PIPE: Web Personalization by Partial Evaluation

Naren Ramakrishnan
Virginia Polytechnic Institute and State University

Partial evaluation is a technique popular in the programming languages community. It is applied here as a methodology for personalizing Web content. Two very different implementations of the methodology demonstrate its effectiveness.

Personalization of Web content is one of the fastest-growing segments of the Internet economy. Because it can help in reducing information overload and give users a more customized experience of a Web site, personalization has spawned a multimillion-dollar industry. Companies like Netperceptions (http://www.netperceptions.com) and Yodlee (http://www.yodlee.com) are building custom personalization solutions for individual client specifications.

There are two main approaches to personalization.

- The simplest approach uses *content-based techniques* that filter content by keywords, string matching, and so on. Most Web search engines use these techniques, but they harness only a fraction of the indexable Web (less than 30 percent, according to one study[1]) and, even then, require users to sift through many results to determine relevant selections. One reason for the low coverage is that the majority of Web pages are dynamically generated[2] and hence not directly accessible via hyperlinks. Another reason is the lack of sophisticated conceptual models for Web information retrieval.
- At the other end of the spectrum are c*ollaborative-filtering techniques* that mine user-access patterns, Web logs, preferences, and profiles to tailor the content provided at specific sites (for example, "Since you liked *Sense and Sensibility*, you might also be interested in *Pride and Prejudice*").

There are also some hybrid approaches (see the sidebar, "Related Work in Personalization," next page).

While current personalization systems may use sophisticated algorithms and techniques, they also hardwire the interaction sequences in their interfaces. For example, a personalization facility at an online bookseller may have some users who think of books primarily by title, others who look for a particular author, and still others who would like to personalize with respect to a combination of features. To cover all potential scenarios, the system designer must anticipate every type of situation beforehand and implement customization interfaces (algorithms) for all of them. The absence of an adequate programming model means that designers must make many assumptions and simplifications in the inter-

 1089-7801/00/$10.00 ©2000 IEEE

Related Work in Personalization

Personalization research has evolved to three types of approaches:

- Content-based techniques from the information-filtering and Web database management communities use keywords, string matching, link patterns, and manually compiled identifiers to provide simple "Web query languages" for personalization. Examples include WebSQL, WebOQL, and Florid. For an excellent survey of these and other content-based techniques with a database flavor, see Florescu et al.[1]
- Collaborative-filtering techniques achieve a higher level of sophistication by modeling user behavior, either explicitly or implicitly. This line of research has led to ideas such as clustering Web-access logs, mining user profiles, and collaboratively identifying a community of users. Examples are the GroupLens project at the University of Minnesota[2] and PHOAKS at AT&T Labs.[3]
- Some hybrid systems focus on a major social process (as observed in the real world), model it in its entirety, and use it to form the abstraction for a personalization system. A good example is the Clever search engine[4] (http://www.almaden.ibm.com/cs/k53/clever.html) that models the notion of "authority conferral." Such schemes combine content-based and collaborative-filtering techniques in very sophisticated ways.

Examples of content-based systems include search engines like AltaVista.com and cross-indices such as Yahoo.com. Some search engines augment content-based indexing with link analysis (for example, Google.com) or with conceptual clustering (such as NorthernLight.com). Examples of collaborative-filtering systems include the recommender products of companies like Net Perceptions. For other examples of personalization systems, see the August 2000 special issue of *Communications of the ACM* on this topic.[5]

To my knowledge, no methodology comparable to PIPE exists for designing Web personalization systems, although similar architectures are available for other aspects of information capture and access.[6]

References

1. D. Florescu, A. Levy, and A. Mendelzon, "Database Techniques for the World Wide Web: A Survey," *SIGMOD Record*, vol. 27, no. 3, Sept. 1998, pp. 59-74.
2. J.A. Konstan et al., "GroupLens: Applying Collaborative Filtering to Usenet News," *Comm. ACM*, vol. 40, no. 3, Mar. 1997, pp. 77-87.
3. L. Terveen et al., "PHOAKS: A System for Sharing Recommendations," *Comm. ACM*, vol. 40, no. 3, Mar. 1997, pp. 59-62.
4. S. Chakrabarti et al., "Mining the Web's Link Structure," *Computer*, vol. 32, no. 8, Aug. 1999, pp. 60-67.
5. D. Riecken, "Personalized Views of Personalization," *Comm. ACM*, vol. 43, no. 8, Aug. 2000, pp. 27-28.
6. D. Rus and D. Subramanian, "Customizing Information Capture and Access," *ACM Trans. Information Systems*, vol. 15, no. 1, Jan. 1997, pp. 67-101.

face design. Some of these result from necessity ("This site is organized in this manner and I can't help it"); others may reflect a lack of understanding or appreciation of user needs ("I think this is the best interface to my customers"). In either case, the user may experience serious cognitive and representational frustrations, because the modes of interaction are hardwired (for example, "This interface works only if you specify both the ISBN number and the title").

