Using Memex to archive and mine community Web browsing experience

Soumen Chakrabarti , Sandeep Srivastava , Mallela Subramanyam , Mitul Tiwari

Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Powai, Mumbai 400076, India

Abstract

Keyword indices, topic directories, and link-based rankings are used to search and structure the rapidly growing Web today. Surprisingly little use is made of years of browsing experience of millions of people. Indeed, this information is routinely discarded by browsers. Even deliberate bookmarks are stored passively, in browser-dependent formats; this separates them from the dominant world of HTML hypermedia, even if their owners were willing to share them. All this goes against Vannevar Bush's dream of the *Memex*: an enhanced supplement to personal and community memory. We present the beginnings of a Memex for the Web. Memex blurs the artificial distinction between browsing history and deliberate bookmarks. The resulting glut of data is analyzed in a number of ways. It is indexed not only by keywords but also according to the user's view of *topics*; this lets the user recall topic-based browsing contexts by asking questions like 'What trails was I following when I was last surfing about *classical music*?' and 'What are some popular pages related to my recent trail regarding *cycling*?' Memex is a browser assistant that performs these functions. We envisage that Memex will be shared by a community of surfers with overlapping interests; in that context, the meaning and ramifications of topical trails may be decided by not one but many surfers. We present a novel formulation of the *community taxonomy synthesis problem*, algorithms, and experimental results. We also recommend uniform APIs which will help managing advanced interactions with the browser.

Keywords: Community memory; Collaborative taxonomy synthesis; Browsing assistant

1. Introduction

What are the basic sources of information available from the Web? First-generation search engines (**Alta Vista**) initially exploited the tokens on each page as the sole source of information. Second-generation search engines (**Google** , **Clever**) use hyperlink structure as well.

Popular search sites answer tens of millions of queries per day. We speculate that the total number of browser clicks per day is at least three orders of magnitude larger. This third source of information has a scale that dwarfs the Web itself, and yet is perhaps as rich and valuable as text and links. Some centralized services (**Alexa** , **Third Voice**) distribute a browser plug-in to monitor clicks and add page reviews. Others (**Hotbot** , **WiseWire**) monitor clicks limited within their site to improve relevance ranking or dynamically reorganize content.

Nevertheless, most of the information latent in years of clicking by millions of surfers is carelessly discarded by browsers, unless they are deliberately bookmarked. Even bookmarks are stored passively, in browser-dependent formats; this separates them from the dominant world of HTML hypermedia, even if their owners were willing to share them.

In 1945, Vannevar Bush dreamt of *Memex*: an enhanced, intimate supplement to personal and community memory [5]. Memex would unobtrusively archive, analyze, and index our complete experience, covering books, movies, music, conversation, etc. Since then this theme of a 'living' hypermedium into which we 'weave ourselves' has been emphasized often, e.g., by **Douglas Engelbart** and **Ted Nelson** , and of late by **Tim Berners-Lee** .

Assisted by a Memex for the Web, a surfer can ask the following questions:

- What was the URL I visited six months back regarding compiler optimization at Rice University?
- What was the Web neighborhood I was surfing the last time I was looking for resources on classical music?
- Are their any popular sites, related to my (Web) experience on classical music, that have appeared in the last six months?
- How is my ISP bill divided into access for work, travel, news, hobby and entertainment?
- What are the major topics relevant to my workplace? Where and how do I fit into that map? How does my bookmark folder structure that map on to my organization?
- In a hierarchy of organizations (by region, say), who are the people who share my interest in recreational cycling most closely and are not likely to be computer professionals?

We propose an architecture of a Memex for the Web which can answer the above questions. Memex is a large project involving hypertext data mining, browser plug-in and applet design, servlets and associated distributed database architecture, and user interfaces. In this paper we will give an overview of the major features of Memex, together with the architecture and design decisions.

We have validated the design using a prototype implementation. We will give examples of some of the features using screen shots and measurements on our prototype. The **Memex** service will be made publicly accessible.

1.1. Architecture overview

The main function of Memex are personalization and community formation. Personalization in our context is different from the myriad *MyXYZ* portal sites, which involve manual selection of topic directories from a catalog. In our case, personalization is for both individuals and groups, and is done automatically by mining archives of community browsing experience, thus nurturing communities with clustered topical interest.

A major issue is the level at which communities are defined and analyzed. On one extreme, a central service may be provided. This would be a poor solution, not only owing to the formidable scale of operation, but also the diversity of the population and their interests. At the other extreme, individual browser-level plug-ins have limited access to behavior of anyone other than the user.

We therefore believe that the most successful community formation and personalization tool is at an intermediate level, such as an organization or geographical divisions of an ISP. Trust is not as major an issue at this level as on the Web at large; after all, most of us (have to) trust the administrator of our proxy cache and/or firewall machine. At the same time, the problems of sparse, noisy data from just one user is ameliorated somewhat by a larger user population.

The present release of Memex consists of client and server-side code, communicating via the applet–servlet protocol. The server-side code needs Apache with servlet support and a relational database, such as Oracle or IBM Universal Database installed. We avoid CGI-style solutions because the state exchanged between client and server is rather complex compared to HTML form data, and extended sessions with state are often useful. We anticipate that each logical organization or group will install a server.

We prefer not to assign the role of the Memex server to the proxy or firewall computer. The latter are already heavily loaded with their primary tasks. In the database world, software is separately architected for day-to-day transaction processing (OLTP) as against data analysis tools (warehousing, OLAP and mining). We believe a similar division of labor will be successful for Memex as well.

Unlike earlier browser plug-ins, for maximum portability, the client-side code is a JDK1.2-compliant applet. The applet polls the browser's location and sends the URL and other processed data through an encrypted connection to the server, where more analysis takes place. Java cannot steer us away from platform-dependent code, especially owing to the diverse browser security APIs. The applet runs on Netscape Communicator 4.5+ and Internet Explorer 4+. (A HotJava port is being planned.)

We use Javascript on UNIX and Dynamic Data Exchange (DDE) on Windows to interact with the browser, unlike some other systems like Web Browser Intelligence (WBI) [3] or Third Voice which install a client-side proxy. Unlike PowerBookmarks, we do not use a server-side proxy for monitoring clicks. Memex monitors the network performance and reassigns page fetching and analysis jobs dynamically between client and server. We believe this is an essential feature because a client may wish to connect to a geographically remote server which has a poorer Internet connection, or vice versa.

1.2. Features for archiving and analysis

Memex provides a platform for capturing hypertext and meta-data based on content as well as surfing events, in a structured manner. This lets us build a collection of interrelated and cooperative programs that mine the Memex traces. We discuss the major functions below.

Bookmarks ***are*** *history*. Browsers make an unnecessary distinction between *history* and *bookmarks*. The prevalent view is that bookmarks are valuable links deliberately archived by the user, whereas history is the list of all visited pages, long and mostly useless, therefore suitable for purging now and then. However, the rapidly increasing volume-to-cost ratio of disks makes it unnecessary to discard *anything* from our personal browsing trails, if only the resulting mass of data can be organized and analyzed usefully and effectively (features browsers do not provide).

Note that Memex does not directly undertake to archive the actual contents of the surfed pages. Our design should make it quite simple to interface Memex to other content and usage archives such as **Alexa Internet** or the **Internet Archive** . This will give additional benefits like never losing an obsolete page.

Learning folder structure. We claim that users create bookmarks sparingly only because of the complexity of creating and maintaining structure in their bookmark collection. If the user has a collection of topically coherent *folders*, Memex will learn to guess, using a hypertext topic learning algorithm, the best placement of new history entries. Thus the entire browsing experience of the user will be structured into topics [15].

