PIPE: Personalizing Recommendations via Partial Evaluation

Naren Ramakrishnan
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061
Email: naren@cs.vt.edu

Keywords: Personalization, recommender systems, partial evaluation, personalization applications.

Abstract

It is shown that personalization of web content can be advantageously viewed as a form of partial evaluation — a technique well known in the programming languages community. The basic idea is to model a recommendation space as a program, then partially evaluate this program with respect to user preferences (and features) to obtain specialized content. This technique supports both content-based and collaborative approaches, and is applicable to a range of applications that require automatic information integration from multiple web sources. The effectiveness of this methodology is illustrated by two example applications — (i) personalizing content for visitors to the Blacksburg Electronic Village (`http://www. bev.net`), and (ii) locating and selecting scientific software on the Internet. The scalability of this technique is demonstrated by its ability to interface with online web ontologies that index thousands of web pages.

1 Introduction

Personalization of web content constitutes one of the fastest growing segments of the Internet economy today [19, 21]. It helps to retain customers, reduces information overload, and enables mass customization in E-commerce [16].

Two main approaches have been proposed to conduct personalization. The simplest are web search engines and the information filtering schemes which use *content-based techniques* to alleviate information overload. They, however, harness only a small fraction of the indexable web (one study estimates this to be $< 30\%$ [14]), and still require users to sift through a multitude of results to determine relevant selections. The low coverage of search engines is attributed to at least two reasons: (i) a majority of web pages are dynamically generated [7] (and hence not directly accessible via hyperlinks), and (ii) lack of sophisticated conceptual models for web information retrieval. At the other end of the spectrum, *collaborative filtering techniques* mine user access patterns, web logs, preferences, and profiles to precisely tailor the content provided ("Since you liked 'Sense and Sensibility,' you might be interested in 'Pride and Prejudice' too") at specific sites. As businesses race to provide comprehensive experiences to web visitors, various combinations of these two approaches are used. This has spawned a multimillion dollar industry (NetPerceptions, Imana etc.) that provides custom-built personalization solutions for individual client specifications. See the web portal `www.personalization.com` for information on all aspects of this industry.

In this article, a customizable methodology called PIPE is presented that can be used to design personalization systems for a specific application (involving a collection of sites). PIPE allows the incorporation of both content-based and collaborative filtering techniques. It supports information integration, varying levels of input by web visitors, and facilitates ease of construction by a skilled systems engineer. For example, a designer wishing to construct a personalization facility for 'wines' can model various web resources (pertaining to this domain) using PIPE and create a facility that customizes content for visitors, based on wine preferences and attributes.

The rest of the article is organized as follows. Section 2 introduces *partial evaluation* — the key concept behind the methodology of PIPE. Sections 3 and 4 develop this idea further and present various schemes that extend it to large domains. Section 5 presents two case studies implemented using this framework. The evaluation of both these implementations are undertaken next, in Section 6. Section 7 discusses various aspects of the PIPE framework, its evaluation, and how it relates to other approaches to personalization.

2 Personalization is Partial Evaluation

The central contribution of this article is to model personalization by the programmatic notion of partial evaluation. Partial evaluation is a technique to specialize programs, given incomplete information about their input [12]. The methodology presented here models a web site as a program (which abstracts the underlying schema of organization), partially evaluates the program with respect to user input, and recreates a personalized web site 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 '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. 1 (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

`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 1: 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 [12].

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. Its benefit is obvious when we consider a higher-level loop that would invoke `pow` repeatedly for computing, say, the squares of all integers from 1 through 100. Partial evaluation is now used in a wide variety of applications (scientific computing, database systems etc.) to achieve speedup in highly parameterized environments. Automatic program specializers are available for C, FORTRAN, PROLOG, LISP, and several other important languages. The interested reader is referred to [12] for a good introduction. While the traditional motivation for using partial evaluation is to achieve speedup and/or remove interpretation overhead, it can also be viewed as a technique to simplify program presentation, by removing inapplicable, unnecessary, and 'uninteresting' information (based on user criteria) from a program.

Example 1: We now present a simple example to illustrate how a web site can be abstracted as a program for partial evaluation. Consider a congressional web site, organized in a hierarchical fashion, that provides information about US Senators, Representatives, their party, precinct, and state affiliations (Fig. 2). Later examples will remove this restriction to hierarchical sites. The individual web pages in Fig. 2 are denoted by the nodes (circles), and the links are assumed to be tagged via some labeling mechanism. Such labels can be obtained from the text anchoring the hyperlinks via '`<a href>`s' in the web pages, or from XML tags [5]. A web crawler employing a depth-first search can then be used to obtain a program, that models the links in a way that the interpretation of the program refers to the organization of information in the web sources. For example, the data in Fig. 2 produces the program (the line numbers are shown for ease of reference):

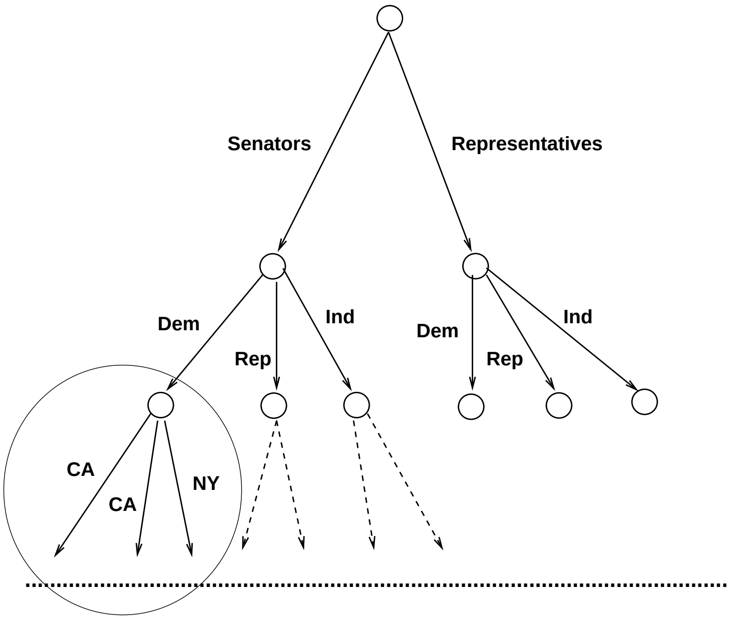

Figure 2: Hypothetical web site organization modeling information about US Senators and Representatives. Only the first few levels of the site are shown; the lower levels can be visualized as modeling individual precincts of politicians, the bills they sponsor, constituencies, their addresses, interests, education etc. The labels on edges represent choices and selections made by a navigator. The circled region indicates the result of personalization for the input 'Democratic Senators.'

```
if (Senators)
   if (Dem)
      if (CA)
        .....
      else if (NY)
        .....
   else if (Rep)
      ....
   ...
else if (Representatives)
   if (Dem)
      ....
```

where the link labels are represented as program variables. The mutually-exclusive dichotomies of links at individual nodes (e.g. 'A Democrat cannot be a Republican') are modeled by `else ifs`[1]. 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