I have developed a customizable methodology called PIPE (short for "Personalization Is Partial Evaluation"). PIPE is able to personalize Web resources, without enumerating the interaction sequences beforehand. It supports information integration, and varying levels of input by Web visitors. PIPE models personalization as a form of *partial evaluation*, a technique that uses incomplete input information to specialize programs.

This article describes the PIPE methodology and presents experimental results demonstrating its effectiveness in two different domains.

PIPE METHODOLOGY

PIPE is a programmatic framework to design personalization systems. It is based on three concepts: partial evaluation, data mining of semistructured data, and information integration.

Partial Evaluation

The input to a partial evaluator is a program and some static information about its arguments. The output is a specialized version of the program (typically in the same language) that uses the static information to "precompile" as many operations as possible.[3]

Partial evaluation is traditionally used to speed up a program and/or remove interpretation overhead, but it can also be viewed as a technique to simplify program presentation by removing information that doesn't apply to a particular user or is otherwise unnecessary. The PIPE methodology models a collection of Web sites as a program that abstracts the underlying organization of information. The program is then partially evaluated with respect to user input, and a personalized Web site is recreated from the specialized program.

Figure 1 shows a simplified example of specializing a C power function, pow, to create a new function, pow2, that computes only the square of an integer. For a user who computes only squares of integers, the specialized program works as well as the full pow function at considerably less com-

putational cost. Furthermore, pow2 can be obtained automatically from pow by precomputing all expressions that involve exponents, unfolding the for-loop, and performing various other compiler transformations such as copy propagation and forward substitution.

Example 1: Abstracting a Web site into a program. Consider a congressional Web site organized in a hierarchical fashion to provide information about U.S. senators, representatives, their parties, precincts, and state affiliations. In Figure 2, the nodes (circles) denote individual Web pages for the site, and the links represent some labeling mechanism such as the HTML <a href>s that anchor hyperlinks in a Web page or XML tags. In this case, a Web crawler employing a depth-first search could be used to obtain a program that models the links such that the interpretation of the program refers to the schema in the Web sources. For example, the data in Figure 2 corresponds to the following program:

```
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 program models the mutually exclusive dichotomies of links at individual nodes by else if statements (for example, "A Democrat cannot be a Republican"). Although this example models only the organization of the Web site, the textual information stored at each of the internal nodes can be modeled by associating augmented data structures with the program variables. Furthermore, the *leaves,* or innermost sections of the program, can store variable assignments to individual home pages.

If a user is interested in personalizing the Web site to provide information about only Democratic senators, the program can be partially evaluated with respect to the variables Dem and Senators (setting them to 1). This produces the following simplified program:

```
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. Partial evaluation technique. A general-purpose power function written in C (left) is specialized (with exponent **= 2) to create a new function,** pow2 **(right), that computes only the square of an integer. Such partial evaluations are performed automatically by software such as C-Mix.**

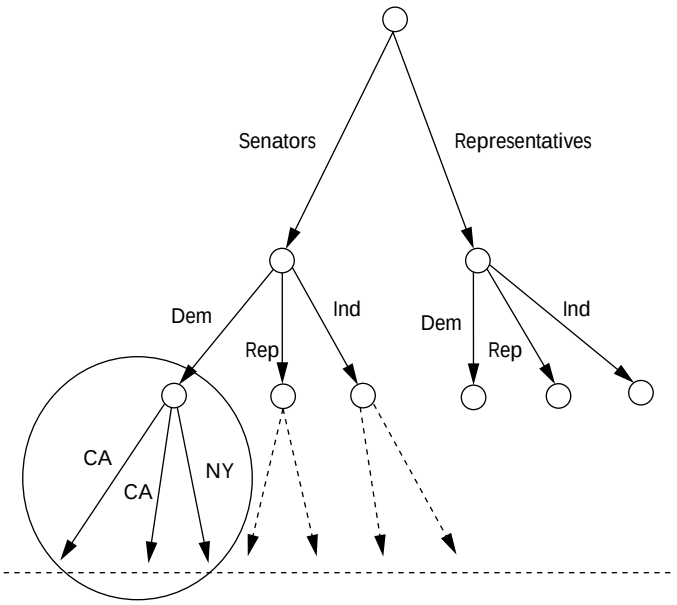

Figure 2. First three levels of a hierarchically organized Web site about U.S. senators and representatives. Lower levels could include detailed information such as congressional precincts, interests, and sponsored bills. The labels on edges represent selections made by a user or navigation program. The circled region indicates the personalization output when the input is "Democratic Senators."

```
if (CA)
    . . .
else if (NY)
    . . .
```

which can be used to recreate Web pages with personalized content (shown by the circular region in Figure 2).

The flexibility in this approach allows personalization when variable values are available for a certain level in the hierarchy but not for levels above

it. For example, say a user desires information about a member of Congress from New York but is unsure whether the member is a senator or representative or a Democrat or Republican. A partially evaluated output will provide personalized content by including New York (by setting NY=1) and excluding other states (by setting variables for other states, such as CA to zero). It can thus simplify lower levels of the tree, without requiring information about the higher levels.

A compelling personalization scenario must integrate information from multiple sources.