Proposing folder structure. Interests and topics evolve with time. A narrow topic may grow until it becomes diffused, warranting a finer subdivision. Memex can propose, based on document similarity, a subdivision of pages belonging to a folder, which can lead to a better (re-)organization of the subtopics associated with that folder.

Trails and topical contexts. An extremely common question, unanswered by current browsers, is of the form: 'Where was I when I was last browsing about classical music?' Studies have shown that visiting Web pages is most strongly visualized using spatial metaphors: your context is 'where you are' and where you are able to go next [22].

Once Memex analyzes history as described above, it becomes possible to reconstruct the recent behavior of the user w.r.t. any topic. This is a valuable feature to return to a goal-directed browsing session and resume with context.

Synthesizing a taxonomy for the community.

[15] It has been pointed out that users may create folders which are not dependent on the contents of pages, but other features. If these features cannot be exposed to our learning programs, there is little hope for the learners. But content-based classification is by far the most common case we have seen.

Memex is best distinguished from bookmark archival services by its ability to synthesize a topic taxonomy from the browsing and bookmarking habits of a user community. This can be used in turn for better folder management and additional discovery of relevant resources from the Web.

1.3. Related work

1.3.1. Bookmark storage services

Internet start-ups have been quick to discover the annoyance of surfers maintaining multiple bookmark files and the opportunity of a central, networked bookmark server. We can list several sites which, using Javascript or a plug-in, import existing Netscape or Explorer bookmarks and thereafter let the surfer visit their Website and maintain it using CGI and Javascript: **Yahoo Companion**, **YaBoo**, **Baboo**, **Bookmark Tracker** and **Backflip** are some examples. Some services like Third Voice enable surfers to attach public or private annotations to any page they visit.

Netscape itself has a 'roaming access' feature which is a convenient utility that can 'FTP' (actually LDAP and HTTP are used) the user's configuration, history and bookmarks to the client at the beginning of a session and 'FTP' them back to the server at the end of the session. No content-based analysis is involved.

Purple Yogi will provide a private client-side browser assistant which will monitor browsing and searching actions and use this data to refine future search results. The technology is not published and the system is not available at the time of writing. No community-level mining seems involved.

Our work is closest in spirit to three well-known and similar systems, VistaBar, PowerBookmarks and the Bookmark Organizer.

VistaBar [23] is a browsing assistant application (later integrated into **Alta Vista Discovery**) that lives on the Microsoft Windows desktop and attaches to the active browser using the DDE (Dynamic Data Exchange) interface. It provides bookmarking, annotation, indexing, find-similar, find-referrer, and classification into a shared Yahoo topic taxonomy. We have been greatly influenced by VistaBar's technique of interaction with the browser.

PowerBookmarks [20] is a semi-structured database application for archiving and searching bookmark files. By visiting the PowerBookmarks site, the user gets 'a browser within the browser' which has controls for navigation and bookmarking that are provided as CGI forms by the PowerBookmarks site. PowerBookmarks uses Yahoo! for classifying the bookmarks of all users. In contrast, Memex preserves each user's view of their topic space, and reconciles these diverse views at the community level. Furthermore, PowerBookmarks does not use hyperlink information for classification or for synthesizing communities.

The Bookmark Organizer [21] is a useful client-side solution for personal organization via clustering and classification, but does not provide community-level themes or topical surfing contexts.

1.3.2. Mapping tools

Several visualization tools have been designed recently that explore a limited radius neighborhood and draw clickable graphs. These are often used for site maintenance and elimination of dead links. Mapuccino and Fetuccino from IBM Haifa are well known examples [4,15]. There is also a mature research literature on *graph drawing*: embedding graphs in the plane or in 3-D space so as to enhance some desirable properties such as reduced edge crossing, hierarchy separation, etc. Spring and Clan Graph Decomposition are some of the well-known techniques [17,27]. Our context viewer could benefit from better drawing techniques.

1.3.3. Supervised and semi-supervised learning

Supervised learning of document topics has been researched extensively in recent years [1,2,6,10,14, 19]. However, less is known about how to best integrate hypertextual features, and specifically how to exploit diverse patterns of individual bookmarking as new features. Unsupervised clustering is a classical field for structured data sources [16]. The clustering problem becomes harder in high-dimensional spaces such as text. Some well-known systems for clustering text are HyPursuit [28], Grouper [29] and Scatter-Gather [11]. Clustering is also related to structure discovery [25].

1.3.4. Resource discovery

Content- and hyperlink-based Web resource discovery has been extensively researched since 1996 [7,9,18]. In most such systems, discovery is deliberate: either a keyword-based query has to be entered (as in the HITS and Clever topic distillation systems) or topic nodes in a taxonomy have to be explicitly populated with examples and marked for exploration (as in the **Focused Crawler**). For ad-hoc similarity search, Netscape's browser provides a 'find similar' button. However, a single page may not provide sufficient context information to discover important, related resources. Dean and Henzinger have provided an enhanced 'find-similar' algorithm [12].

Memex is well-suited to drive the resource discovery process. By synthesizing a topic taxonomy specific to the interest of a user community, Memex provides input for topic-based resource discovery algorithms [9].

1.4. Organization of this paper

The paper is organized as follows. Section 2 discusses the architecture and implementation details, design decisions, etc. It also introduces the look-and-feel of the Memex client to the end-user. Section 3 elaborates on the back-end mining techniques used to provide the client features. These involve understanding statistical relations between terms, documents and folders. Section 4 concludes the paper, with a wish-list for uniform and robust APIs for applets to interact with browsers, and a summary of ongoing and future work.

2. System architecture

Using the Memex client (Fig. 1) should not need any software installs or browser updates. In view of secure firewalls, proxies, and ISPs' restrictions on browser setups, the client should communicate with the server over HTTP. The data transferred should be encrypted if desired to preserve privacy.

On the server side (Fig. 1), the system should be robust and scalable. It is important that the server recovers from network and programming errors quickly, even if it has to discard one or two client events. For both client and server we also want a rapid prototyping environment. It should be possi-

