2), April 1998.

[78] A. Wexelblat and P. Maes. Footprints: History-Rich Web Browsing. In *Proceedings of the Conference on Computer-Assisted Information Retrieval (RIAO)*, pages 75–84. 1997.