Mining Semistructured Data

Example 1 illustrates the concept of using partial evaluation for personalization, but it is not realistic. Most Web sites are not strictly hierarchical and several use links for purposes other than narrowing on an information resource. Realistic sites are best abstracted by semistructured data models that describe implicit, loose, irregular, and constantly evolving schema of information.

To scale the PIPE methodology for semistructured data, we can use data-mining techniques that extract compressed schema from Web sites. The basic idea can be illustrated by using the approximation model of Nestorov et al.,[4] which treats Web pages as atomic objects and models the links between the pages as relations between the atomic objects.

Example 2: Extracting structure from a Web site. In the hypothetical Web site depicted in the top left part of Figure 3, the links are assumed to be tagged via some labeling mechanism. The first step in extracting structure from the site is to type the data—that is, to determine the minimum number of entities needed to model the Web schema. For example, the S2 node in the top left part of Figure 3 can be typed as

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

which indicates that S2 can be reached from S1 (using a link tagged by the label *a*) and has a link to P1 (using a link tagged by the label *e*).

This kind of typing, expressed in the form of a logic program, might not yield any compression of the original data source; so various approximations and simplifications are applied to reduce its size before partial evaluation. Commonalities that show up in encountering the same page multiple times (top right of Figure 3) are first identified. They can be easily found by using a hash indexed by page URL in the Web crawler.

Next, the Nestorov algorithm uses program-theoretic techniques to find the minimal set of types necessary to accurately model the original data source. For example, P1 and P2 have the same input labels and output labels (to the same page). As a result, they can be compressed into a single type P1,2 by computing the greatest fixed point of the logic program.

Finally, one type can be expressed as the superposition of multiple other types, thus further reducing the schema of the original data source and, therefore, the size of the specialized program. In this example, P3 can be subsumed by a combination of P1,2 and P4. Such compressions and approximations serve to reduce only the schema for partial evaluation; they do not discard the textual content in individual pages. The end result of this process (bottom right of Figure 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 (preleaf nodes).

For Web sites that are purely hierarchical and do not contain cycles, Nestorov et al.[4] provide more simplifications that enable efficient implementations of the mining algorithm.

Information Integration

Examples 1 and 2 illustrated only a single site, but a compelling personalization scenario must integrate information from multiple Web sites and other sources of information, such as recommender systems and topic-specific cross-indexes. Recommender systems make selections of artifacts by mining profiles of customer choices and buying patterns (for example, see Aggarwal et al.[5]). Topic-specific indexes provide ontologies and taxonomies by cross-referencing information from multiple sites (as in the Yahoo! taxonomy).

The PIPE methodology can be used to integrate these approaches into the design of personalization systems.

Example 3: Personalizing stock quotes. The Yahoo! Finance cross-index at http://quote.yahoo.com provides a ticker symbol lookup for stock charts, finan-