[1]This issue is addressed in more detail in Section 7.

the Senator/Representative home pages can be stored.

Assume that a user is interested in personalizing the web site to provide information only about 'Democratic Senators.' This is easily achieved by partially evaluating the above program with respect to the variables `Senators` and `Dem` (setting them to 1). This produces the simplified program:

```
3:        if (CA)
4:          .....
5:        else if (NY)
6:          .....
```

which can be used to recreate web pages, thus yielding personalized web content (shown by the circular region in Fig. 2). The flexibility of this approach is that it allows personalization even when variable values for certain level(s) are available, but not 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, yielding:

```
1:  if (Senators)
2:     if (Dem)
6:          .....
7:     else if (Rep)
8:         ....
9:     ...
10: else if (Representatives)
11:    if (Dem)
12:        ....
```

The approach is thus responsive to varying levels of user input, ranging from no information (wherein the original program will be reproduced in its entirety) to choices that completely determine the end web page(s).

3 Mining Semi-Structured Data

The simplistic approach presented in *Example 1* will be infeasible for realistic web sites, which are not strictly hierarchical, and best abstracted by 'semi-structured' data models, a term that has come to denote implicit, loose, irregular, and constantly evolving schema of information. To scale this methodology for semi-structured data, we propose the application of data mining techniques that extract compressed schema from web sites.

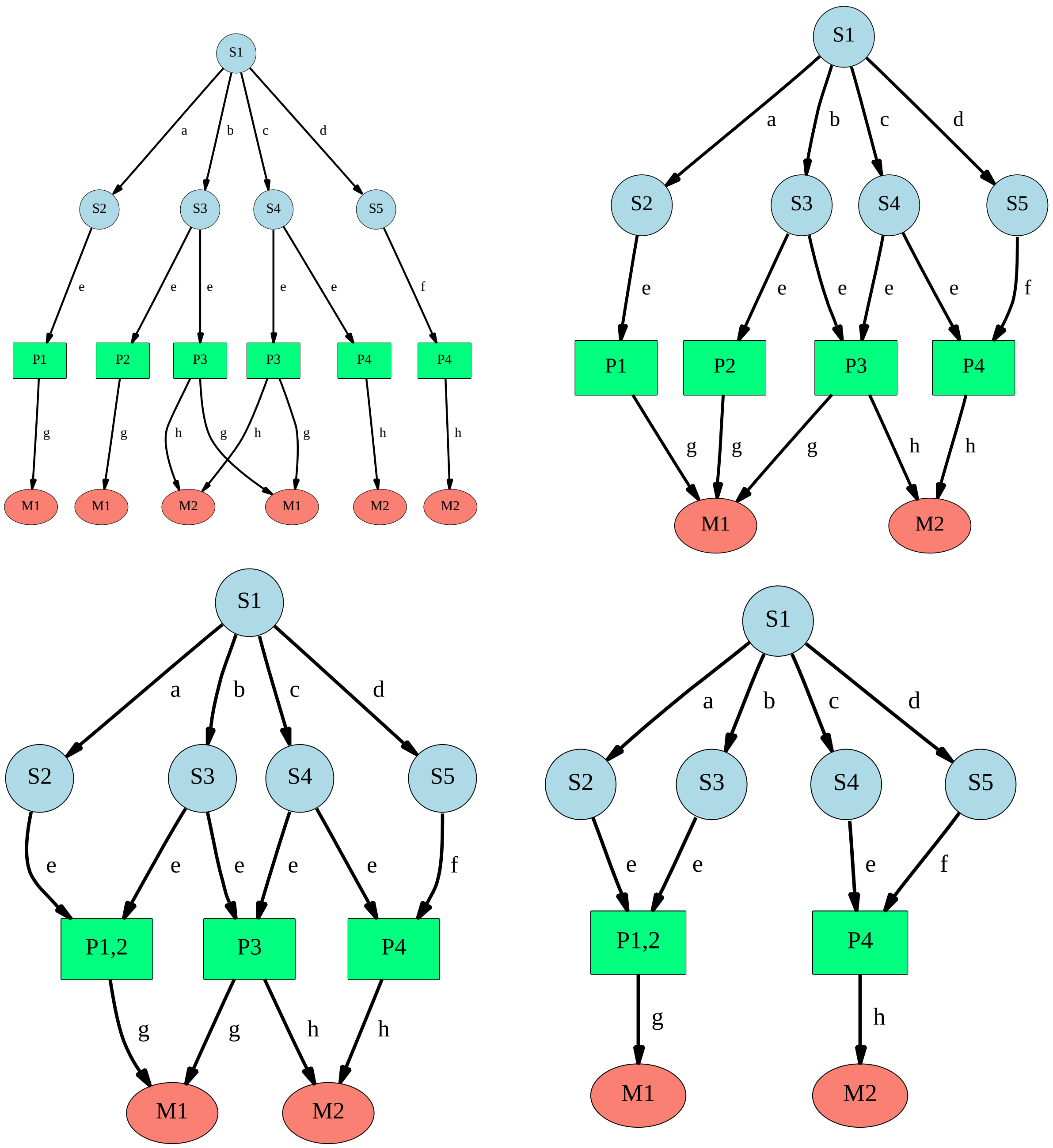

Figure 3: Four stages in mining schema from a semi-structured data source. The input is assumed to be a graph with labeled and directed edges (top left). Commonalities encountered in tree-building are factored first (top right). At this stage, multiple internal nodes may possess the same input and output labels (for example, `P1` and `P2`). The algorithm then proceeds to type the data, thus collapsing `P1` and `P2` (bottom left). Finally, nodes are allowed to belong to multiple types, rendering `P3` to be redundant (bottom right).

