The TrailTRECer Framework — A Platform for Trail-enabled Recommender Applications

Erich Gams, Tobias Berka, and Siegfried Reich

Salzburg Research - SunTREC
Jakob Haringer Strasse 5/III
5020 Salzburg, Austria
E-Mail: {egams, tberka, sreich}@salzburgresearch.at

Abstract. In their everyday work people are confronted with ever growing amounts of information and thus often feel overloaded with data. Trails, built from information about the users' browsing paths and activities, are an established approach to assist users in navigating vast information spaces and finding appropriate information. While existing systems focus on web browsers only, we argue that trails can be generated by any application. We describe *TrailTRECer*, a framework which supports trail-based information access, and which is open to any application. The usability of the framework and the concept of user trails were tested by building a trail-enabled browser client and a print manager client. Initial user evaluations indicate the usefulness of this approach.

1 Introduction

Looking for specific information or finding colleagues working on similar topics, has always been and still is an open issue. Growing numbers of documents within Intranets and – even more — on the World Wide Web make it increasingly difficult to find appropriate information. When navigating through information spaces people have to make choices without sufficient personal experiences of the alternatives available. In addition to the high number of choices available, these choices vary widely in quality. Therefore, following everyday life where people rely on recommendations from other people — either by directly getting in contact or indirectly through their trails — so-called recommender systems [17] propose to support the recommendation process by information technology and thus enable *social navigation* [5]. Recommendations can provide a manageable and useful view or filter of the whole information space. Such recommendations can be based on experiences and on opinions of other people who are more familiar with a particular domain.

In our work we take an approach based on user trails, much like, footprints users leave when handling information [14]. With trail-based recommender systems, which base access to and retrieval of information on user trails, users could manage their individual information spaces more efficiently. Most existing systems provide a kind of history mode of recently accessed web pages and thus focus on one application domain only. In this paper we present an open, distributed and adaptable framework, named *TrailTRECer* which supports generation, manipulation and query by any application [15]. Due to this property we consider this to be an open system. For our project this means that implementation of new features can be added without modifying the basic structure. Its name indicates that, similar to the way “Trackers” follow trails in nature, “TrailTRECers” make their way through digital information spaces.

This paper is structured as follows: Section 2 is a survey of related work. After a brief discussion, we identify phases and the requirements for a framework supporting trail-enabled applications in Section 3. The framework will be based on a data model and an architecture defined in Section 4. For validation purposes of the framework and the data model that we constructed, we introduce two sample applications in Section 5, a BrowserManager and a PrintManager. Finally, Section 6 summarises the results of first user experiments and conclusions gained from them.

2 Related Work

Trail based systems recommend related items and provide assistance in navigation, and therefore share similarities with recommender systems and browsing advisors. Following the approach of recommending related items, two main groups have been derived: collaborative or social filtering systems and content based filtering systems [17]. Other research concerning assisting users in navigating the web can be loosely grouped together under the term browsing advisors. Trail based systems, providing recommendations based on user navigation, constitute an additional fourth category.

2.1 Content based and Collaborative Filtering Systems

Content based filtering systems are based upon correlation between the content of a document and the user’s preferences. Webwatcher [19] is a content based tour guide agent assisting users in browsing and navigating the Web. The agent accompanies the user by suggesting appropriate hyperlinks identified by keywords of interest. However, in order to expand the range of documents and in order to support serendipity, people rely on explorations. Letizia and Powerscout [10] are examples of so called reconnaissance agents — programs that look ahead in the user’s browsing activities and recommend the best path to follow. Both learn a profile of user preferences by recording and analysing the user’s browsing activities. Content based systems assume, that documents have to be machine parseable (hence most these systems focus on text documents) or attributes are assigned to them manually. By contrast trail based systems handle multimedia data by using navigation to implicitely determine the relevance of data.

Collaborative or social filtering systems generate personal recommendations by computing the similarity between a user’s preferences and those of people with similar interests. PHOAKS [20] for instance collects URLs that have been positively mentioned in an electronic message or in a FAQ document. The URLs are filtered, e.g. through the number of distinct recommenders of a resource and provided to the people interested in a newsgroup. In Grouplens [9], a recommender system based on news messages, users assign numeric scores to each news article they read. The scores of different users are correlated with each other in order to find users who share a similar taste. Modified news clients allow rating by numbering. Summarizing collaborative filtering systems rely on user profiles, expressed by the users manually, or on algorithms automatically weighting people's interests with similar taste to produce recommendations. Users also need to use dedicated clients in addition to their favourite common applications.