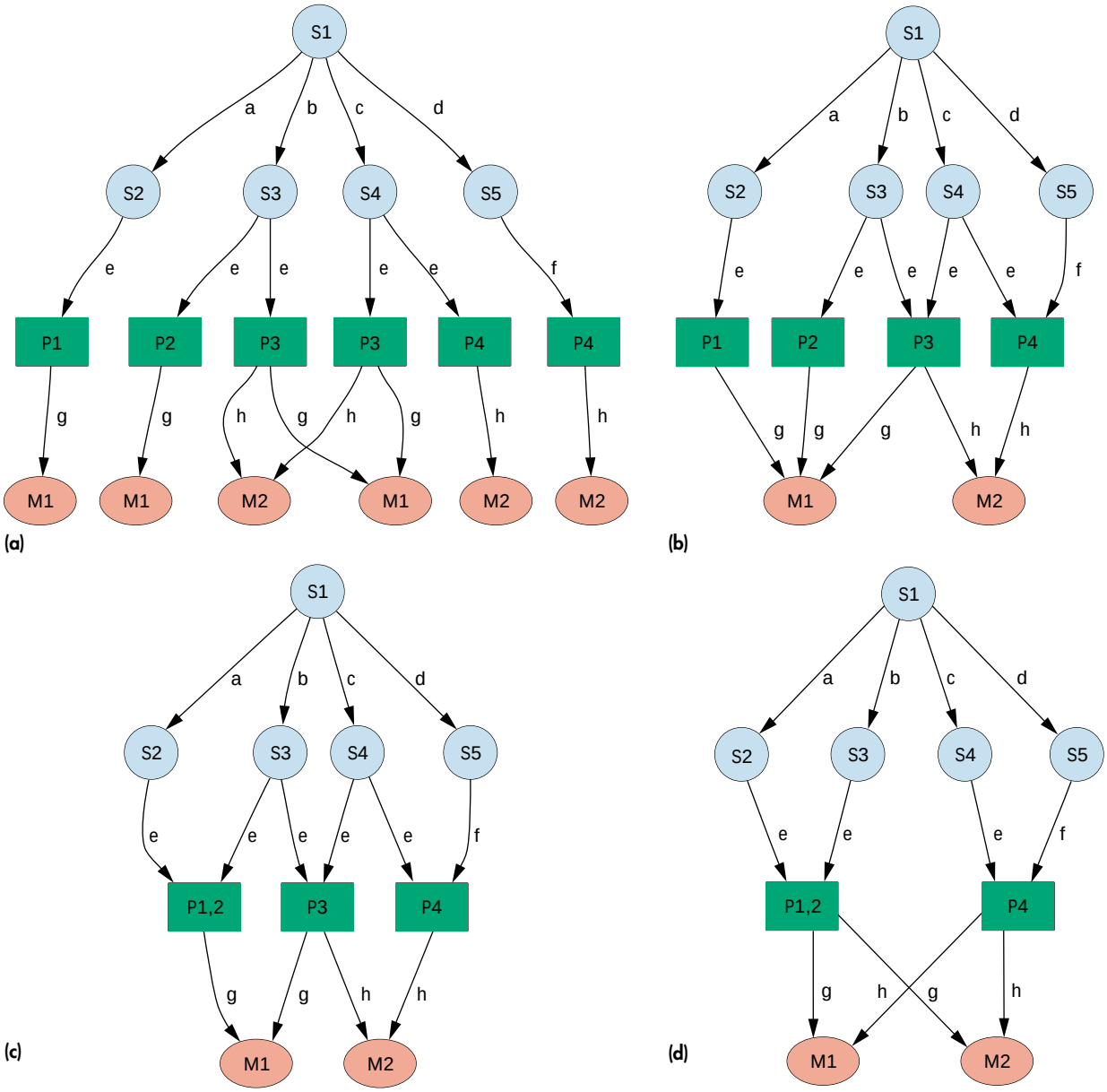

Figure 3. Four stages in mining schema from a semistructured data source. The input is a graph with labeled and directed edges (top left). Commonalities encountered in building the tree are factored first (top right). At this stage, multiple internal nodes may possess the same input and output labels (for example, P1 and P2). An algorithm then types the data, thus collapsing P1 and P2 (bottom left). Finally, the algorithm allows nodes to belong to multiple types, rendering P3 redundant and expressing it as a superposition of P1,2 and P4 (bottom right).

cial statistics, and links to company profiles. It is easy to model and personalize this site by partially evaluating with respect to ticker symbol.

But what if a user knows only the company name and does not know the ticker symbol? What if he or she wants to build an index on the basis of an online brokerage's recommendations? The key issue, then, is to integrate information from Web resources that use different terminology and organizational schema. For example, the online brokerage might refer to its recommendations by company name (Microsoft), but the Yahoo! cross-index uses the ticker symbol (MSFT). Financial terms also carry with them the twin idiosyncrasies of synonymy and polysemy:

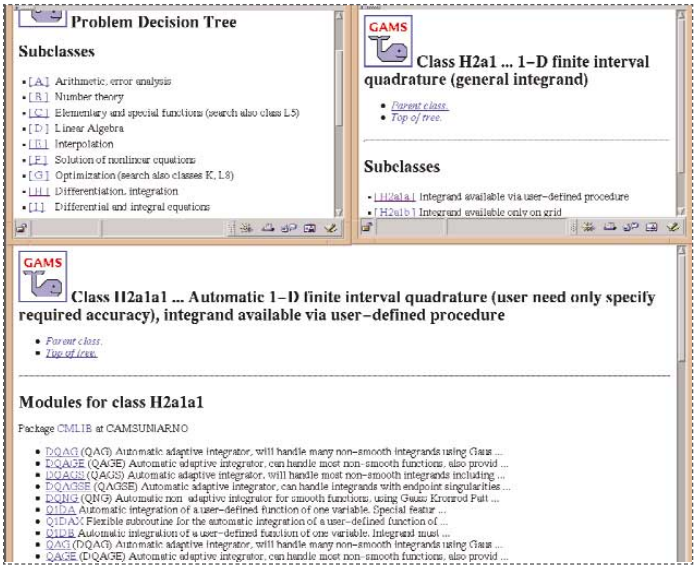

Figure 4. Snapshot of the GAMS interface (http://gams.nist.gov) at three levels of the hierarchy. The Problem Decision Tree (upper left) is the highest level of the cross-index. Finite Interval Quadrature (the H2a1 node, upper right), is four branches down. At the lowest level is a list of algorithms (bottom).

"Investments" on one Web site might be "ventures" on another (synonymy), but "ventures" might also mean something completely different on a Web site not focused on finance (polysemy).

To solve these problems, the PIPE methodology can model the choices made by an individual recommender system as statements in a program that abstracts the control flow of the selection algorithm. For stocks, a special function can take the current user profile as input and return a ticker symbol recommendation. This function can be called from a main() routine to model the Yahoo! Finance cross-index for ticker symbols. The routine can then use the ticker symbols to set variables for the personalized program.

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, which could then abstract the task models underlying the application. This is the most domain-specific part of the methodology and cannot be easily automated. The literature on information integration proposes various solutions to this problem, notably wrappers and mediator-based schemes.[2]

CASE STUDIES

Implementing the PIPE methodology first requires the identification of "starting points" for personalization at different Web sites. This is a domain-specific consideration. The system developer should model the site schemas by using labeled graphs and modeling semistructured data. The second step is to extract typing rules from each site structure by using 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. The information space represented by the composite program is called a *recommendation space*.

