

Nineteen International Joint Conference on Artificial Intelligence

WORKSHOP PROGRAM



**Multi-Agent Information Retrieval
and Recommender Systems**

Esma Aïmeur, Chair
Robin Burke, Co-Chair
Abdel-Ilhah Mouaddib, Co-Chair

**Sunday
31 July 2005**

Edinburgh, Scotland

Foreword

Providing effective access to on-line information is a problem that brings together many Artificial Intelligence topics: knowledge representation, natural language, user modelling and machine learning, as well as distributed computing, Human-Computer Interaction and databases. Over the last few years, artificial intelligence techniques have provided substantial progress toward solving the problems of information access through the following three domains: models and techniques for multi-agents systems, information retrieval and recommender systems.

In the first domain, there is a large community that develops architectures, language models and techniques for interaction and communication between agents. An important focus for the information retrieval domain has been the development of techniques for coping with the heterogeneous and dynamic information space represented by the World Wide Web and the impact of new retrieval devices and interfaces. However, information retrieval systems do not traditionally care about the individual searcher, preferring instead to focus on the development of global retrieval techniques rather than those adapted for the needs of the individual. Consequently, recommender systems research has focused on the interaction between information retrieval and user modelling in order to provide a more personalized and proactive retrieval experience and to help users choose between retrieval alternatives and refine their queries.

In the future, one can easily imagine a virtual organization of agents with specific tasks such as profile acquisition, web searching and recommendation making. These agents cooperate and negotiate in order to satisfy their users while they put their precious human cycles to a better use. The agents interact to complement their partial solutions or to solve conflicts that may arise. The integration of agent technology will help create and maintain this virtual organization.

Rapidly evolving computer technology, coupled with the exponential growth of the services and information available on the Internet, have already brought us to the point where hundreds of millions of people should have fast, pervasive access to phenomenal amounts of information. The challenge of complex environments is therefore obvious: software is expected to do more in more situations, there is a variety of users, there is a variety of interactions and there is a variety of resources and goals. To cope with such environments, the promise of Multi-Agent Systems is becoming highly attractive.

This workshop emphasizes the interaction between these different research areas, with a view to establishing a framework for more flexible multi-agent retrieval and recommendation solutions.

Organizers

Esma Aïmeur, Chair
Université de Montréal, Canada
<http://www.iro.umontreal.ca/~aimeur>

Robin Burke, Co-Chair
DePaul University, USA
<http://josquin.cti.depaul.edu/~rburke>

Abdel-Ilah Mouaddib, Co-Chair
Université de Caen, France
<http://users.info.unicaen.fr/~mouaddib>

Program Committee

Esma Aïmeur, Université de Montréal, Canada
Robin Burke, DePaul University, Chicago, USA
Nick Jennings, Southampton University, UK
Lewis Johnson, University of Southern California, USA
Wessel Kraaij, TNO, Delft, The Netherlands
Mounia Lalmas, Queen Mary University of London, UK
Benoit Leloup, ENST Paris, France
Abdel-Ilah Mouaddib, Université de Caen, France
Jian-Yun Nie, Université de Montréal, Canada
Mehran Sahami, Stanford University and Google, Inc., USA
Mark van Setten, Telematica Instituut, Enschede, The Netherlands
Barry Smyth, University College, Dublin, Ireland
Katia Sycara, Carnegie Mellon University, Pittsburgh, USA