Various graph-based models have been proposed to model semi-structured data which, again, use directed labeled arcs to model the connection between web pages and between web sites. The data mining techniques that operate on such models obey the 'approximation model' of data mining. That is, they start with an accurate (exact) model of the data and deliberately introduce approximations, in the hope of finding some latent/hidden structure to the data. We illustrate the basic idea using the approximation model of Nestorov et al. [15], which treats web pages as atomic objects, and models the links between web pages as relations between the atomic objects.

Example 2: Consider the hypothetical web site depicted in the top left part of Fig. 3. The individual web pages are denoted by `M1`, `M2` etc., the pre-leaf nodes by `P1`, `P2` etc., while the other internal nodes are represented as `S1`, `S2` etc. Notice again that the links are assumed to be tagged via some labeling mechanism. The first step in extracting structure from the web site is to proceed to *type* the data i.e., determine the minimum number of entities needed to model the web schema. For example, the `S2` node in the top left part of Fig. 3 can be typed as:

```
S2(Y) :- S1(X), link(X,Y,'a'), P1(Z), link(Y,Z,'e')
```

which indicates that it is reachable from `S1` (using the `a` tag), and has a link to `P1` (via the label `e`). Such a typing (expressed in the form of a logic program) might not yield any compression to the original data, so various approximations and simplifications are employed to reduce its size before partial evaluation. We first identify commonalities due to encountering the same page multiple times (top right of Fig. 3). This is easily achieved by using a hash indexed by page URL in the web crawler. Next, the algorithm of Nestorov et al. [15] uses program-theoretic techniques to find the minimal set of types necessary to accurately represent the original data. For example, `P1` and `P2` have the same input and output labels (to the same page), and can be compressed into a single type `P1,2`, by computing the greatest fixed-point of the logic program (see [15]). And finally, allowing one type to be expressed as the superposition of multiple other types helps further reduce the size of the logic program. In this case, `P3` can be subsumed by a combination of `P1,2` and `P4`. The end-result of this process (see bottom right of Fig. 3) is a succinct schema that can be used for personalization. The cost of the mining algorithm is double-quadratic in the size of the web site (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 [15]. ___

4 Information Integration

The methodology described 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 the graph. For example, if a web page has two links `L1` and `L2` to other pages, the text anchoring `L1` (upto the start of `L2`) is associated with the node corresponding to `L1` in the graph, and so on. While data mining techniques are available that deal with textual information, we do not employ them in our study and implementations. We thus restrict our studies to web sites where most of the information content to be personalized is found at the 'leaves' of the structure tree. In addition, the ideas presented above are restricted to a single site. It is well understood that to provide compelling personalization scenarios, information needs to be integrated from multiple web sites and other sources of information, such as recommender systems [1, 13, 23] and topic-specific cross-indices. Recommender systems, as introduced in Section

1, make selections of artifacts by mining profiles of customer choices and buying patterns. Topic-specific indices provide ontologies and taxonomies by cross-referencing information from multiple sites (e.g. the Yahoo taxonomy).

Example 3: Consider personalizing stock quotes for potential investors. The Yahoo! Finance Cross-Index at [`quote.yahoo.com`] provides a ticker symbol lookup for stock charts, financial statistics, and links to company profiles. It is easy to model and personalize this site by the techniques presented in previous sections. The program obtained could be partially evaluated with respect to ticker symbol to yield specialized information. However, what if the user does not know the ticker symbol, but has access to only the company name? What if he/she desires to index based on recommendations from an online brokerage? The key issue thus is to provide support for information integration from multiple web resources. This entails the inherent inconsistencies and uncertainties in representing and modeling textual labels. The online brokerage might refer to its recommendations by company name (e.g. 'Microsoft'), while the Yahoo! cross-index uses the ticker symbol ('MSFT'). More seriously, financial terms carry with them the twin idiosyncrasies of synonymy and polysemy. For example, 'Investments' referred to in one web site might be listed as 'Ventures' in another, which might mean something completely different in a non-financial setting.

We now outline possible solutions to these issues. The choices made by an individual recommender system can be modeled as statements in a program that abstract the control flow of the selection algorithm. For example, in the financial setting above, a special function can be written that takes as input the current user profile and returns a ticker symbol recommendation. This function can be called from a `main()` routine, which can then use the resulting ticker symbol to set variables for the program obtained by mining the Finance Cross-Index. In addition, the issue of synonymy can be addressed by introducing additional assertions such as:

```
if (MSFT)
   Microsoft= TRUE;
if (Microsoft)
   MSFT = TRUE;
```

at the beginning of the composite program, thus abstracting the task models underlying the application. This is the most domain-specific part of the methodology and cannot be easily automated (this aspect is discussed in more detail later in the paper). The literature on information integration proposes various solutions to this problem, notably wrappers and mediator-based schemes [7]. ___

5 Case Studies

The above three aspects of (i) partial evaluation, (ii) mining semi-structured data, and (iii) information integration form the basis of the PIPE methodology for personalization. PIPE serves as a customizable framework that can compose individual content-based and collaborative engines to form full-fledged personalization systems in a specific domain. To the best of the author's knowledge, there exists no comparable methodology for designing personalization systems, though similar architectures are available for other aspects of information capture and access [20]. We now summarize the various steps of this methodology.

The first step is to identify the different 'starting points' for personalization — a domain specific consideration. The schema in these various sites should be modeled by labeled graphs involving semi-structured data. The second step is to extract typing rules from each of the site structures by the mining algorithm. The third step is to merge the diverse schema into a composite program, taking care to ensure that entities referred to in different ways by individual web sources are correctly merged together. We refer to the information space represented by the composite program as a 'recommendation space.' These steps constitute the *off-line* aspect of the methodology and need be performed only once for a specific implementation. The final step is the online aspect of partially evaluating the composite program and reconstructing the original information from the specialized program. We present empirical evidence for the effectiveness of this approach by application to two diverse domains:

- Creating personalized web pages for scientists and engineers who are trying to locate software on the Internet, and
- Delivering web-based tourist information for visitors to the Blacksburg Electronic Village (`http://www.bev.net`) (BEV).