2.2 Browsing Advisors

Broadway [7] is an example of a cooperative browsing advisor for the WWW, that uses case-based reasoning to advise pages by tracing past navigation sessions of users. The advise is mainly based on similarity of ordered sequences of past accessed documents. Footprints [21] consists of a set of tools that base access to interaction history between users and digital objects by navigation. The system doesn't use the history to make recommendations, but to contextualize Web pages the user is seeing.

2.3 Trail-based Systems

Most browsing advisor systems reuse paths followed by a user or do analysis based on browsing behaviour. In the context of a trail based system, we assume the user not only to search for information on the Web to acquire additional knowledge, but also pass a workflow of creating or editing a document. Yet we differ from the approach of watching the interaction history of each document separately [6]. This qualifies trails for entirely different application domains than just browsing the Web [14]. In our understanding, trails constitute a specific path through a set of documents not limited to the WWW.

Thus, with respect to all the systems described above we conclude, that they do not fully support the notion of trails in terms of objects that can be created, copied and modified. Also, trails must not be dependent on the links an author of a hypertext created in the document e.g. links within a Web page. Trails can exists between documents of different type that have no explicit linkage between them. Memoir [18] has implemented the basic notion of trails within a Web environment and demonstrated the feasibility of trails for achieving the goals of matching users with similar interests. We argue that trail-based systems should be open with respect to the applications integrated and the activities traced. Consequently, we propose to develop an distributeable, adaptable component framework that provides an open set of services for recording, storing, processing and navigating trails through well-defined interfaces. By *Framework*, we refer to a software environment [2] that is designed to simplify the development and management of applications by providing a re-usable context for components; therefore, we see the framework and its components, as cooperating technologies [8]. In order to extract requirements for the construction of a trail based recommender system, we start with a brief overview of user scenarios. Basically these scenarios consist of: tracing and recording of trail data, finding related documents to the currently active one, finding related trails and related users (e.g. people with the same interest). The basic intention of getting recommendations is to find related documents and users with the same interests. By documents we refer to all kinds of nodes of a trail that can be identified by a URL.

3 Phases and Requirements

More precisely, three main phases can be extracted out of these scenarios: *Acquisition* of trail data from clients of everyday use, *Processing* of recorded trails to provide accurate recommendations to the user and *Management* of recorded trail data *over time* (forming a loop with processing)

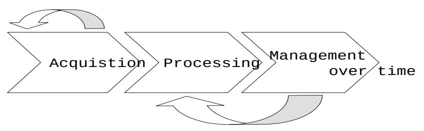

Fig. 1. Main phases of a trail-based system

Based on the phases of the trail processing workflow we have identified the following requirements, that will be implemented in the data model and the architecture of *TrailTRECer*:

R1 User Groups When searching for information users are presumed to play different roles and thus can be assigned to one or more user group. Users can be divided into groups according to different tasks or projects or groups considering the departmental structure. Groups can have subgroups.

R2 Scalability If the number of users increases, the recommendations should improve in terms of quality and trueness, but not in terms of quantity of the results. The performance should not slow down or make the system more a burden than a help for the user.

R3 Guaranteeing user's privacy Recording a trail should be possible with guaranteeing the users' privacy, therefore it must be ensured that recording takes place only with the user's consent [13].

R4 Openness of the system to arbitrary applications Users want to produce trails with common desktop applications. Trails can be generated by any application, i.e., mail clients, word processors, etc. not limited, as most systems do, to Web browsers [18, 21, 7]. Similar to the way open hypermedia link services offer themselves to a variety of adapted or purpose built applications the integration of trail-based client applications [4, 22] with trail-services should be envisaged. A modified print manager, for instance, might be used to log those documents that have been printed (printing could indicate a document's importance). An interface between the components of the framework and the applications will provide access to basic functionalities, such as processing of trail data and management over time.

R5 Associate activity performed on a document with this document As we argued in [15] the type of activity is important for the documents' relevance and therefore has to be associated with the document and traced. Activities can provide additional metadata and allow users to filter relevant documents from their trails. Moreover, the management of trails over time requires additional data for efficient filtering and retrieval of documents.

R6 Common and extensible data model Besides activity the definition of trails also includes capturing contextual settings. For the basic entities see [15]. The data model should be open enough to support the addition of captured properties.

R7 Trails should be treated as first class objects Trails are objects in their own right, which can be edited, deleted, or copied. Other services like exchanging trails between users would also be possible. A trail node can be anything (e.g. document, website) that can be clearly identified by a URL and has certain properties such as date, duration, activity.