Table of Contents

Alexander Birukov, Enrico Blanzieri and Paolo Giorgini Implicit: A recommender system that uses implicit knowledge to produce suggestions	1
Robin Burke and Bamshad Mobasher Trust and bias in multi-agent recommender systems: A position paper	8
Conor Hayes, Paolo Avesani and Marco Cova Language games: Learning shared concepts among distributed information agents	11
William H. Hsu Relational graphical models for collaborative filtering and recommendation of computational workflow components	18
Fabiana Lorenzi, Daniela Scherer dos Santos and Ana L. C. Bazzan Negotiation for task allocation among agents in case-base recommender systems: A swarm-intelligence approach	23
Joel Pinho Lucas, Beatriz Wilges and Ricardo Azambuja Silveira Making use of FIPA multiagent architecture to develop animated pedagogical agents inside intelligent learning environments	28
David McSherry Conversational CBR in multi-agent recommendation	33
Timothy A. Musgrove and Robin H. Walsh Utilizing multiple agents in dynamic product recommendations	41
François Paradis, Jian-Yun Nie and Arman Tajarobi Discovery of business opportunities on the Internet with information extraction	47
Maria Salamó, Barry Smyth, Kevin McCarthy, James Reilly and Lorraine McGinty Reducing critiquing repetition in conversational recommendation	55
Elhadi Shakhshuki, André Trudel, Yiqing Xu and Boya Li A probabilistic temporal interval algebra based multi-agent scheduling system	62
Hugues Tremblay-Beaumont and Esma Aimeur Feature combination in a recommender system using distributed items: The case of JukeBlog	70
Wolfgang Woerndl and Georg Groh A proposal for an agent-based architecture for context-aware personalization in the Semantic Web	75
Tong Zheng and Randy Goebel An NCM-based framework for navigation recommender systems	80

Implicit: A recommender system that uses implicit knowledge to produce suggestions

Alexander Birukov, Enrico Blanzieri, Paolo Giorgini

University of Trento

Department of Information and Communication Technology

14 via Sommarive, Povo(TN), Italy 38050

{aliaksandr.birukou,enrico.blanzieri,paolo.giorgini}@dit.unitn.it

Abstract

The number of accessible web pages in Internet increases every day and it becomes more and more difficult to deal with such a huge source of information. In literature many approaches have been proposed to provide users with high-quality links extracted from the thousands of irrelevant ones. In this paper, we present *Implicit*, a system that combines recommender system and multi-agent system approaches and is intended to be used within a community of people with similar interests. It complements the results obtained by search engines with suggestions obtained by means of implicit knowledge of the members of the community. Within the system, agents interact one another, share knowledge and use similarities among users' behaviors in order to increase quality of the recommendations.

1 Introduction

Although searching the Internet is a day-to-day task for many people, the problem of providing effective access to the information available on-line is still open. Due to the huge number of pages on the World Wide Web it is difficult to discover relevant and (or) interesting pages among those provided by a search engine. Therefore, web search is often a rather time-consuming task.

There exist several approaches aimed at solving the stated problem. Search engines are a common and prevailing tool for searching the Web. However, they have several shortcomings. For instance, a query can produce a huge quantity of the pages. Another drawback is a lack of personalization, namely that sometimes "different users may merit different answers to the same query" [Gori and Witten, 2005]. The first shortcoming could be alleviated by formulating an appropriate query for a search engine. Such a reformulation requires, however, a certain intuition and experience of the user. To overcome the lack of personalization, we see the need of supporting the user rather than simply responding to a keyword, which is context-free and impersonal.

Another solution is the use of Internet agents to assist the web browsing. In this field, we find personal assistants that collect observations of their users' behavior in order to recommend previously unseen relevant web pages. There exist

also multi-agent systems, where personal agents collaborate with one another to improve the quality of the suggestions. This approach overcomes the shortcomings of the search engine approach from the personalization point of view. On the other hand, there are other drawbacks like the low number of suggestions generated or even the absence of them in the case of a keyword that has been previously unseen for the personal assistant agent. Sometimes personal agents require extra effort from the user, e.g. specifying his/her area of interests or answering additional questions.

Recommender systems can be also considered as tools for the effective access to the available information. They can be classified as content-based, collaborative filtering, or hybrid systems. Content-based systems produce recommendations by analyzing the content of previously browsed pages and using the obtained information to find pages with similar content. Collaborative filtering systems calculate similarity between the different users and provide the user with the pages that have been selected by the similar users. Hybrid recommender systems exploit both approaches to a certain extent. However, the majority of the recommender systems need user feedback and those systems that collect this feedback in explicit form force user to perform some extra work, like rating the items.