The common denominator among these applications is their strong emphasis on multiple information resources (needed to achieve the desired effect), the heterogeneity of the sources, and the desire to provide integrated support for interesting personalization scenarios.

5.1 Personalizing Content for Scientists and Engineers

The PIPE methodology has been used in the context of creating personalized recommendations about mathematical and scientific software on the web. One of the main research issues here is understanding the fundamental processes by which knowledge about scientific problems is created, validated and communicated. Designing a personalization system for this domain involves at least three different sources:

- **Web-based software repositories:** In our study, we chose Netlib (`http:// www.netlib.org`), a repository maintained by the AT & T Bell Labs, the University of Tennessee, and the Oak Ridge National Laboratory. Netlib provides access to thousands of pieces of software. Much of this software is organized in the form of FORTRAN libraries. For example, the QUADPACK library provides software routines for the domain of numerical quadrature (the task of determining the areas under curves, and the volumes bounded by surfaces).
- **Individual Recommender Systems:** These systems take a problem description and identify a good algorithm, that satisfies user specified performance constraints (such as error, time etc.). For example, the GAUSS recommender system [18] successfully selects algorithms for numerical quadrature. The issue of how to identify good algorithm recommendations is a very complex and domain-specific one, and is not covered here. We refer the reader to [17, 18] for more details on this problem and a promising approach. It may be noted that GAUSS uses collaborative filtering to correlate variations in algorithm performance to specific characteristics of the problem input. The patterns mined by GAUSS are relational rules that model the connection between algorithms and the problems they are best suited to.
- **Cross-Indices of Software**: The GAMS (Guide to Available Mathematical Software) system (`http:// gams.nist.gov`) provides a web based index for locating and identifying algorithms for scientific computing. GAMS indexes nearly 10,000 algorithms for most areas

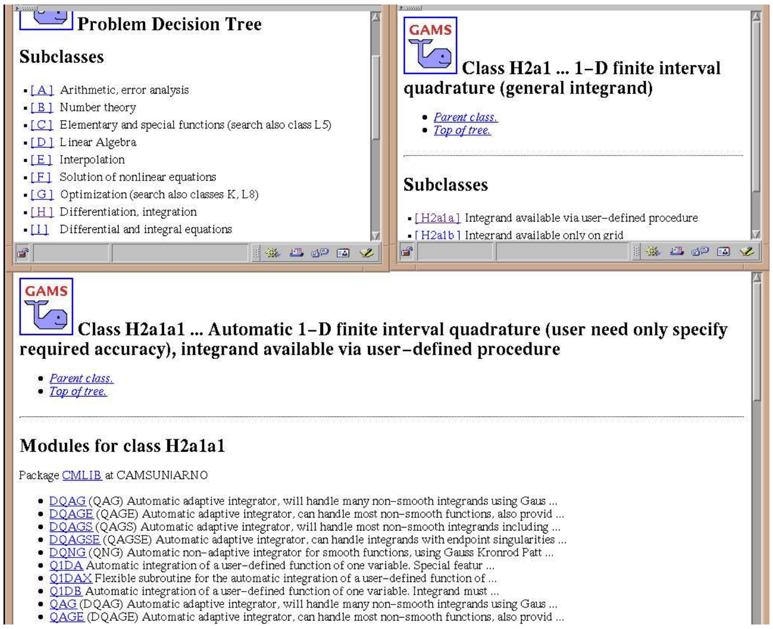

Figure 4: Snapshot of the GAMS Interface (`http:// gams.nist.gov`) at three levels of the hierarchy.

of scientific software. While providing access to four different Internet repositories, GAMS's main contribution to mathematical software, however, lies in the tree structured taxonomy of mathematical and software problems used to classify software modules. This taxonomy extends to seven levels and provides a convenient interface to home in on appropriate modules. Fig. 4 describes three screen shots during a GAMS session. The top left part of Fig. 4 depicts the root GAMS node, where the user is expected to make a selection about the type of problem to be solved (arithmetic/linear algebra/quadrature etc.). The user selects the 'H' category and then proceeds to make further selections. The class H2 corresponds to numerical quadrature, H2a corresponds to one-dimensional numerical quadrature and so on. The H2a1 node is shown in the top right part of Fig. 4. Continuing in this manner, the 'leaves' are arrived at (bottom part of Fig. 4), where there still exist several choices of algorithms for a specific problem. For some domains, it can be shown that there are nearly 1 million software modules that are potentially interesting and significantly different from one another! Recommenders for scientific software exist (as listed above) but they work only in specific isolated subtrees of the GAMS hierarchy (like the GAUSS system).

We present an implementation of PIPE for personalizing recommendations about quadrature software (GAMS category `H2a`). In the absence of PIPE, scientists typically use GAUSS to obtain a recommendation for a quadrature algorithm, then manually navigate the GAMS taxonomy starting from the root, looking for the right category containing an implementation for the recommended algorithm, and finally browse the Netlib site to download the source code and documentation for the recommendation. It is clear that any one of these resources does not provide enough information for personalization.