These three preliminary steps are performed offline and only once for any one specific implementation. The final step is the online partial evaluation of the composite program and reconstruction of the original information from the specialized program. I have implemented this approach in two different domains with good results.

Scientific and Engineering Software

The first implementation created Web pages to recommend mathematical and scientific software for scientists and engineers. An effective personalization system in this domain involves at least three different sources:

- In the experiment reported here, I chose the Netlib *software repository* maintained by AT&T Bell Labs, the University of Tennessee, and the Oak Ridge National Laboratory (http://www.netlib.org). Netlib provides access to thousands of pieces of software, much of it organized into Fortran libraries. For example, the Quadpack library provides software routines for numerical quadrature.
- The GAUSS *recommender system* selects algorithms for numerical quadrature.[6] Recommender systems in the computational science domain take a problem description and identify an algorithm that satisfies user-specified performance constraints for errors, time, and so on. The recommendation process is a very complex, domain-specific issue, and is not covered here (for more details, see Ramakrishnan, Houstis, and Rice[6]).
- The Guide to Available Mathematical Software (GAMS) system (http://gams.nist.gov) provides

a *Web-based index* for locating and identifying algorithms for scientific computing. GAMS indexes nearly 10,000 algorithms and employs a tree-structured taxonomy of mathematical and software problems to classify software modules. Figure 4 shows three screen shots of a GAMS session, in which a user selects the H category and proceeds to make further selections, finally arriving at the leaves, where there are several choices of algorithms for a specific problem.

```
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 */ ...
}
```

Figure 5. Final program for the first case study.

PIPE was implemented to personalize recommendations about quadrature software. Without PIPE, scientists typically use GAUSS to obtain a recommendation for a quadrature algorithm. Then they manually navigate through the GAMS taxonomy, starting from the root and seeking the category that contains an implementation for the recommended algorithm. Finally, they browse the Netlib site to download the source code and documentation for the recommendation. Clearly, no one of these resources provides enough information for personalization.

Experimental setup. The Netlib site's schema was first extracted and personalized for the input (Quadpack=1, which provides the algorithms for quadrature). The tree-building algorithm (written in Perl) used the navigation capabilities of the Lynx Web browser. Commonalities encountered in tree building were identified, in part, by Perl hashes that use arrays indexed by page URL.

The mining algorithm did not yield any compression to the original Netlib schema for Quadpack because the site has a strict two-level hierarchy. A simplified portion of the schema obtained from Netlib is shown below:

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

Next, tree building and data mining were conducted for the GAMS Web site, rooted at the H2a node (one-dimensional numerical quadrature). The compressions from mining the GAMS schema were of two main varieties: reductions achieved by factoring common nodes at the preleaf level (typically module sets) and those achieved by mining links that violated the tree taxonomy. Tree building reduced 80 internal nodes in the H2a tree to 74 nodes. Data mining and collapsing multiple roles further reduced the number to 69 nodes. Thus, a compression of 14 percent was observed for the H2a GAMS subtree. At this stage, the composite program is given by

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

where Quadrature_Problem and Automatic_Accuracy are the link labels at the GAMS site. The recommendation rules from GAUSS are already in programmatic form; they take a vector of problem features and performance criteria as input and make a recommendation for an algorithm.[6] Therefore, the three schema (and their respective programs) can be merged, taking into account their inconsistent labeling. 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,[7] which provides procedural, rule-based, object-oriented paradigms for representation. The final program is structured as shown in Figure 5 to model the control flow that is then partially evaluated.

Figure 6 (next page) shows PIPE's end-user interface. The user provides the input problem in self-describing mathematical terms. PIPE first parses the input symbolically to obtain as many features as possible. For instance, in the example shown in