In this paper we present *Implicit*, a multi-agent recommender system. It combines Internet agents and a recommender system. *Implicit* uses a search engine in order to obtain a certain number of suggestions for any entered keyword. Personal agents communicate and collaborate in order to produce recommendations more suitable in the context of the current community¹. Thus, we complement search engine results with the recommendations produced by the agents. This helps us to add personalization without decreasing significantly the number of the pages. As in many recommender systems we attempt to learn the user needs from the observations of his/her behavior.

This paper differs from the previous work in the field of recommender systems and advances the state of the art in the following ways. The system described here is designed to be used within a small organizational community of people,

¹Here we do not give any precise definition of the *community*. We refer to the *community* in a general sense, a group of people working in the same environment and having common interests.

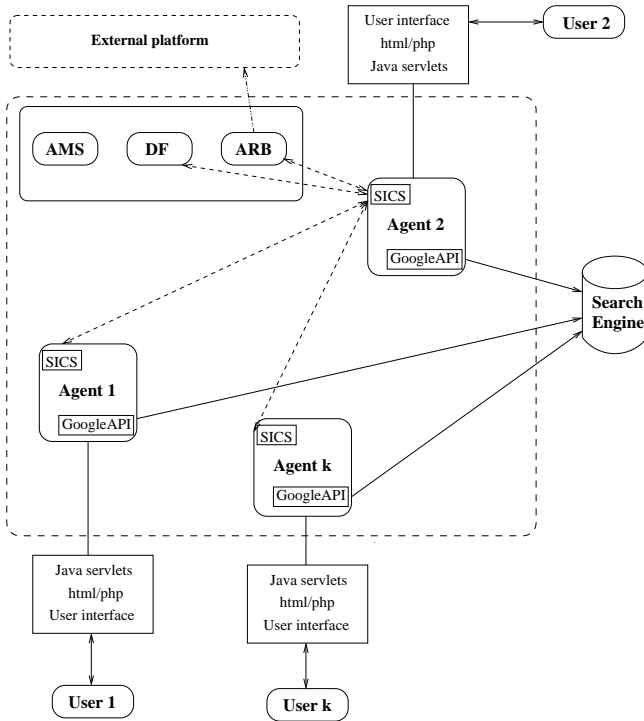


Figure 1: The architecture of the system. *Personal agents* process queries from *users* and interact with each other to exchange links; *SICS* is a part of the personal agent that is responsible for the recommendation creation process; *GoogleAPI* allows to query Google search engine; an *Agent Management System (AMS)* exerts supervisory control over the platform. It provides agent registration, search, deletion and other services; a *Directory Facilitator (DF)* provides agents with other personal agents' IDs. An *Agent Resource Broker (ARB)* deals with links to the services available on the other platforms.

but is not intended for big groups or emergent online communities. We use the universal filtering framework to produce different types of suggestions: links, which are shown to the user, and agents IDs, which are used internally to identify agents to contact. In order to access the information provided by the system, the user does not need to install ad-hoc plugins or a new browser, it is just necessary to register and then load the system homepage. Moreover, we use implicit feedback collection mechanism that requires no additional work from the user. The system structure is rather general in a sense that different data mining techniques can be implemented within the described framework. The methodology given here, once tested, can be moved to another domain, different from web search (see for instance [Sarini *et al.*, 2004]).

The rest of the paper is structured as follows. Section 2 describes the *Implicit* system in detail and Section 3 contains some experimental results on the use of our system. Final Sections 4 and 5 reviews related work and concludes the paper, respectively.