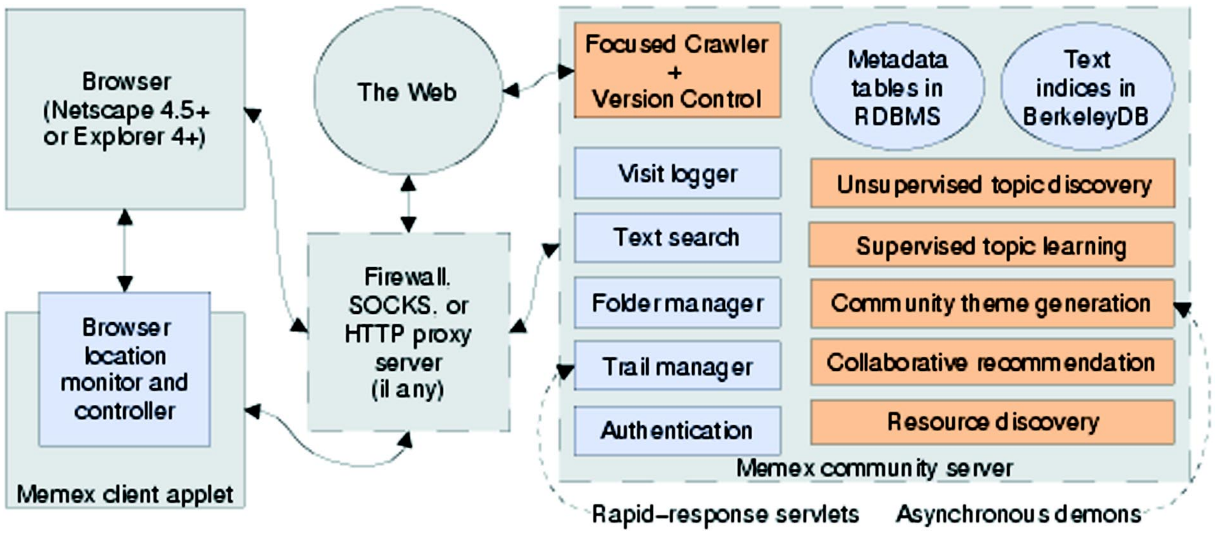
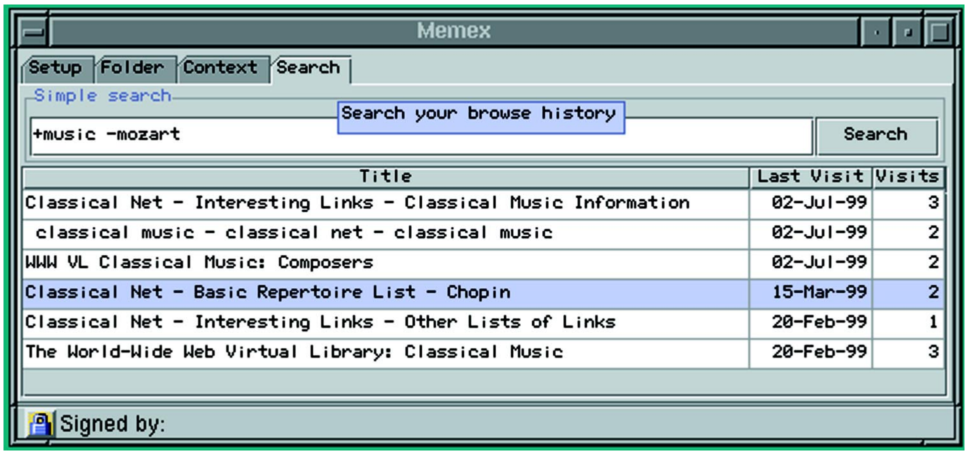

Browser (Netscape 4.5+ or Explorer 4+)
The Web
Focused Crawler + Version Control
Metadata tables in RDBMS
Text indices in BerkeleyDB
Visit logger
Unsupervised topic discovery
Text search
Supervised topic learning
Folder manager
Community theme generation
Browser location monitor and controller
Firewall, SOCKS, or HTTP proxy server (if any)
Trail manager
Collaborative recommendation
Authentication
Resource discovery
Memex community server
Memex client applet
Rapid-response servlets
Asynchronous demons

Fig. 1. Block diagram of the Memex system.



+music -mozart

Title	Last Visit	Visits
Classical Net - Interesting Links - Classical Music Information	02-Jul-99	3
classical music - classical net - classical music	02-Jul-99	2
WWW VL Classical Music: Composers	02-Jul-99	2
Classical Net - Basic Repertoire List - Chopin	15-Mar-99	2
Classical Net - Interesting Links - Other Lists of Links	20-Feb-99	1
The World-Wide Web Virtual Library: Classical Music	20-Feb-99	3

Fig. 2. The Search tab enables boolean and free text search over all archived surfing history. Results can be sorted by various columns.

ble to distribute updates and new features effortlessly to users.

The Memex client is an applet that interacts with the browser and connects to the Memex server over HTTP. The server consists of servlets that perform various archiving and mining functions autonomously or triggered by client action. An important aspect of the engineering is the division of labor between the client and the server. Another interesting aspect of the server architecture is the loose data consistency model supported across text and user metadata in a relational database and text indices stored in lightweight storage managers. More details are given in Section 2.2.

2.1. Client-side design

The Memex client is a signed Java applet which can be invoked from a Memex server Website, or using a `.html` and a `.jar` or `.cab` file downloaded to local disk (a faster option, but one has to manually check for new versions and updates).

The main challenge with client-side system design is to pack in a host of useful features into small and precious screen real estate. Many users find browser panels already too cluttered since the days of Lynx and Mosaic. After quite some dummy trials, we came up with about a 500-by-400 pixel panel with several function tabs.

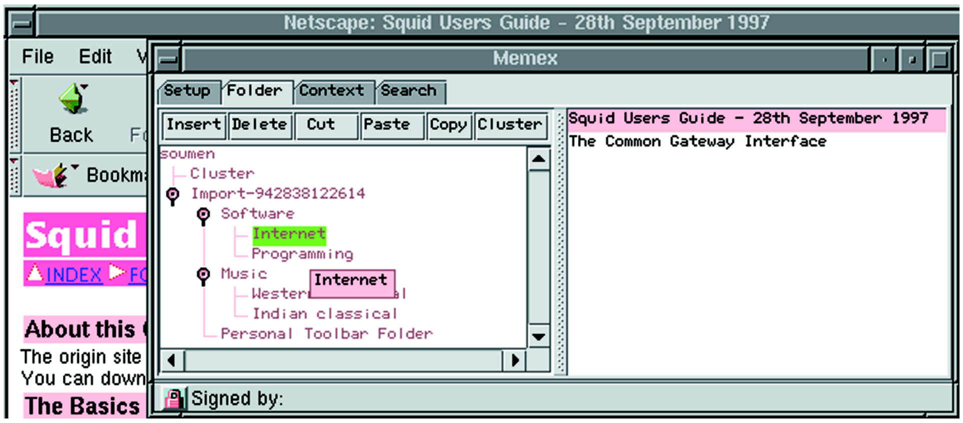
2.1.1. Search tab

Memex shows a three-way privacy choice: the user can choose not to archive clicks, to archive them as his/her private data, or share it with others. If and when the user so permits, page visits are reported to the server. The client and server cooperate to populate server-side tables with node, link, and visit meta-data, as well as a full-text index. Currently we handle only HTML, but we plan to add PS and PDF, which will be very useful for users in the academic community. The Search tab, shown in Fig. 2, enables boolean and free text search over the full text of surfing history. The responses can be sorted in a number of ways and clicking on a response takes the browser to that page.

2.1.2. Folder tab

Each Memex user has a personal folder/topic space shown in the Folder tab in Fig. 3. Most users will already have a collection of links in a browser-specific format. Memex can import these and attach it to their personal space as shown (Import-*timestamp*), it can also export to the popular formats.

Folders are key to characterizing and differentiating between topics of interest to an individual and to a community. Yet, some users will have no folder structure in their bookmarks, and others will keep accumulating bookmarks which are not placed in any folder (by default, in the root folder which characterizes ‘any topic’).



Squid Users Guide - 28th September 1997
The Common Gateway Interface

Fig. 3. Each user has a personal folder/topic space. The user can import existing browser-specific bookmark folders into Memex, which by default go into the '*log-in*/Import-*timestamp*' directory. Folders and their contents can be edited freely. The special 'Cluster' folder is explained later in Section 3.3.2.

Thus for both initial and continual use, surfers need two functions: *discovering* a topic taxonomy and *maintaining* it. If the user needs suggestions for (re-)organizing a set of URLs, s/he copies them into the special folder marked 'Cluster' in Fig. 3 and clicks the Cluster button. The user can then evaluate the result and cut and paste acceptable folders into the regular space.

Users will not manually classify most of their history. Memex will do this lazily in the background, populating the right panel with URLs whose folder membership has been 'guessed,' visually flagged. The user can then correct mistakes in folder assignment by cutting and pasting [26]. This constitutes valuable training by the user, and helps Memex more accurately reflect the user's notion of topics.