R8 Trails defined as another hypertext domain Following the Open Hypermedia Systems (OHS) community a trail-based system can be designed as another component of a Component-Based-OHS [16]. Trailbased functionality would be another hypertext application domain such as the navigational, taxonomic or spatial domain. Trails should be mapped into an Open Hypertext Model [11], offering new ways of navigation, representing trails in different views.

R9 The pertinence of trails can change over time Knowledge and interests of people and also organisations change over time and so do trails. Some trails may attract more attention, than others. The frequency of trails' usage indicates that some trails are more important than others.

4 System Architecture

Based on the requirements above, we will describe the trail data model and the architecture of the *TrailTRECer* framework. Following the Open Hypermedia

Systems Working Group's reference architecture [16], trail functionality could simply be seen as another middleware service. However, an alternative option would be to follow the fundamental open hypermedia model (FOHM, [11]) and express trail data using that model. FOHM distinguishes between three basic hypertext domains. By adapting FOHM, interoperability between the different hypertext domains themselves will be possible (Adressing R4 and R8). FOHM is a common data model capable of representing structures and implementing operations from any of the three domains (See R8). The basic data model can briefly be summarized as follows: Associations hold vectors of bindings, a relationship type, and a structural type. Bindings glue data references and feature vectors together. Feature vectors can be defined arbitrarily.

Following that model, trails can be defined as associations. Trail marks resemble bindings, i.e., they relate nodes (which are references to the actual documents) and trails together. Activities are modeled as feature values. With respect to retrieving relevant trail marks (and subsequently nodes), we argue that FOHM should include a notion of relevance. This could be a standard feature that would in the simplest case accommodate a basic ranking mechanism; it could also include more elaborate definitions such as general contextual data (subject, place, and time). Additionally, sequential ordering of bindings is needed in order to allow for time dependent ordering and ranking. To summarize, FOHM could be adapted with some simple modifications to hold trail data.

The more accurate the system's recommendation to the users needs to be, the more complex the processing will be. Therefore, different paradigms such as software agents offer themselves for application in this domain [3]. SoFAR [12], the **So**uthampton **F**ramework for **A**gent **R**esearch, is a multi-agent framework that addresses the problem domain of distributed information management (addressing R2) and thus is well suited for our trail-based framework *TrailTRECer*. SoFAR provides information sharing between agents, promoted through the matchmaking mechanism of a registry agent. Every agent of the system subscribes to the registry agent in order to inform about the own functionality and to find certain capabilities supported by other agents running in the system.

In order to communicate and exchange information via an agent communication language, agent systems separate intention from content, by using a predefined set of performatives to carry the messages and an ontology as the topology of messages. We developed a trail ontology, which enables the exchange of related documents, trails or persons. The root of the ontology hierachy is an abstract term and a predicate, a term that can be queried about. *Related* is the parent predicate of the trail ontology. Any agent, that wants to receive related documents, can query for the *Related* predicate. *Related* only consists of a single field, namely *Trailmark*. Trailmark contains the properties of the current viewed document (See [15] for a definition) (also addressing R6 and R5). For exchange of related persons, documents or trails *Related* can be derived into different supported subtypes *RelatedPersons, RelatedDocuments, RelatedTrails*.

The *TrailTRECer* framework is built upon this trail ontology, enabling communication about any trail related information. All the tasks, such as collecting data, or visualization are distributed on different agents (addressing R1). Figure 2 depicts the architecture of the framework, containing two platforms, on which the agents reside.

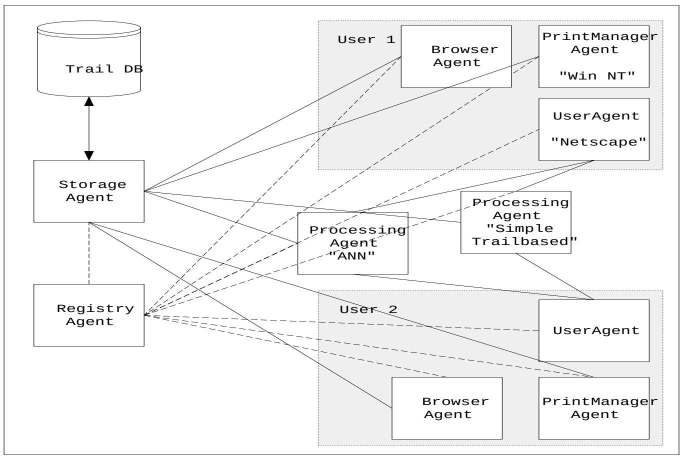

Fig. 2. Framework architecture

Each user holds an individual user platform, that contains all trail acquisition and trail visualization agents. Trail acquisition agent can be specialized in recording individual activities. Furthermore, each user can have an arbitrary number of individual trail acquisition and trail visualization agents (R4). The group platform acts as a wrapper for several users, preferably assigned according to the basic grouping of the recommender system. This platform hosts various agents, the processing agents, that can deliver "Related" documents or trails, and the storage agent, responsible for saving trails in a global database. In a further expansion, the framework can also hold agents managing the growth of trail data over longer usage of the system via aging algorithms (R9).