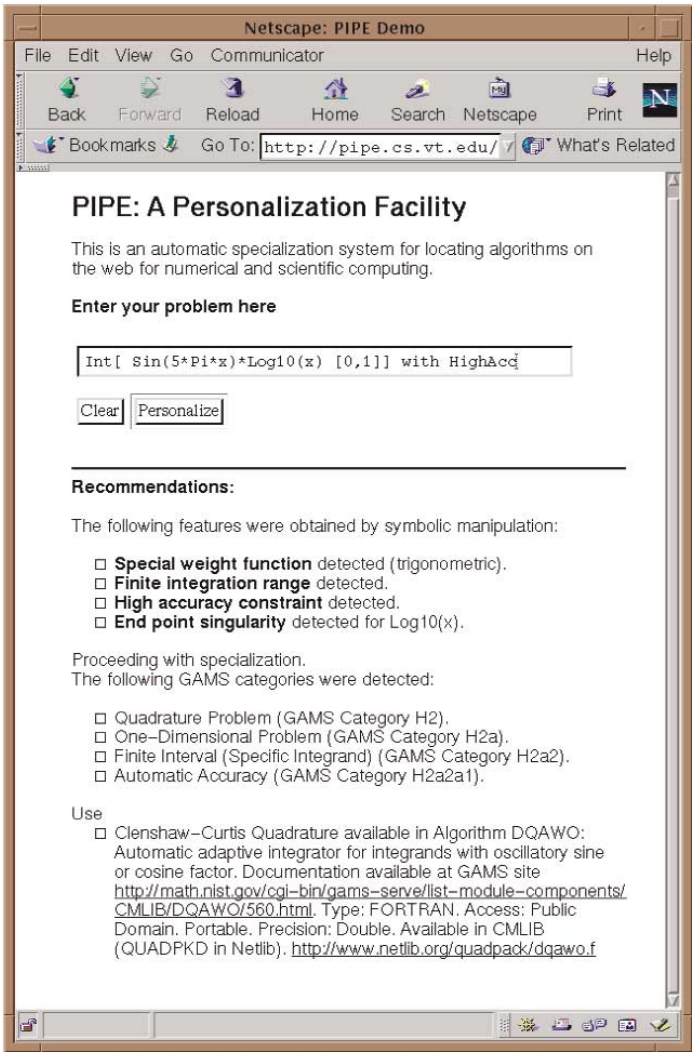

Figure 6. Sample result of PIPE implementation for personalizing content for a numerical quadrature problem. The recommendation includes details of the algorithm (and its implementation), the GAMS site where documentation is available, and the Netlib Web repository where the source code can be downloaded.

Figure 6, simple parsing reveals that the problem is a quadrature problem (indicated by the Int operator), is one dimensional (indicated by the range restriction), and has an oscillatory integrand on a finite domain (indicated by the Sin (...) operator and the range of [0,1], respectively).[6]

Partially evaluating the earlier CLIPS program with this information by setting the appropriate feature values to 1 starts a domino effect of program simplification, removing nearly 95 percent of the original information. The recommendation rules from GAUSS are partially evaluated, in turn navigating the GAMS taxonomy rules and moving toward finding the selected algorithm in Netlib. In this case, the evaluation is actually a complete evaluation because the user has provided enough information to find 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 by the program segment

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

to produce output that includes the algorithm, the GAMS annotation, and the Netlib annotation showing where the algorithm can be downloaded (Figure 6).

The current implementation customizes HTML content by using Perl's text-manipulation capabilities, but programmatic reconstruction of Web pages through systems like WebStrudel[2] can eliminate restructuring when more Web sources or additional rating mechanisms are introduced.

Evaluation. This case study involved a domain important to computational scientists. Scientists and engineers would be experts at building models in their particular domain, but would be novices at understanding the intricacies of the mathematical models and the software systems required to solve them. Therefore, to characterize the results, I used a benchmark set of problems,[6] ran the recommended algorithms to see whether they indeed satisfied the user's constraints, and ensured that the Web links from GAMS and Netlib were properly associated with all the recommendations.

A selection was considered invalid if the algorithm was inappropriate for the given problem or if a wrong page from GAMS/Netlib was indexed. All selections made by this PIPE implementation were valid. Furthermore, the best algorithm (the one that achieved the highest accuracy with the smallest computational cost) was selected for 87 percent of the queries, the second best for 7 percent, and an acceptable choice for 3 percent. A wrong selection was made for the remaining 3 percent of the queries, but these arose from uncertainties in the GAUSS recommender system, not from PIPE's methodology.

In short, 97 percent of PIPE's recommendations in this case study were suitable to the query.

BEV Tourist Information

The Blacksburg Electronic Village (http://www.bev.net) provides a community resource for the New River Valley in southwestern Virginia, USA, where nearly 70 percent of the population actively uses the Internet. Now in its seventh year, BEV offers a wide array of information pertaining to arts, religion, sports, education, tourism, travel, museums, health, and more. The goal of this implementation of PIPE is to direct tourists to appropriate resources in the town of Blacksburg.

The first plan for implementing PIPE in this context was to use two personalization sources: the BEV Web site (and various other pages that it links to) and the Blacksburg Community Directory (an offshoot of the BEV site). However, ambiguities in the descriptions of BEV entities caused problems early on. For example, assume that a user queries for art galleries. Blacksburg has nearly 25 art galleries, but only nine places describe themselves as such. Other such businesses register their organizations with BEV as showrooms, centers, or museums.

Experimental setup. To overcome this problem, I introduced a third personalization source, called Topic, which was a computational distillation of basic keywords and topics from the BEV site. The exact computational algorithm is beyond the scope of this article, but 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. The resulting distilled "index" identifies hidden structures in word usage, thus enabling searches that go beyond simple keyword matching (for an example of this approach, see Booker et al.[8]; for more information on algorithmic and schematic issues involving the organization of topical Web resources, see Terveen, Hill, and Amento[9]).

The Topic source produced rules (such as Microsoft = MSFT) that could be used to model recurrent low-dimensional subspaces in the BEV site.

I applied the same data-mining algorithm to the BEV site that was applied to Netlib and GAMS, but it did not compress the original data as much. One reason for this could be the lack of global "controls" in the construction of Web pages by BEV users and administrators.