2.1.3. Context tab

Studies have shown that visiting Web pages is best expressed using spatial metaphors: your context is 'where you are' and 'where you are able to go' next [22]. Users surf on many topics with diverse priorities. Because browsers have only a transient context (one-dimensional history list), surfers frequently lose context when browsing about a topic after a time lapse.

Once Memex has guessed the folders for unclassified history pages, providing a topical context becomes possible. Figs. 4 and 5 show how selecting a folder lets the user resume their browsing on a topic 'from where they left off'. A user-configurable parameter determines how many of the most recent nodes are shown.

As the user continues on the trail, new nodes get added to the graph. It may not be possible to classify these in real time. New nodes will be added to the display, and if/when the server responds with a folder assignment, nodes that do not belong to the currently selected folder may be dropped from the display.

Apart from restriction by topic, it is also possible to restrict context by site. Web surfing often involves fanning out and backing up. Mapping a small neighborhood of the current page being viewed lets the user perform these maneuvers with less effort. We also plan to integrate backlinks into the Context tab [8].

2.2. Server-side design

System design on the server side is guided by the concerns of robustness and scalability. We used a 3-tier approach (applet communicates with servlets which use JDBC to connect to relational databases) for two main reasons.

HTTP tunneling. Firewalls and proxies will often forbid non-HTTP traffic. Although several data-

[26] We cannot use drag-and-drop in many places because the full set of Java Foundation Classes functionality is not available from most browsers' JVM.

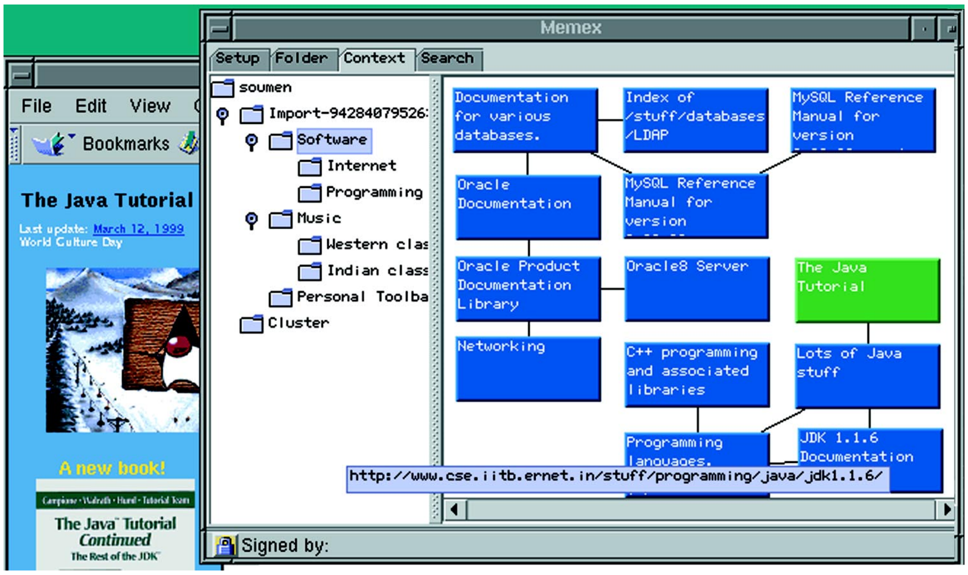


Fig. 4. The Trail tab shows a read-only view of the user's current folder structure. All pages visited by the user are (lazily) classified by Memex into the folders. When the user selects a folder, Memex shows (a configurable number of the) most recently browsed pages which belong to the selected topic, reminding the user of the latest topical context. In the screen-shot above, the chosen folder is /*Software*.

base and middleware vendors are starting to provide HTTP tunneling support with JDBC, we wanted a lightweight solution for simplicity and efficiency.

Authentication and access control. Relational databases provide table- and view-level access control to users having log-in IDs on the host machine. In our case, access control information is itself in the tables.

Fig. 6 shows the basic entities represented in the Memex server: users, Web pages, visits, links, and folders. The database design needs to address the issues of efficient history archival taking care of user preferences, authentication, and storage of semi-structured HTML documents.

Server state is managed by two storage mechanisms: a relational database (RDBMS) such as Oracle or DB2 for managing meta-data about pages, links, users, and topics, and a lightweight Berkeley DB storage manager to support fine-grained term-level data analysis for clustering, classification, and text search. Storing term-level statistics in an RDBMS would have overwhelming space and time overheads.

An interesting aspect of the Memex architecture is the division of labor between the RDBMS and Berkeley DB. Planning the architecture was made non-trivial by the need for asynchronous action from diverse modules. There are some servlet events that must be guaranteed immediate processing. Typically, these are generated by a user visiting a page, or deliberately updating the folder structure. With many users concurrently using Memex, the server cannot analyze all visited pages, or update mined results, in real time. Background demons continually fetch pages, index them, and analyze them w.r.t. topics and folders. The data accesses made by these demons have to be carefully coordinated. This would not be a problem with the RDBMS alone, but maintaining some form of coherence between the meta-data in the RDBMS and several text-related indices in Berkeley DB required us to implement a loosely consistent versioning system on top of the RDBMS, with a single producer (crawler) and several consumers (indexer and statistical analyzers). Fig. 1 shows a block diagram of the system.

3. Mining the Memex archive

In this section we describe the core algorithms that run on the databases maintained by the Memex front-end. We also report on our experiences with