5 Sample Application Scenarios

In order to promote the feasability of trails for recommendation and to test the applicability of our framework, two trail recording agents a *Browser Agent* and a *PrintManager Agent* were implemented. The *Browser Agent* is wrapped around a HTTP proxy server watching and recording the browsing activity of the user. The *PrintManager Agent* has to be more dependent on the operating system used. For WinNT, we developed a component wrapped into a JNI interface, watching the print jobs of all running applications and recording the documents printed. Both agents pass their trails to the Storage Agent which connects to a relational database and stores them.

Furthermore two processing agents were integrated in the system. If the current document, identified by its URL, is found in a previous trail, the next neighbour documents will be recommended by the *Simple-Trail-Based Agent* ordered by the activity associated and the frequency of occurrence. A more advanced version will recommend a trail, that the user can follow, as a whole (R7). Additionally, an *Artificial Neural Network ANN Agent* recommends related domains based on ratings from an artificial neuronal network trained with trail data. (For details on the implementation see [1])

For a priming phase, trail data has been generated automatically from proxy access logs of several months browsing activity of the members of our department. In order to test the accurancy of recommendations, our *User Agent* connected to a sidebar integrated in Netscape 6.2. From there the user can access the delivered recommendations and also control the recording of trails (R3). The name of the processing agents together with the results fitting to the current viewed document are displayed. The documents viewed with the browser are indicated through the HTTP address and the documents printed through the file name. So far, no login component has been integrated into the framework, instead we identified the user by extracting the login name from the operating system (See R2). Figure 3 shows the recommendations of the two integrated processing agents in the left part of Netscape.

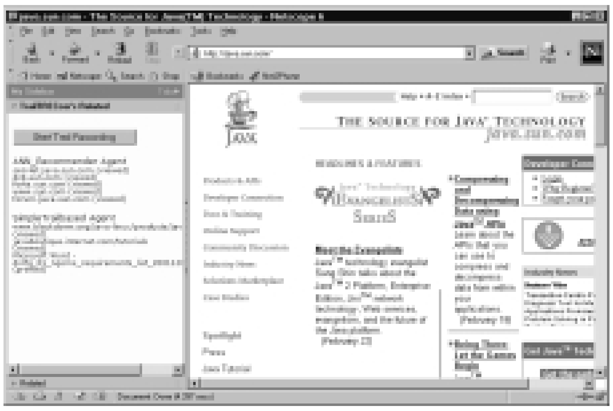

Fig. 3. Screenshot

The initial user evaluation of the *TrailTRECer* prototype included ten users with varying IT skills and educational backgrounds. The evaluation indicated that the processing agents' time consumption are acceptable, under the following conditions: The user can decide which agent to use, or other non-real-time methods of user notification are used, e.g. email [3]. The agent selection should be supported by a textual description of the agents.

Another important issue is the presentation of the agents' results. Since the algorithm parses a great number of documents, simply displaying the name or title of the document residing at that address may lack transparency, a problem which can be evaded by presenting the results in a manner similar to search engines (by displaying link target title, meta-data or excerpts). In the case of trails, it would be helpful to show the start and end trailmark, the trailmarks corresponding with the current trail, or extract a description out of the trail e.g. number of trailmarks (See R7).

6 Summary and Conclusion

In this paper, we introduced requirements, a software architecture and prototype components of a framework that enables the development of trail-enabled applications. Our framework focuses on reuseable, inter-operating components supporting services for processing, visualizing and for managing the relevance of trail data over time. The system can record and process trails with minimal effort from the user and at the same time provide recommendations appropriate to the currently active document. Furthermore, we proved the openness of our framework by implementing two sample applications, tracing print and browse activities of users. More clients can be easily integrated in the framework and involved in the recommendation process. Our next steps will focus on the integration of further applications, such as a newsgroup client and an email client, tracking email communication and newsgroup activities.

We believe that the usefulness of the approach chosen, i.e., using trails as basis for recommender systems, has been demonstrated by the positive feedback gained in the initial user study.