Partial evaluation, on the other hand, yielded very effective results, as shown in the sample query in Figure 7. In this case, the evaluation is truly partial, reproducing a collection of subtrees pertaining to coffee shops. The second result depicted in Figure 7 is a false positive, resulting from the association Topic made of the word *coffee* with *café*, although none of the resources identified in Blacksburg To Go are related to coffee shops.

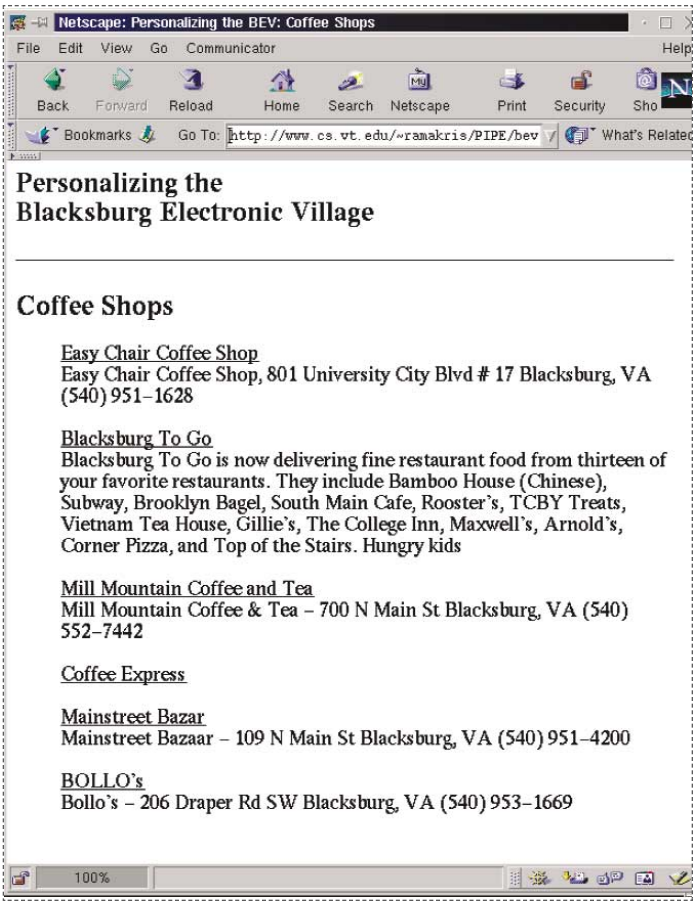

Figure 7. Results of personalizing the Blacksburg Electronic Village for the query "Coffee Shops." Information such as addresses and phone numbers is indexed alongside the nodes as annotations and is displayed when PIPE selects a node to include in the final answer.

Evaluation. This implementation was trickier to evaluate, but the results illustrate the true potential of the PIPE methodology. First, 10 Blacksburg residents were selected randomly and asked to identify 10 queries each that might be pertinent to a visitor (for example, hiking trails). The 10 most frequently cited queries were used to test PIPE. The queries were standardized by eliminating stopwords (terms like "of" and "the" in queries) and by stemming words (for instance, "hikers" was mapped to "hiking").

Then 25 different Blacksburg residents were given the 10 queries and asked to enumerate their answers before PIPE was executed. After it was executed, they were given the personalization results and asked if

they would like to change their original answers or if they thought the results were deficient in any respect. For each query, each user voted on the mismatch between the personalization results and any expected answers, using a scale of 1 to 5 (with 1 indicating complete satisfaction with the results).

The hypotheses for all 10 queries were accepted at the 95 percent level.

Among the 250 votes cast, all but 32 were in the 1–2 range, and the 32 votes were 3s. For each query, I then conducted a distribution-free test (Kruskal-Wallis[10]) of the hypothesis that the results were unanimous (versus the alternative that they were not all equal). A "unanimous" result indicates statistical agreement among the 25 residents about the outcome. The hypotheses for all 10 queries were accepted at the 95 percent level, indicating conclusively that the results were very close to the expected answers.

The 32 votes cast for "3" were spread over seven people who were less "effusive" than others with their ratings. Handling the effusivity factor in user-supplied ratings and evaluations is a problem not unfamiliar to recommender systems research, as noted by Aggarwal et al.[5] One way to overcome this problem is to replace absolute ranks with relative ranks (or by using linear transformations) so that they could be captured by certain two-way statistical tests (such as the Friedman, Kendall, and Babington-Smith test[10]).

The results for one query—namely, trails—had consistently lower ratings from nearly all 25 resident participants. The results failed to reproduce two of the most popular trails in the vicinity. Not surprisingly, these trails were not mentioned on any Web pages in the considered collection.

The PIPE results fared well when compared with the traditional Web search facilities available in the BEV site. For example, at the time of this writing, the standard BEV search engine produced no results for the query "coffee shops" (or "coffee").

DISCUSSION OF RESULTS

The effectiveness of the PIPE methodology relies on three factors—systematic design of the personalization system, accurate modeling of Web resources, and consistent methodologies for evaluation. We discuss PIPE under each of these categories.