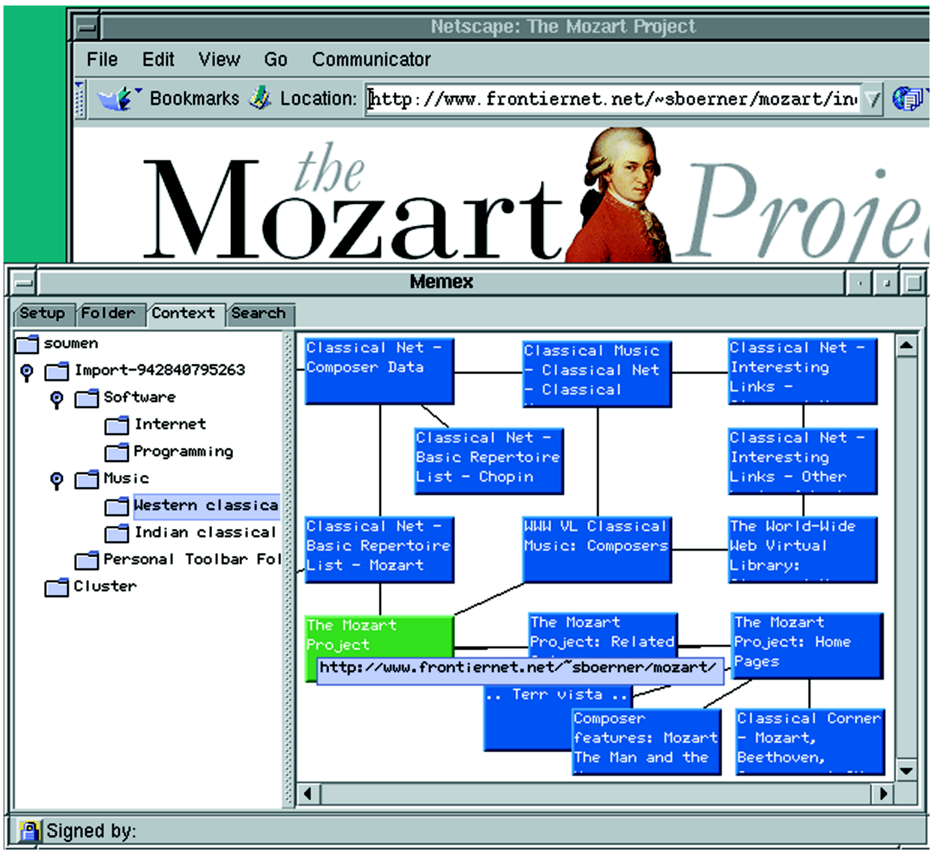

Netscape: The Mozart Project
File Edit View Go Communicator
Bookmarks Location: http://www.frontiernet.net/~sboerner/mozart/in
the Mozart Proje
Memex
Setup Folder Context Search
soumen
Import-942840795263
Software
Internet
Programming
Music
Western classica
Indian classical
Personal Toolbar Fol
Cluster
Classical Net - Composer Data
Classical Music - Classical Net - Classical
Classical Net - Interesting Links -
Classical Net - Basic Repertoire List - Chopin
Classical Net - Interesting Links - Other
Classical Net - Basic Repertoire List - Mozart
WWW VL Classical Music: Composers
The World-Wide Web Virtual Library:
The Mozart Project
The Mozart Project: Related
The Mozart Project: Home Pages
http://www.frontiernet.net/~sboerner/mozart/
.. Terr vista ..
Composer features: Mozart The Man and the
Classical Corner - Mozart, Beethoven,
Signed by:

Fig. 5. Changing the selection to */Music/Western Classical* instantly gives the user a new topical context. The current/latest page is highlighted.

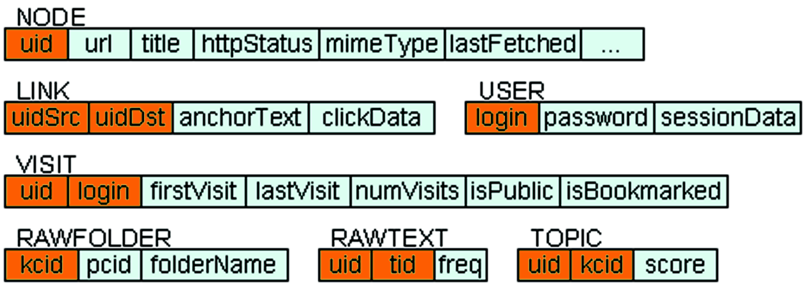

NODE
uid url title httpStatus mimeType lastFetched ...
LINK
uidSrc uidDst anchorText clickData
USER
login password sessionData
VISIT
uid login firstVisit lastVisit numVisits isPublic isBookmarked
RAWFOLDER
kcid pcid folderName
RAWTEXT
uid tid freq
TOPIC
uid kcid score

Fig. 6. The schema for server-side data. RAWFOLDER, TOPIC and RAWTEXT are processed into text indices for search, clustering and classification. Primary keys are dark. RAWTEXT is stored as a gzipped binary object for space efficiency. Inverted indices and folder are stored in Berkeley DB.

some of these algorithms. Most of our client features concern the analysis of relations between people, pages and folders (topics). (Although we provide a text-search capability over the history, we regard this as a standard feature.)

3.1. Experimental test-bed

For continuity, we shall inline our experimental results into the following subsections as we formulate problems and design solutions. For our experiments we obtained bookmark files from 23 Web surfers. They are Web-savvy but not computer scientists. After converting their Netscape and Explorer bookmark to a common format and throwing away default folders inserted by those browsers, we were left with 223 folders. All but two folders were at the first level; for simplicity we flattened those. These folders together pointed to 1693 URLs, of which 1424 were distinct.

Memex server-side code has been implemented in Java and JDBC. It has been tested on a 450 MHz Pentium 2 running RedHat Linux 5.2 with IBM Universal Database 5.2. Page visit archiving is real-time, but classification and taxonomy synthesis take several minutes. These are performed lazily by demons once or twice a day. In this paper we will be concerned more with the quality of results than running times.

3.2. Data models

As mentioned in our description of the Memex architecture (Section 2), the main entities represented in our analyses are Web documents, denoted d; hyperlinks, denoted (d_1, d_2); people, denoted p; and folders, also called topics or classes, denoted f. Each person has an associated universe of all Web documents s/he has ever seen, called the history $H(p)$. These documents are connected by hyperlinks, some of which have been traversed by the user.

Each person also owns a tree-structured hierarchy of folders (which may be just the root node). A typical user will start by importing their folder structure from their browser. Because Netscape bookmarks are stored in a flat file, a folder imported from Netscape will have an ordering on its children. Internet Explorer uses directories for folders, so no such ordering exists. The folder hierarchy is not fixed for all time, but our discussion below assumes a snapshot.

Folders reflect the owner's notion of topics and their refinement. Some of the documents in the history are 'bookmarked'; these are denoted $B(p)$. Bookmarks belonging to a particular folder are called $B(p, f)$. This means that a hyperlink to such a document is placed in some folder by a person. A document can be bookmarked by many people, but a person can only place a document in one folder. (In ongoing work we are relaxing this restriction.)

3.3. Learning and discovering topics

In supervised learning, the goal is to use the association between $B(p)$ and the folders to learn to assign documents in $H(p)\backslash B(p)$ to the user's predefined folders. In unsupervised topic discovery, Memex inputs a set of documents from $H(p)$ and proposes a tree-structured topic hierarchy that clusters the documents based on their contents.

3.3.1. Supervised learning

Statistical model-based learning [13,24] has been one of the most effective methods for learning document topics. In its simplest form, our text classifier regards a document d as a bag or multi-set of terms: the term t occurs $n(d,t)$ times and the document has a total length of $n(d) = \sum_t n(d,t)$. We do not consider multi-term phrases as features (yet).

Each folder f has an associated parameter $\theta(f,t)$ for each term t over a suitably large universe T of terms. Roughly, $\theta(f,t)$ is the rate at which term t occurs in documents belonging to folder f, and can be estimated from $B(p, f)$. One can regard $\theta(f,t)$ as the probabilities on the face of a die with T-faces, where $\sum_t \theta(f,t) = 1$. A document d that is known to belong to folder f (in statistical terms, be generated from folder f) is written as follows: first an arbitrary length $n(d)$ is decided, then the above die is tossed so many times, and the terms corresponding to the faces that come up written down. Thus the probability of generation of the document is:

$$\Pr(d|f) = \binom{n(d)}{\{n(d,t)\}} \prod_{t \in d} \theta(f,t)^{n(d,t)}. \qquad (1)$$

Using Bayes rule, we can conversely find the probability $\Pr(f|d)$ that a document was generated by the distribution associated with a given folder.

It turns out that modeling the text alone does not give sufficient classification accuracy, because people tend to bookmark pages with little textual and much graphical content, as well as many links. In Section 3.3.3 we will discuss how to improve the accuracy of folder assignment using co-placement of URLs in folders. The Memex engine continually scans the `NODE` table in search of new classifications to do, and updates the `TOPIC` table.

3.3.2. Unsupervised learning

Topics will change with time, and users will reorganize their folder structure. The most common requirement for reorganization is refinement of a topic: the user's initial notion of a topic may be broad and shallow, but with time, the user collects a large number of links under that topic, which now needs to be refined into subtopics. This is a classic case of unsupervised learning or clustering [16].

An extension of the vector space model [26], similar to the bag-of-words model is used for clustering documents. In the basic vector space model, each document d is represented as a point in multi-dimensional Euclidean space; each dimension is a term t, and the coordinate in that dimension is $n(d,t)$[27]. Documents are normalized to have L_2 norm equal to one.