Experimental Setup: Schema was first extracted from the Netlib site and personalized for the input (QUADPACK=1) (which provides the algorithms for quadrature). The tree-building algorithm was written in Perl using the navigation capabilities of the `lynx` web browser. Perl hashes, which use arrays indexed by page URL, helped identify commonalities in tree-building. The mining algorithm did not yield any compression to the original Netlib schema for QUADPACK, since it is a strict two-level hierarchy. A portion of the schema (simplified for presentation) obtained from Netlib is shown below:

```
if (dqc25s.f)
     URL = "http://www.netlib.org/quadpack/dqc25s.f"
....
if (readme)
     URL = "http://www.netlib.org/quadpack/readme"
```

Next, tree-building and data mining were conducted for the GAMS website rooted at the `H2a` node (one-dimensional numerical quadrature). Notice that while links in GAMS (and most web sites) are not *typed*, we interpret the text anchoring the '`<a href>`'s in the web pages as the label when following the associated link. Furthermore, the labels for certain links (typically to software modules) are very long that they cannot be listed intelligibly on the originating page. In such cases, the label is suffixed with "..." and continued on the page pointed to (see bottom part of Fig. 4). For the purposes of personalization, we cannot ignore the continuation of the label as it may contain important keywords that describe the module. The compressions arising from mining GAMS schema were of two main flavors: (i) reductions due to factoring common nodes at the pre-leaf level (typically module sets), and (ii) reductions arising from links that violate the tree taxonomy. In overall, 80 internal nodes in the `H2a` tree were reduced to 74 nodes (after tree-building) and later, to 69 nodes (after data mining and collapsing multiple roles). Thus, a compression of 14% was observed for the `H2a` GAMS subtree. The schema at this stage is given by:

```
if (Quadrature_Problem)
   if (One-Dimensional_Problem)
      if (Finite_Interval)
         if (Specific_Integrand)
           if (Automatic_Accuracy)
             ....
```

where `Quadrature_Problem`, `Automatic_Accuracy` are the link labels at the GAMS site. Finally, the recommendation rules from GAUSS [18] are already in programmatic form (they take a vector of problem features and performance criteria as input and make a recommendation for an algorithm):

```
if (Int)
  if (Osc)
    if (finite)
       if (HighAcc)
          if (EndPtSing)
             algorithm = "Clenshaw-Curtis Quadrature"
          ....
   .....
```

The above three schemas (programs) were merged taking into account the inconsistencies in the labeling of the three web sources. For example, `Int` in GAUSS is referred to as `Quadrature_Problem` in GAMS, `Finite` in GAUSS is cross-referenced as `Finite_Interval` in GAMS, and so on. The composite program was represented in the CLIPS programming language, which provides procedural, rule-based, and object-oriented paradigms for representation [10]. The final program is structured as:

```
main()
{
        /* assign feature values */
        /* code for matching variables that are cross-referenced */
        /* include program from GAUSS recommender system */
        /* include program from GAMS H2a website */
        /* include program for Netlib site */
}
```

which models the control flow that gets partially evaluated. The end-user interface for the personalization system is shown in Fig. 5. As shown, the user provides the input problem in self-describing mathematical terms. The implementation of PIPE first parses the input to obtain as many symbolic features as possible (it is to be noted that for this domain, this involves some sophisticated mathematical reasoning). For instance, in the example problem shown in Fig. 5, simple parsing reveals that the problem is a quadrature problem (from the presence of the `Int` operator) and that it is a one-dimensional problem (from the range restriction). In addition, further mathematical reasoning reveals the presence of an oscillatory integrand on a finite domain. For more details on how this is achieved automatically, we refer the reader to [18].

Partially evaluating the CLIPS program above with this information by setting the appropriate feature values to 1 starts a cascading effect of program simplification, removing nearly 95% of the original information. The recommendation rules from GAUSS get partially evaluated, in turn navigating the GAMS taxonomy rules, in turn narrowing down on the Netlib URL for the selected algorithm. In this case, the evaluation is actually a *complete evaluation*, since the user has provided enough information to zoom in on a final leaf. The resulting program is then parsed to determine the individual program variables that are set at the end of this process. These are then used to produce the output shown in Fig. 5 that includes (i) the algorithm, (ii) the GAMS annotation, and