First, PIPE integrates the design of personalization systems with the task models that underlie the assumed scenarios of interaction. This property lets the designer view the personalization system as a composition of individual *subsystems* (to use a programming metaphor). PIPE is therefore restricted to those domains that are most amenable to such decomposition and analysis techniques. It is not a good fit for more amorphous domains, such as social networks in an organizational setting.

Second, the implementations described here require the link labels to represent navigational choices and the values for such labels (program variables) to be ascertainable from user input. This was easy to achieve in both case studies because the GAMS/GAUSS and BEV sites serve as ontologies to help guide the personalization process. In the absence of such ontologies, personalization would be only as effective as the ease with which link labels are supplied by the user or extracted from user input. For example, if a medical informatics site is organized according to scientific names of diseases and ailments, personalizing for "headaches" would require a parallel ontology or cross-index that maps everyday words into scientific nomenclature.

In addition, data-mining techniques that extract structures (DTDs) at the level of a single page can be incorporated in the PIPE framework (for a modeling method for a single page, see Garofalakis et al.[11]). Moreover, the loops and complex control structures that can make partial evaluation a costly computation are seldom present in the kinds of applications considered here. The PIPE data-mining process factors out links that point back to higher levels of the hierarchy, thus avoiding code blowups. Further studies are nevertheless required to characterize the scale-up for larger domains. Also, PIPE's use of `else if` statements supports both disjunctions and conjunctions in the personalization queries. In typical Web sites, the links are either mutually exclusive (as in Example 1 and the GAMS case study) or inclusive (as in the BEV case study). Automating support for this aspect of personalization in a Web crawler requires more study. Metadata or explicit user direction offer possible alternatives.

Finally, the BEV case study shows that it is acceptable (even desirable) to be less strict in variable assignments for certain domains, even though it yields more false positives. In the GAMS case study, more stringent demands are made of personalization. These factors hence directly affect the design and efficacy of the end system.

CONCLUSION

The role of methodologies like PIPE in the future of the Internet is captured well in an analogy drawn by Rus and Subramanian.[12] I conclude here with an annotated quotation of it:

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

ACKNOWLEDGMENTS

I would like to thank Saverio Perugini, Akash Rai, and Priya Lakshminarayanan for helpful discussions on the PIPE methodology. B. Arul Pandian assisted with the implementation of the BEV case study, and Mary Beth Rosson provided guidance on conducting its evaluation. I also thank the nearly 40 volunteers who helped in the evaluation of the BEV study. Feedback from several anonymous referees helped clarify the presentation and improve the article.

REFERENCES

1. S. Lawrence and C. Lee Giles, "Searching the World Wide Web," *Science*, vol. 280, no. 5,360, 1998, pp. 98-100.
2. D. Florescu, A. Levy, and A. Mendelzon, "Database Techniques for the World Wide Web: A Survey," *SIGMOD Record*, vol. 27, no. 3, Sept. 1998, pp. 59-74.
3. N.D. Jones, "An Introduction to Partial Evaluation," *ACM Computing Surveys*, vol. 28, no. 3, Sept. 1996, pp. 480-503.
4. S. Nestorov, S. Abiteboul, and R. Motwani, "Extracting Schema from Semistructured Data," *Proc. ACM Int'l Conf. Management of Data* (SIGMOD), ACM Press, New York, 1998, pp. 295-306.
5. C.C. Aggarwal et al., "Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering," *Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining*, ACM Press, New York, 1999, pp. 201-212.
6. N. Ramakrishnan, E.N. Houstis, and J.R. Rice, "Recommender Systems for Problem Solving Environments," Tech. Report WS-98-08 (*Working Notes of the AAAI-98 Workshop on Recommender Systems*), H. Kautz, ed., AAAI/MIT Press, Menlo Park, Calif., 1998, pp. 91-95.
7. J.C. Giarratano, *CLIPS User's Guide*, version 5.1, NASA, Houston, Texas, 1991.
8. A. Booker et al., "Visualizing Text Data Sets," *IEEE Computing in Science and Eng.*, vol. 1, no. 4, July/Aug. 1999, pp. 26-34.
9. L. Terveen, W. Hill, and B. Amento, "Constructing, Organizing, and Visualizing Collections of Topically Related Web Resources," *ACM Trans. Computer-Human Interaction*, vol. 6, no. 1, Mar. 1999, pp. 67-94.
10. M. Hollander and D.A. Wolfe, *Nonparametric Statistical Methods*, John Wiley & Sons, New York, 1973.
11. M. Garofalakis et al., "XTRACT: A System for Extracting Document Type Descriptors from XML Documents," *Proc. ACM Int'l Conf. Management of Data* (SIGMOD), ACM Press, New York, 2000, pp. 165-176.
12. D. Rus and D. Subramanian, "Customizing Information Capture and Access," *ACM Trans. Information Systems*, vol. 15, no. 1, Jan. 1997, pp. 67-101.

Naren Ramakrishnan is an assistant professor of computer science at Virginia Tech. His research interests include recommender systems, problem-solving environments, personalization, and data mining. He obtained a PhD in computer sciences from Purdue University in August 1997. He is a member of the ACM, the IEEE Computer Society, and AAAI.

Readers may contact Ramakrishnan via e-mail at naren@cs.vt.edu.