7 Acknowledgments

Trailist is supported by the Austrian Fonds zur Förderung der wissenschaftlichen Forschung (FWF) under grant No. P14006-INF.

References

1. Tobias Berka, Werner Behrendt, Erich Gams, and Siegfried Reich. A trail based internet-domain recommender system using artificial neural networks. *Accepted at the Int. Conf. on Adaptive Hypermedia and Adaptive Web Based Systems*, 2002.
2. Philip A. Bernstein. Middleware: A model for distributed system services. *Communications of the ACM*, 39(2):86–98, February 1996.
3. Leslie A. Carr, Wendy Hall, and Steve Hitchcock. Link services or link agents? In *Procs. of the '98 ACM Conference on Hypertext, 1998*, pages 113–122, 1998.
4. Hugh C. Davis, Wendy Hall, Ian Heath, Gary J. Hill, and Robert J. Wilkins. Towards an integrated information environment with open hypermedia systems. In *ECHT '92. Procs of the ACM conference on Hypertext, 1992*, pages 181–190, 1992.

5. Andreas Dieberger. Supporting social navigation on the world wide web. *Int. Journal of Human-Computer Studies*, 46(6):805–825, 1997.
6. W. Hill, J. Hollan, D. Wroblewski, and T. McCandless. Edit wear and read wear. In *Procs. of ACM Conf. on Human Factors in Computing Systems, CHI'92.*
7. Michel Jaczynski and Brigitte Trousse. Broadway: A case-based system for cooperative information browsing on the world-wide-web. In *Collaboration between Human and Artificial Societies*, pages 264–283, 1999.
8. Ralph E. Johnson. Frameworks = (components + patterns). *Communications of the ACM*, 40(10):39–42, October 1997.
9. Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.
10. Henry Lieberman, Christopher Fry, and Louis Weitzman. Exploring the web with reconnaissance agents. *Communications of the ACM*, 44(8):69–75, 2001.
11. David E. Millard, Luc Moreau, Hugh C. Davis, and Siegfried Reich. FOHM: A fundamental open hypertext model for investigating interoperability between hypertext domains. In *Procs. of the '00 ACM Conference on Hypertext, 2000*, pages 93–102, 2000.
12. Luc Moreau, Nick Gibbins, David DeRoure, Samhaa El-Beltagy, Wendy Hall, Gareth Hughes, Dan Joyce, Sanghee Kim, Danius Michaelides, Dave Millard, Sigi Reich, Robert Tansley, and Mark Weal. SoFAR with DIM agents. An agent framework for distributed information management. In *Int. Conf. on The Practical Application of Intelligent Agents and Multi-Agents. PAAM 2000*, pages 369–388, 2000.
13. D. Nichols. Implicit rating and filtering. In *Procs. of the 5th DELOS Workshop on Filtering and Collaborative Filtering*, 1997.
14. Siegfried Reich, Leslie A. Carr, David C. DeRoure, and Wendy Hall. Where have you been from here? Trails in hypertext systems. *ACM Computing Surveys — Symposium on Hypertext*, 31(4es), December 1999.
15. Siegfried Reich and Erich Gams. Trailist - focusing on document activity for assisting navigation. In *Procs. of the Twelfth ACM Conference of Hypertext and Hypermedia*, pages 29–30, 2001.
16. Siegfried Reich, Uffe K. Wiil, Peter J. Nürnberg, Hugh C. Davis, Kaj Grønbæk, Kenneth M. Anderson, David E. Millard, and Jörg M. Haake. Addressing interoperability in open hypermedia: The design of the open hypermedia protocol. *New Review of Hypermedia and Multimedia*, 5:207–248, 1999.
17. Paul Resnick and Hal R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, March 1997.
18. D. De Roure, W. Hall, S. Reich, G. Hill, A. Pikrakis, and M. Stairmand. Memoir - an open distributed framework for enhanced navigation of distributed information. *Information Processing and Management*, 37:53–74, 2001.
19. D. Freitag T. Joachims and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Procs. of IJCAI97*, 1997.
20. Loren Terveen, Will Hill, Brian Amento, David McDonald, and Josh Creter. PHOAKS: A system for sharing recommendations. *Communications of the ACM*, 40(3):59–62, 1997.
21. Alan Wexelblat and Pattie Maes. Footprints: History-rich tools for information foraging. In *Conf. on Human Factors in Computing Systems*, pages 270–277, 1999.
22. Uffe Kock Wiil and Peter J. Nürnberg. Evolving hypermedia middleware services: Lessons and observations. In *Procs. of the ACM Symposium on Applied Computing (SAC '99)*, pages 427–436, 1999.