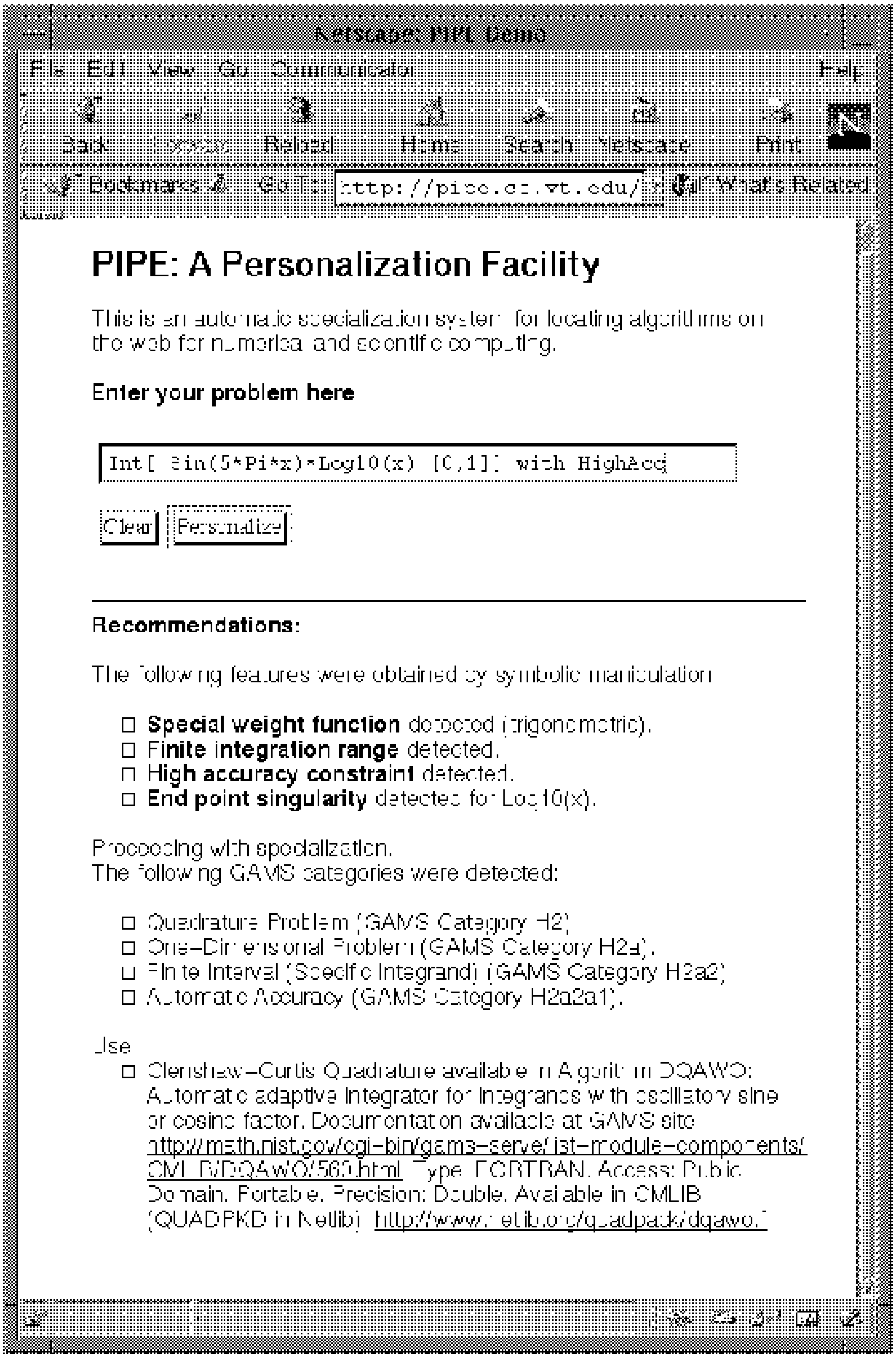

Figure 5: Sample session with the PIPE implementation for personalizing content for a numerical quadrature problem. Notice that the recommendation includes details of the algorithm (and its implementation), the GAMS site from where documentation is available, and the Netlib web repository from where the source code can be downloaded.

(iii) the Netlib annotation indicating the resource from which it can be downloaded. The program segment producing the output shown in Fig. 5 is given by:

```
printout Algorithm "available in" GAMS_annotation
"Available in CMLIB (QUADPKD in Netlib)" URL;
```

Notice the use of the `GAMS_annotation` variable accompanying the node that was evaluated which provides information about the documentation for the algorithm. This implementation of PIPE should not be confused with a service run by Wolfram, Inc. (`www.integrals.com`) that evaluates quadrature problems symbolically.

5.2 Personalizing the BEV

The BEV (`http://www.bev.net`) provides a community resource for the New River Valley in Southwestern Virginia, USA, where nearly 70% of the population use the Internet actively [3]. In its seventh year, BEV offers a wide array of services — information pertaining to arts, religion, sports, education, tourism, travel, museums, health etc. An implementation of PIPE was designed for this facility where the goal is to direct tourists to appropriate resources in the town of Blacksburg. A first plan was to use two resources — the BEV web site (and various other pages that it links to) and the Blacksburg Community Directory (an offshoot of the BEV site). We experienced early problems with this approach due to ambiguities in the descriptions of the BEV entities. Assume that a user is querying for 'art galleries.' Blacksburg boasts of nearly 25 galleries; only 9 of which describe themselves as 'galleries.' Others register their organizations as 'showrooms,' 'centers,' or 'museums' with the BEV site.

Experimental Setup: To overcome these problems, we introduced a third personalization source — TOPIC, a computational distillation of basic keywords and topics used in the BEV site. The basic idea is to use orthogonal decompositions (such as Singular Value Decompositions of the term-document matrix, or Lanczos decompositions) to geometrically model semantic relationships. These approximations identify hidden structures in word usage, thus enabling searches that go beyond simple keyword matching. The exact computational algorithm is beyond the scope of this article, but we refer the reader to papers such as [2, 22] for this approach. The goal of TOPIC is to produce rules similar to the 'Microsoft/MSFT' matching that can be used to model recurrent low-dimensional subspaces in the BEV site(s). The mining algorithm was applied to the BEV site but it did not yield as much a compression to the original data as in the previous example. One reason for this could be the lack of global 'controls' in the construction of web pages by the BEV users and administrators. However, partial evaluation did yield very effective results as shown in the sample query of Fig. 6. In this case, the evaluation is truly partial, since it reproduces a collection of subtrees pertaining to 'coffee.' Notice that the second result depicted in Fig. 6 is a 'false positive,' since TOPIC associates the word 'coffee' with 'cafe' whereas none of the resources identified in 'Blacksburg to Go' are related to coffee shops. A more detailed evaluation of both these case studies follows next.