The similarity $s(d_1, d_2)$ between unit-length documents d_1 and d_2 is the cosine of the angle between their vectors, i.e., the inner product $\langle d_1, d_2 \rangle$. Since the role of clustering is to find large similar sets of documents, we use the definition of self-similarity of a set Γ from the Scatter-Gather system [11]:

$$s(\Gamma) = \frac{1}{|\Gamma|(|\Gamma|-1)} \sum_{d_1,d_2 \in \Gamma, d_1 \neq d_2} s(d_1, d_2)\,. \qquad (2)$$

[27] Various scale factors such as term frequency times inverse document frequency (TFIDF) are often used, but we omit these details for clarity.

We use a hierarchical agglomerative approach [11,16,21], in which we initially have each document to be clustered in its own group Γ, and at each step, that pair (Γ, Δ) is merged which leads to the maximum $s(\Gamma \cup \Delta)$. This step is repeated until only one group is left with all documents in it.

3.3.3. Exploiting hints from folder membership

Suppose a page d has to be classified for a person p_1, where d has already been bookmarked in some folder c_2 by another user p_2. Suppose many other members of c_2 have been bookmarked by p_1 as being in folder c_1; then there is reason to guess that d may belong to c_1 too. More generally, regarding p_2's folders as a *reference* taxonomy (such as Yahoo!), we can take all of p_1's bookmarked pages, and run them through p_2's classifier. We also run d through p_2's classifier, obtaining its class $c_{p_2}(d)$. We consider all documents D from p_1 which are in this same class, and guess the most frequently occurring class label (from p_1's folder system) as the class for d.

This experiment was performed with the first level of Yahoo!. The results shown in Fig. 7, while intuitive, are quite instructive: almost half the time, user folders are pairwise consistent at the first level of Yahoo! There is also a mild locality effect: if the folder is represented in HTML (as in Netscape) and two URLs are a short distance apart, this makes it more likely that their Yahoo-topics are the same.

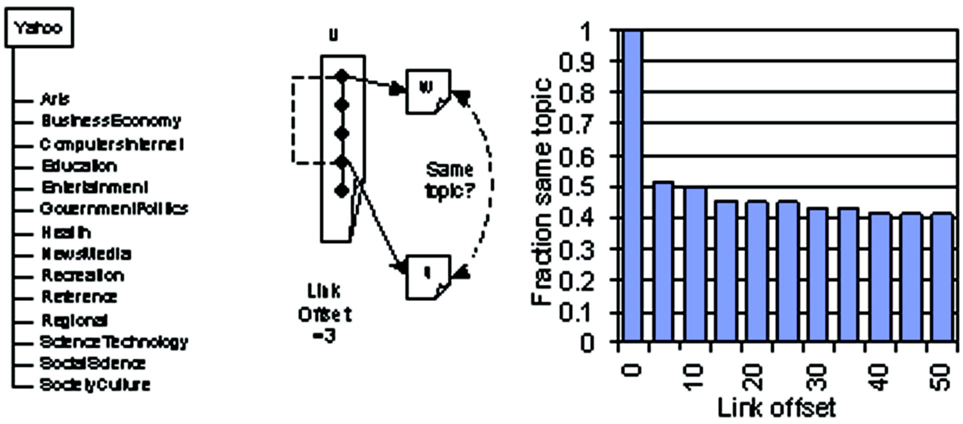


Fig. 7. Verification that folders tend to be reasonably coherent as judged by a reference taxonomy. Pairs of URLs were selected from our Netscape bookmark collection, the page fetched, its frequent terms formed into a query, and this query classified using Yahoo, to see if both belong to the same Yahoo class. The fraction of times this happens is plotted against the distance (in number of intervening out-links) between the two URLs.

This suggests the following algorithms:

(1) Build text-based classifiers for all users p.

(2) We are given a document d to classify under p_1. Find d in other users' folders to find a set of folders C_2. (If d is not found in anyone else's folders then fail.) Next we have two variants, *folder* and *locality*.

(3) (Folder) Classify all documents under each folder in C_2 as per p_1's folder system. Consider the majority class as the class for d.

(4) (Locality) Suppose d is found in a folder c_2. Call this offset zero. Scan left and right until d_{-i} and d_{+j} are found in c_2, such that d_{-i} and d_{+j} are classified under p_1. If these classes are the same, output this class, otherwise, fail.

We measured recall (coverage) and precision (for what fraction of documents was the proposed class approved by the user). The results are shown in Fig. 8. Because the algorithm might flag a failure, recall is not perfect, but, in the cases where it can generate a folder assignment, the answer is significantly more accurate than using the text accumulated by user p_1 alone. Basically, an extended vocabulary contributed by the whole community makes far more accurate classification. We also verified that our collection of bookmarks were reasonably spread out across Yahoo's taxonomy, hitting eight of the fourteen classes of Yahoo shown in Fig. 7.

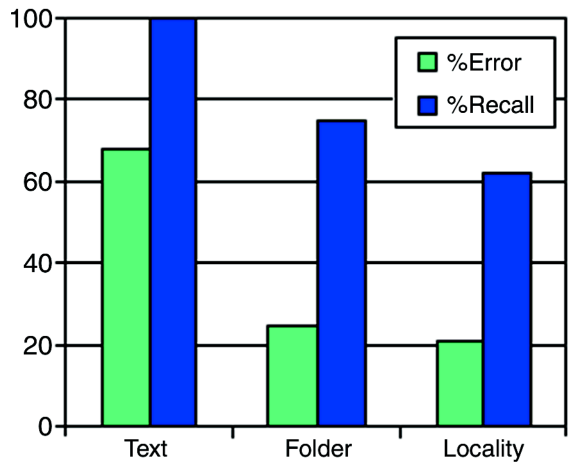


Fig. 8. Results of exploiting folder co-location and link locality for classifying history.

3.4. The community taxonomy synthesis problem

The results above clearly show that *relations* between pages (such as being placed in same or different folders) are quite important as features for learning. HyPursuit [28] recognized this, but only offered hand-tuned weights for different feature types. Similar issues also arise in combining text, HTML tags, and hyperlinks into one model. Basically, we want to exploit the fact that co-location of documents in a folder hint at a semantic similarity which should be factored in together with our text-based similarity in Eq. 2).

3.4.1. Problem formulation

The input to the problem is the following initial graph N with three layers of nodes, D for documents, G for groups, and F for folders. Initially, there are as many groups as documents, and each document is in its own group. We convert the original bipartite relation between folders and documents into a tripartite representation to help standardize the graph cost models in our algorithms (in Section 3.4.2). There is a many-to-many mapping between folders and groups. A folder maps to at least one group and vice versa. The F-to-G and G-to-D mappings induce each folder f to include a set of documents D_f; these sets can be overlapping. D_f induces a term distribution T_f for folder f. We have various choices for characterizing T_f, some will be discussed later.

The goal of taxonomy synthesis is to derive a new three-layer graph N'. In N', $D' = D$ and $F' = F$, but G' and the connecting edges have been modified. Groups no longer contain a single document in general. The F-to-G' mapping is many-to-many as before. Associated with such three-layer graphs is a cost model. N has some initial cost. N' has some (hopefully) lower cost. Our goal is to derive N' with as small a cost as possible (see Fig. 9).