2 Structure of the System

In this section we present a detailed description of *Implicit*. The system exploits the notion of Implicit Culture [Blanzieri

and Giorgini, 2000] to produce suggestions by means of peculiarities found in the community in which it works. Each user of the system has a dedicated personal agent whose task is to assist the user during his/her search and to provide him/her with the links in response to the entered keyword. For this purpose agents contact a search engine and produce recommendations by means of the Systems for Implicit Culture Support (SICS) module. This module uses implicit knowledge of the community members to find links that are considered relevant. Hereafter, by *relevant* links we mean links that are relevant to a certain keyword, from the agent's point of view. From the user's point of view, these links point to the relevant web pages. The framework that produces these links is universal in a sense that it is also exploited in order to discover which agents it would be useful to contact to obtain more relevant links. The general architecture of the system is represented in Figure 1.

Implicit consists of the client part and the server part. There is an html/php user interface on the client side. On the server side there are Java servlets and a multi-agent platform implemented using JADE (Java Agent Development Framework) [Bellifemine *et al.*, 2001]. JADE is a framework for developing multi-agent systems according to FIPA² standards. Here we present basic terms used in JADE and in our system.

A *personal agent* is an agent running on the server side that receives search tasks from its user and then produces recommendations in response to a query. The process of generating suggestions consists of several parts, implemented as behaviors. A *behavior* is a procedure that implements tasks, or intentions, of an agent. The agent is able to execute each task in response to different internal and external events. Behaviors are logical activity units that can be composed in various ways to achieve complex execution patterns and that can be concurrently executed. A *scheduler* is an internal agent component that automatically manages the scheduling of behaviors and determines which behavior to run now and what action to perform as a consequence. An *inbox* is a queue of incoming messages (ACL) from the user and from other agents. In order to produce recommendations agent uses its *resources* that consist of *beliefs* and *capabilities*. An agent's beliefs are the information available to the agent (e.g. information on user actions) and the capabilities are particular functionalities used in the behaviors (e.g. the SICS module). The structure of the personal agent is represented in Figure 2.

The basic sequence of actions while searching is as follows: a user logs into the system and enters a keyword. The interface generates a query message and sends it to the agent. When the agent receives the query message from the interface, it starts Search behavior. Search behavior produces results by means of internal (information about previous user searches) and external (communication with the agents) resources and these results are shown to the user.

The agent's Search behavior consists of the Google search behavior and the Platform search behavior, which comprises the Internal search behavior and the External search behavior. During the Google search behavior the agent process query

²FIPA. Foundation for Intelligent Physical Agents. <http://www.fipa.org/>.

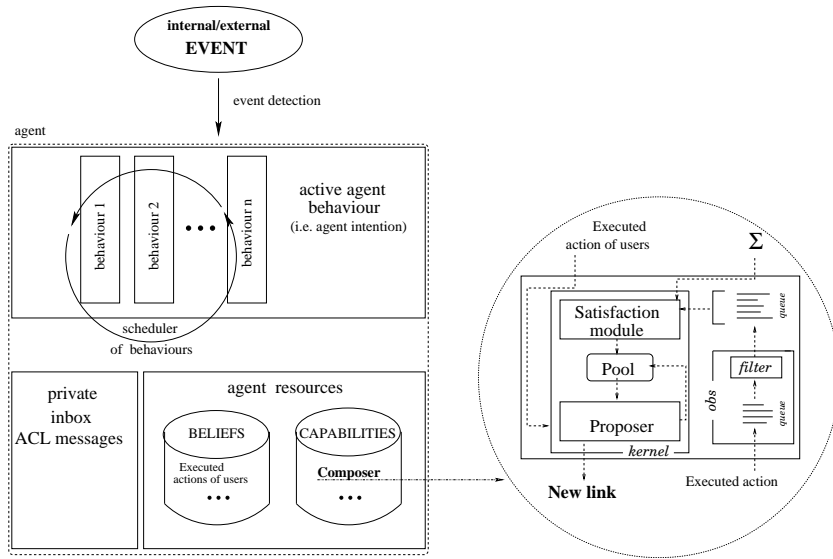


Figure 2: The internal architecture of the personal agent. Agent executes a *behavior* in response to different internal and external events. A *scheduler* manages execution of the behaviors. ACL messages received from the user or from other agents are stored in *inbox*. The *resources* consisting of *beliefs* (information available to the agent) and *capabilities* (available functionality) are used to produce suggestions. The *satisfaction module* selects links to the *pool* using behavior patterns produced by the *inductive module* from the observations on executed actions. The *proposer* selects the best link from the pool.