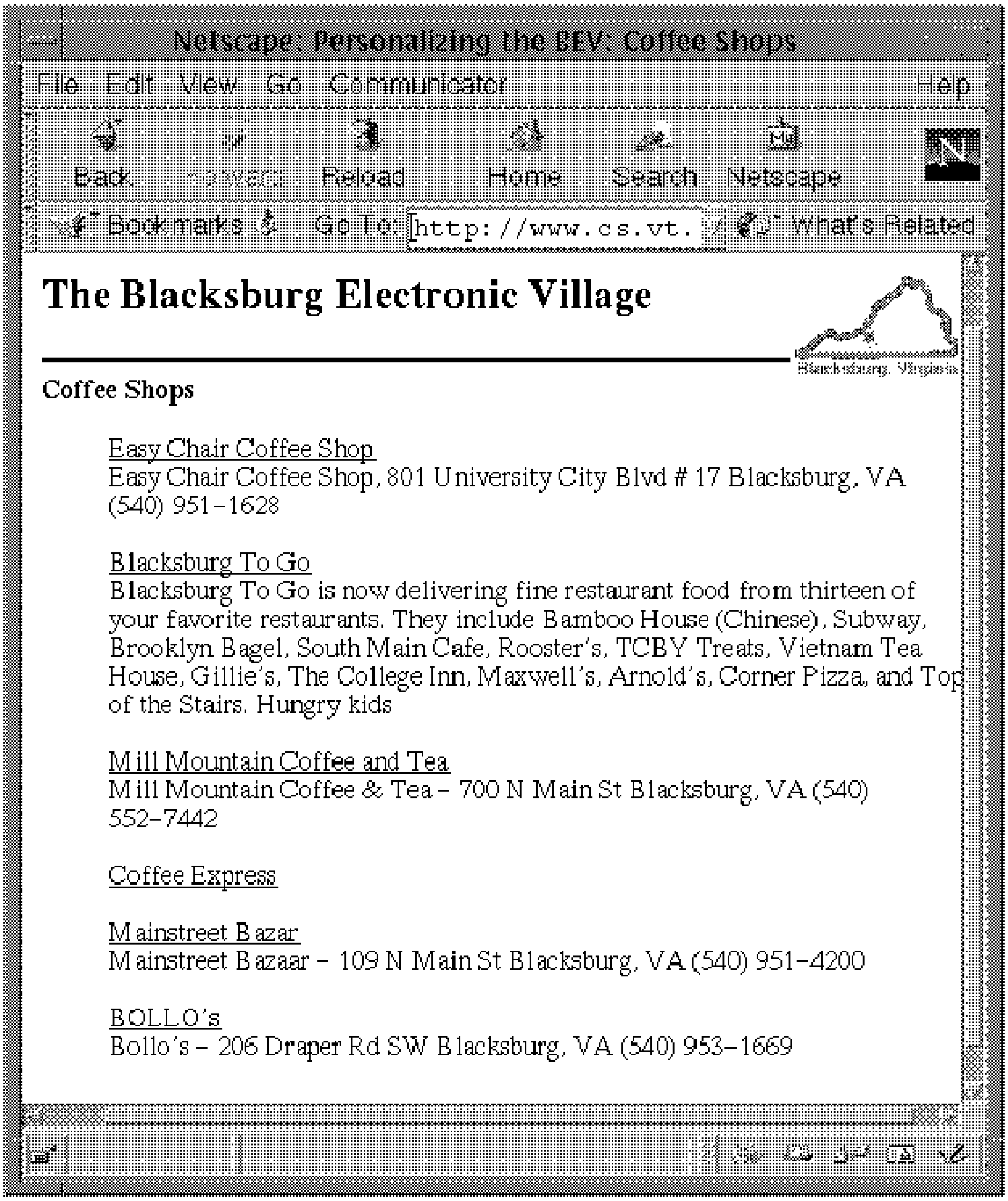

Figure 6: Personalizing the BEV for the query 'Coffee Shops.' The addresses of the various results are stored alongside the nodes as annotations and are reproduced when nodes are selected in the final answer, as in the query above.

6 Evaluation of PIPE

The above described implementations of PIPE involve substantially different methodologies for evaluation. Consider the first case study, a domain of importance and immediate relevance to the computational scientist. Scientists and engineers would be experts at building models in their particular domain, but novices at understanding the intricacies of these mathematical models and the software systems required to solve them. In other words, their ratings/feedback *cannot* be used to evaluate PIPE's recommendations. In addition, there does not exist any comparable comprehensive facility as described in this article. This implementation of PIPE is hence a novel application of personalization technology. To characterize the results, we used a benchmark set of problems described in [18] and applied (ran the algorithms, effectively) the recommendations to see if they indeed satisfied the user's constraints. In addition, we ensured that the web links from GAMS and Netlib were properly associated with all the recommendations.

First, all selections made by this implementation of PIPE were 'valid' (a selection is considered 'invalid' if the algorithm is inappropriate for the given problem, or if a wrong page from GAMS/Netlib was indexed). In addition, we recorded the accuracy of the final selections (a selection is accurate if the selected algorithm does result in solutions satisfying the requested criteria). The best algorithm was selected for 87% of the cases, and the second best algorithm for 7% of the cases. An acceptable choice was made for 3% of the cases (which was not first or second best) and a wrong selection was made for only 3% of the cases. It is to be mentioned that the 'mistakes' arise from the uncertainties in the GAUSS collaborative filtering system, and not as a result of PIPE's methodology. Thus, it is seen that a suitable recommendation is made most of the time.

The evaluation of the second implementation of PIPE is more tricky and serves to illustrate the true potential of our methodology. We adopted the following approach. 10 people were randomly selected and requested to identify 10 queries (each) that might be pertinent to a Blacksburg visitor ('hiking,' 'mountains,' 'trails' etc.). We selected the 10 most frequently cited queries as test cases for PIPE's implementation. Stopword elimination (discarding terms like 'of' and 'the' in queries) and stemming ('hikers' was mapped to 'hiking,' for instance) were first applied to standardize these queries. These queries were then provided to 25 Blacksburg residents who were asked to enumerate the answers (from their point of view) *before* personalization was conducted for these queries. The results obtained were then provided as feedback and they were asked if they would like to change their original answers or if they thought the results were deficient in any respect. Each user voted on a scale of 1–5 the mismatch between the results and any 'expected' answers (where a 1 indicates that he/she is completely satisfied with the results), for each of the queries. We now summarize our results.

First, all votes were in the range 1–2, with the exception of 32 votes with the value 3 (more on this later). For each of the queries, we then conducted a distribution-free test (the Kruskal-Wallis test [11]) where the hypothesis tested was that the results were unanimous versus the alternative that they are not all equal. All 10 hypotheses were accepted at the 95% level, indicating conclusively that the results were very close to the expected answers. The 32 votes with the value 3 were spread over seven people who were less effusive with their ratings than others. This is a standard problem with rating-based collaborative filtering; one way to overcome this is to replace absolute ranks by 'relative ranks' so that they could be captured by certain two-way statistical tests (such as the Friedman, Kendall, and Babington-Smith test [11]). Another approach to overcome effusivity of ratings (or the lack thereof) is presented in [1]. In addition, one of the queries had consistently lower ratings by nearly all 25 participants. This was 'Trails' which failed to reproduce two of the most popular trails in the vicinity of Blacksburg. Not surprisingly, there were no web pages containing information about these trails in the considered collection. The results also fared well when compared with the traditional web search facilities available in the BEV site. For example, the standard BEV search engine produced *no* results for the query 'coffee shops' (or coffee).

7 Discussion and Comparisons with Other Approaches

The effectiveness of the methodology presented above relies on several factors, which are outlined below.

- The PIPE methodology allows programmatic composition to design full-fledged systems and hence, comparisons with individual recommender systems or other specialization facilities for particular domains are not strictly valid. With this in mind, one of its main advantages arises from integrating the design of personalization systems with the task model(s) underlying the assumed interaction scenario. It is this property that allows the designer to view the personalization system as a composition of individual subsystems, using a programming metaphor. PIPE is hence restricted to those domains that are most amenable to such decomposition and analysis techniques. More amorphous domains such as personalizing social networks in an organizational setting might not fit this framework.