Now we design the cost model. The cost of a graph is the sum of the *distortion* and *mapping* costs. Distortion is a measure of how much the new term distribution T'_f of a folder differs from the 'true' distribution T_f (i.e., what is the penalty for changing the red subgraph). Distortion is denoted $D(T'_f \| T_f)$, also called the Kullback–Leibler distance. For the initial graph, distortion is zero by definition. The

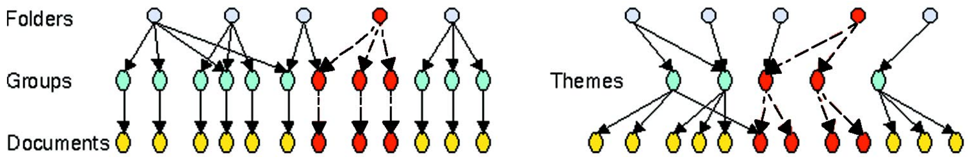


Fig. 9. A formulation for the community taxonomy synthesis problem. Initially, each document is a group or theme by itself. Finally, we want a highly compressed representation of the tripartite mappings while distorting the term distributions for original folders (induced by the dark subgraphs with dotted edges) as little as possible.

mapping cost is a measure of the complexity of specifying G or G' and the edges in the graph.

Why do we seek to minimize the total cost? This leads to the 'simplest' description of the data, which is the best model, according to the Minimum Description Length principle. For many formulations of the distance measure, the problem is at least as hard as set-cover. Hence we resort to fast heuristics.

3.4.2. Heuristics and experience

The first problem is with the KL-distance. Over the set of terms T, let I_T range through the set of all possible *document events*; i.e., all possible vectors of term counts. The KL distance of a distribution T' with reference to another distribution T is:

$$D(T'\|T) = \sum_{I_\mathsf{T}} \Pr_T(I_\mathsf{T}) \log \frac{\Pr_T(I_\mathsf{T})}{\Pr_{T'}(I_\mathsf{T})}. \tag{3}$$

The KL distance is not symmetric; however, it is non-negative and zero only for $T = T'$. The sum over all I_T is impractical to compute; even with the binary model, there will be $2^{|\mathsf{T}|}$ terms in the sum. Therefore, we take recourse to inter-term independence as in the bag-of-words model, except we use a *binary* document model, where word counts are ignored for simplicity.

$$D(T'\|T) = \sum_t \phi_t(\log\phi_t - \log\phi'_t) + (1-\phi_t)(\log(1-\phi_t) - \log(1-\phi'_t)) \tag{4}$$

where $\phi_t = \Pr_T(t)$ and $\phi'_t = \Pr_{T'}(t)$.

Mapping costs for the edges can be approximately estimated by encoding the vertex IDs using a Shannon-optimal code. Details are omitted for lack of space.

With this two-part cost model in mind, we will compare three heuristics: *LeafUnion*, *SingleBest* and *Bicriteria*, as specified next.

LeafUnion. This is the trivial mapping corresponding to each folder simply maintaining a 'link' to the original documents. The distortion cost is zero by definition, but the mapping cost is large. The mapping is not exploiting document similarities at all.

SingleBest. Once the hierarchical agglomerative cluster tree (dendogram) is built, each folder is assigned to exactly one node in the dendogram. Here we are trying to cut down on mapping cost, hoping that the clustering step will bring related documents under the same subtree so that few links will suffice. This may entail significant distortion.

Bicriteria. This heuristic recognizes the tension between mapping and distortion cost. It starts similar to hierarchical agglomerative clustering (HAC) with a frontier of groups, each with one document. In the end, it has a smaller frontier, with the original folders mapping to the surviving groups. Unlike in *SingleBest*, a folder can map to many groups.

The algorithm is a greedy cost-based sequence of merges. When two groups Γ and Δ on the frontier are evaluated for merging, we do not use $s(\Gamma \cup \Delta)$ as the goodness measure. Assuming all folder pointing to either Γ or Δ will point to their union, we estimate

- the decrease in mapping cost because of edges saved by the merger,
- the increase in distortion because of extraneous documents in Γ and Δ, and
- perform that merge with minimum increase in overall cost.

Sample results are shown in Fig. 10. As expected, *LeafUnion* pays too much in mapping cost and *SingleBest* pays too much in distortion cost. Although evaluation of discovered themes is necessarily subjective at this point, inspection of *Bicriteria* results showed that themes *were* being factored out of partially overlapping topics corresponding to folders of different users.

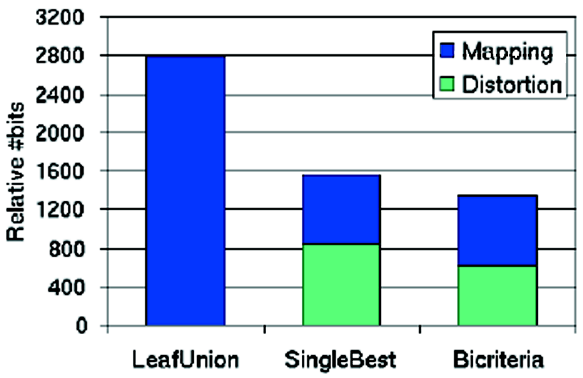


Fig. 10. Results for taxonomy synthesis, showing that both collocation in folders and term distribution are important features.

4. Conclusion

We have reported on the design and implementation of Memex, a companion on the Web for individuals and communities. We have demonstrated that Memex is effective in archiving and analyzing surfing patterns to bring out useful organization of surfing history. Specifically, Memex helps organize history into coherent topics, and relate topics between different users. It also enables search over the entire surfing history.

4.1. Status report

Memex is currently being tested by local volunteers after which we will make the service available for beta-testing. The client-side code runs on Netscape and Internet Explorer. A port is planned for HotJava and the JDK1.2 plug-in. Initial user feedback is that the client is intuitive and easy to use. However, the server is non-trivial to set up; knowledge of Apache, JServ, and some relational database like Oracle is needed. We wish to make server installation much simpler using scripts.

4.2. Recommendations for browser vendors

Anyone who develops serious Java code for browsers knows that Java's promise is sadly distant. The leading browsers differ significantly in their security API, as well as the API through which applets can get access to browser information. Dealing with this lack of standardization was quite difficult. Even Javascript differs across browsers; on Explorer there is a dialect called JScript.

The Memex client monitor module communicates with the browser using DDE on Windows and LiveWire (Java-to-Javascript communication tool) on UNIX, to read browser state variables.

We hope that our experiences with grafting Memex into the two most popular browsers will encourage the vendors of these browsers to evolve a more friendly and uniform browser interaction API which they or we can use for more advanced browser assistants. Specifically, we recommend that trusted applets get a handle to a `Browser` object which can then be used to interact closely with the browser:

```
import com.netscape.browser.*;
// or import com.ms.browser.*;
Browser b = Browser.getBrowser();
Vector wl = b.getBrowserWindows();
BrowserWindow bw = (BrowserWindow) wl.elementAt(0);
bw.addPageLoadListener(new PageLoadListener() {
    public void actionPerformed(LoadEvent le) {
        URL location = le.getURL(), referrer = le.getReferrer();
        String title = le.getTitle();
        Vector outLinkList = le.getOutlinks();
        // use these ...
    }
};
```

It goes without saying that the non-standard security and browser interaction interfaces cause much grief and prevent any serious browser-based Java developer from writing simple, clean code. The Java

2 plug-in will remove the security-related problems, but not the browser interaction issues.

4.3. Interaction with other systems and meta-data

Planned extensions to Memex involve integration with a caching proxy using ICP (Internet Caching Protocol). Currently, proxies associate URLs with content, and run variants of weighted LRU strategies. We believe that Memex, by analyzing content, can more intelligently advice the proxy about prefetching and eviction policies, thus tuning to the community that it serves. Memex can also help proxy servers detect mirror sites, reducing pressure on the cache. This will in turn help Memex to get its bulk of document fetches from the proxy.