- The above implementations assume that (i) the link labels represent choices made by a navigator, and (ii) it is possible to ascertain the values for such labels (program variables) from user input. In both cases this is conveniently achieved, since the GAMS/GAUSS and BEV sites serve as ontologies that help guide the personalization process. While the notion of partial evaluation works for any site even in the absence of ontologies (as shown in *Example 1*), personalization will only be as effective as the ease with which the link labels could be determined or supplied by the user. For example, if a medical informatics site is organized according to scientific names of diseases and ailments, personalizing for 'headaches' will require a parallel ontology or cross-index that maps everyday words into scientific nomenclature. In addition, the BEV case study shows that for certain domains it is acceptable (or even desirable) to be less strict in variable assignments, thus yielding more false positives. For other domains, personalization might pose more stringent demands.

- Personalizing textual content within web pages is not currently-addressed in the PIPE methodology which relies on the accuracy of the links to point to appropriate information. Combining textual mining techniques such as those presented in [2] with the partial evaluation concept is an interesting research issue for future investigation. Other approaches to content-based personalization arise from the database management [7] and information filtering [8] communities. While languages like WebSQL, WebOQL, and Florid [7] provide simple 'web database' lookups, they are not directly suited for personalization purposes since they accept queries in only a limited form, and are more attuned to structure-querying across known levels. However, such systems can be used for program creation and associating augmented data structures with the program that is partially evaluated. In addition, algorithms have been recently proposed that extract structure at the level of a single page. These techniques infer Document Type Definitions (DTDs) and page schemas from example web pages [9]. The integration of link-based analyses and such content-based schemes in a programmatic context also deserves exploration.

- In typical web sites, the links are either mutually exclusive (e.g. in *Example 1* and the GAMS case study), or are inclusive (e.g. the BEV case study). These are currently modeled by the presence or lack of `else if`s in our programs (respectively). This has the advantage of supporting both disjunctions and conjunctions in the personalization queries. Automating this aspect in a web crawler requires more study, e.g. via meta-data or via explicit user direction.

- Partial evaluation, in general, is 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 studies considered here. Even links that point back to higher levels of the hierarchy do not cause code blowups since they are factored by the mining process. As a result, the cost of partial evaluation is not a severe bottleneck. The most expansive implementation of PIPE is the first case study which involved sites that have tens of thousands of web pages. Further analyses are needed, however, to characterize the scale up with respect to an ever greater number of web pages.

- It is instructive to characterize how much of the encouraging results are obtained by partial evaluation vis-a-vis carefully creating a tightly integrated implementation. In the absence of handcrafting, partial evaluation will still lead to personalization, but such a scenario is highly unrealistic. Partially evaluating the program mined from a beverages web site with respect to 'Coke' might not yield any results if the link label says 'Coca-Cola.' Approaches to alleviate this problem include the design of public ontologies and meta-data standards for commercial domains of expertise. Until such a time when such ontologies become prevalent on the web, this problem will not permit any general solutions. In the absence of partial evaluation, though, a more complicated strategy has to be in place to ensure that the personalization system handles all possible types of queries, spanning all combinations of levels of the link labels.

- Domain-specific techniques for making individual recommendations or choices are not part of the PIPE framework *per se* but will nevertheless form a critical aspect of any successful personalization system. Various techniques have been proposed for making recommendations; we refer the reader to [1, 4, 8, 13, 23]. The current methodology thus allows the designer to choose the rating mechanism (value-neutrality). In the case of the first case study, this is the GAUSS recommender system. A different system, trained on qualitatively different examples, might be appropriate in another situation.

- Little attention has been devoted toward automating the determination of appropriate 'starting points' for personalization. In the general case, this should involve a systematic way of finding authoritative sources. A good reference is [4] which uses linear-algebraic matrix transformations to determine the most 'cited' resources (via hyperlinks) for a given topic.

- The integrated methodology of PIPE is similar in spirit to various other systems, most notably Levy's WebStrudel (a web site management system) [6]. Such systems typically take a specification of a web site as input (graphs, rules etc.) and define the structure of the site in terms of the underlying data model. Currently, our implementations customize HTML content using the text manipulation capabilities provided in Perl. Programmatic reconstruction of web pages is a possible future extension of this work and systems like WebStrudel can also eliminate the need for restructuring when more web sources or additional rating mechanisms are introduced.

- In conclusion, the remark of Rus and Subramanian in [20] is especially pertinent:

> "Whether users of the information superhighway prefer to build their own 'hot rods' [by methodologies like PIPE], or take 'public transportation' [web search engines] that serves all uniformly is an empirical question and will be judged by history."

Acknowledgements: The author acknowledges the input of Sammy Perugini, Akash Rai, Mary Beth Rosson and the nearly 40 volunteers who helped in the evaluation of the second study. Feedback from several anonymous referees helped clarify the presentation and improve the article.

References

[1] 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.

[2] 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.

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

[4] 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*, pages 60–67, August 1999.

[5] S.S. Chawathe. Describing and Manipulating XML Data. *Bulletin of the IEEE Technical Committee on Data Engineering*, 22(3):pp. 3–9, 1999. Available at `http://www.cs.umd.edu/~chaw/pubs/`.

[6] 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.

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

[8] 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.

[9] 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. to appear.

[10] J. C. Giarratano. *CLIPS User's Guide, Version 5.1.* NASA Lyndon B. Johnson Space Center, 1991.

[11] M. Hollander and D.A. Wolfe. *Nonparametric Statistical Methods.* John Wiley and Sons, 1973.

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

[13] 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.

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

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

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

[17] N. Ramakrishnan. Experiences with an Algorithm Recommender System. *Proc. CHI'99 Workshop on Interacting with Recommender Systems*, 1999.

[18] N. Ramakrishnan, J.R. Rice, and E.N. Houstis. GAUSS: An Online Algorithm Recommender System for One-Dimensional Numerical Quadrature. *ACM Transactions on Mathematical Software*, 2000. to appear.

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

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

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

[22] 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.

[23] 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.