The **Resource Description Framework** (RDF), based on XML, provides a uniform syntax for defining and exchanging meta-data about Web pages in a format that can be manipulated by programs. Various communities are defining and agreeing on resource descriptions suited to their applications. In the context of Memex, a description language for topic taxonomies and associated resources, such as developed by the *Open Directory*, would be very useful. We intend to adapt that format to the description of individual and community interest profiles as well, through designing suitable description formats for folders and synthesized taxonomies.

Our premise is that Memex servers will find utility at the workgroup or organization level. In the next phase, we plan to develop and design protocols for interaction between distributed Memex servers for consolidating documents, topics, and communities across diverse geographical regions. Such cooperative analysis will help generate strategies for bandwidth and cache management for long-haul ISPs as well.

Acknowledgements

Thanks to S. Sudarshan for helpful discussions, to Hannes Marais for tips on using DDE, and to the reviewers for helping improve the presentation of the paper. Thanks to the School of Information Technology, IIT Bombay, for providing supplementary network resources during the work. Memex builds upon many pieces of public domain software: HTTPClient, Cryptix, JFlex, and Berkeley DB.

References

[1] C. Apte, F. Damerau and S.M. Weiss, Automated learning of decision rules for text categorization, ACM Transactions on Information Systems, 1994, IBM Research Report RC18879.

[2] C. Apte, F. Damerau and S.M. Weiss, Towards language independent automated learning of text categorization models, in: SIGIR, 1994, IBM Research Report RC19481.

[3] R. Barrett and P.P. Maglio, Intermediaries: new places for producing and manipulating web content, in: 7th International World Wide Web Conference, Brisbane, 1998, Online versions: HTML, PDF.

[4] I. Ben-Shaul, M. Herscovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalheim, V. Soroka and S. Ur, Adding support for dynamic and focused search with Fetuccino, in: 8th World Wide Web Conference, Toronto, May 1999.

[5] V. Bush, As we may think, The Atlantic Monthly, July 1945, Online at http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm.

[6] S. Chakrabarti, B. Dom, R. Agrawal and P. Raghavan, Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies, VLDB Journal, Aug. 1998, Invited paper, online at http://www.cs.berkeley.edu/∼soumen/VLDB54_3.PDF.

[7] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan and S. Rajagopalan, Automatic resource compilation by analyzing hyperlink structure and associated text, in: 7th World Wide Web Conference (WWW7), 1998, Online at http://www7.scu.edu.au/programme/fullpapers/1898/com1898.html.

[8] S. Chakrabarti, D.A. Gibson and K.S. McCurley, Surfing the Web backwards, in: WWW, Vol. 8, Toronto, May 1999, Online at http://www8.org.

[9] S. Chakrabarti, M. van den Berg and B. Dom, Focused crawling: a new approach to topic-specific web resource discovery, Computer Networks 31 (1999) 1623–1640 (first appeared in the 8th International World Wide Web Conference, Toronto, May 1999), Available online at http://www8.org/w8-papers/5a-search-query/crawling/index.html.

[10] W.W. Cohen, Integration of heterogeneous databases without common domains using queries based on textual similarity, in: SIGMOD, Seattle, WA, 1998, ACM.

[11] D.R. Cutting, D.R. Karger and J.O. Pedersen, Constant interaction-time scatter/gather browsing of very large document collections, in: Annual International Conference on Research and Development in Information Retrieval, 1993.

[12] J. Dean and M.R. Henzinger, Finding related pages in the

world wide web, in: 8th World Wide Web Conference, Toronto, May 1999.

[13] R. Duda and P. Hart, Pattern Classification and Scene Analysis, Wiley, New York, 1973.

[14] S. Dumais, J. Platt, D. Heckerman and M. Sahami, Inductive learning algorithms and representations for text categorization, in: 7th Conference on Information and Knowledge Management, 1998, Online at http://www.research.microsoft.com/~jplatt/cikm98.pdf.

[15] M. Hersovici, M. Jacovi, Y.S. Maarek, D. Pelleg, M. Shtalheim and S. Ur, The Shark-Search algorithm — an application: tailored web site mapping, in: 7th World Wide Web Conference, Brisbane, Apr. 1998, Online at http://www7.scu.edu.au/programme/fullpapers/1849/com1849.htm.

[16] A.K. Jain and R.C. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.

[17] T. Kamada and S. Kawai, An algorithm for drawing general undirected graphs, Information Processing Letters 31 (1989) 7–15.

[18] J. Kleinberg, Authoritative sources in a hyperlinked environment, in: ACM–SIAM Symposium on Discrete Algorithms, 1998, Online at http://www.cs.cornell.edu/home/kleinber/auth.ps.

[19] D. Koller and M. Sahami, Hierarchically classifying documents using very few words, in: International Conference on Machine Learning, Vol. 14, Morgan-Kaufmann, Los Altos, CA, July 1997, Online at http://robotics.stanford.edu/users/sahami/papers-dir/ml97-hier.ps.

[20] W.-S. Li, Q. Vu, D. Agrawal, Y. Hara and H. Takano, PowerBookmarks: a system for personalizable Web information organization, sharing and management, Computer Networks 31, May 1999 (first appeared in the 8th International World Wide Web Conference, Toronto, May 1999), Available online at http://www8.org/w8-papers/3b-web-doc/power/power.pdf.

[21] Y.S. Maarek and I.Z. Ben Shaul, Automatically organizing bookmarks per content, in: Fifth International World Wide Web Conference, Paris, May 1996.

[22] P.P. Maglio and T. Matlock, Metaphors we surf the Web by, in: Workshop on Personalized and Social Navigation in Information Space, Stockholm, 1998.

[23] H. Marais and K. Bharat, Supporting cooperative and personal surfing with a desktop assistant, in: Proc. of UIST'97, ACM, Oct. 1997, pp. 129–138, Online at http://www.research.digital.com/SRC/personal/Johannes_Marais/pub/uist97/uist97paper.pdf.

[24] T. Mitchell, Machine Learning, McGraw-Hill, New York, 1997.

[25] R.R.P. Pirolli and J. Pitkow, Silk from a sow's ear: extracting usable structures from the web, in: ACM CHI, 1996.

[26] G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, New York, 1983.

[27] F.-S. Shieh and C. McCreary, Directed graphs by clan-based decomposition, in: Graph Drawing, 1995, pp. 472–482.

[28] R. Weiss, B. Velez, M.A. Sheldon, C. Nemprempre, P. Szilagyi, A. Duda and D.K. Gifford, HyPursuit: a hierarchical network search engine that exploits content-link hypertext clustering, in: Proc. of the Seventh ACM Conference on Hypertext, Washington, DC, Mar. 1996.

[29] O. Zamir and O. Etzioni, Grouper: a dynamic clustering interface to Web search results, in: 8th International World Wide Web Conference, Toronto, May 1999, pp. 283–296, Elsevier, Amsterdam.

Soumen Chakrabarti received his B.Tech in Computer Science from the Indian Institute of Technology, Kharagpur, in 1991 and his M.S. and Ph.D. in Computer Science from the University of California, Berkeley in 1992 and 1996. At Berkeley he worked on compilers and runtime systems for running scalable parallel scientific software on message passing multiprocessors. He was a Research Staff Member at IBM Almaden Research Center between 1996 and 1999. At IBM he worked on hypertext analysis and information retrieval. He designed the *Focused Crawler* and part of the *Clever* search engine. He is currently an Assistant Professor in the Department of Computer Science and Engineering at the Indian Institute of Technology, Bombay. His research interests include hypertext information retrieval, Web analysis and data mining.

Sandeep Srivastava, **Mallela Subramanyam**, and **Mitul Tiwari** are junior-year students in the Computer Science and Engineering Department at IIT Bombay. Their research interests include hypertext databases and data mining.