to Google search engine [Brin and Page, 1998] using Google Web API. As soon as the agent receives the answer, it shows the obtained links to the user and starts the Internal search behavior. In the Internal search the goal of the SICS module is to generate the http, ftp or resource links based on the past user actions. All the generated links are stored in the memory and the External search behavior is started. This behavior also uses the SICS, but the goal of the SICS in this case is to propose agents to contact. If there are no suggestions then agent contacts Directory Facilitator. Directory Facilitator (DF) according to the FIPA standards is a special agent that provides yellow pages service on the agent platform. Actually, in our case, DF simply provides the agent with the IDs of other personal agents on the platform. Having filled the list of agents to contact, personal agent starts an interaction — it sends a query to every member of the list. When all the agents are contacted the External search behavior queries new agents that were suggested during the search and so on. When all suggested agent queries have been answered the system shows all the obtained links to the user.

In the present implementation, the agent performs the three types of search in the following order: first the Google search, then the Internal search and finally, the External search. Agents may also query each other and in this case the respondent does not use the capability of contacting a search engine, because the questioner has this capability too. Agent-responder runs the Internal search behavior and uses its own observation history in order to produce links that the user of the agent-questioner will probably accept. It also starts the External search behavior in order to recommend to the questioner other agents to contact. The techniques used within these two behaviors are the same and are implemented within the SICS module.

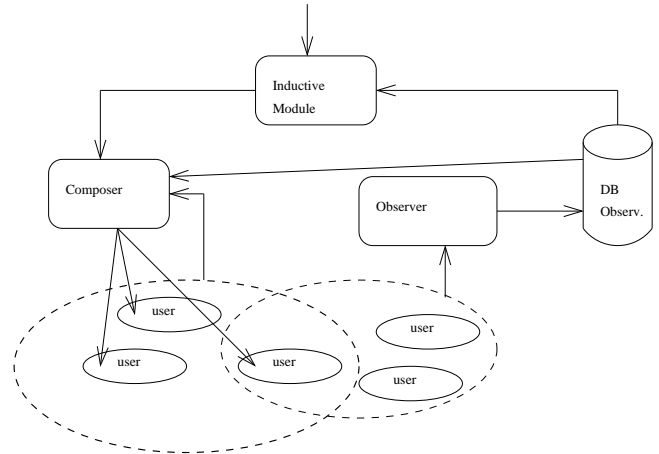


Figure 3: The System for Implicit Culture Support. The basic architecture for the System for Implicit Culture Support consists of the following three basic components: the *observer* that stores in a database (DB) the information about the executed user actions in order to make it available for other components; the *inductive module* that analyzes the executed actions in order to discover patterns of user behaviors; the *composer* that produces the links to suggest the user

The basic architecture for the SICS is shown in Figure 3 and consists of the following three basic components: the *observer module* is the part of the SICS that watches and records the actions performed by the user during the use of the system; the *inductive module*, analyzes the stored observations and implements data mining techniques to discover patterns in the user behavior; the *composer* exploits the information collected by the observer and analyzed by the inductive module in order to produce better suggestions to its user or to

other agents.

The SICS architecture requires the solution of two learning problems: a problem of browsing patterns learning (inductive module) and a problem of prediction of links the user will accept (composer). The inductive module problem is a rather standard learning problem: inducing the behavior patterns of the groups from the observations. The problem is not solved yet. The solution of the composer problem exploits the principles of instance-based learning (namely, memory-based or lazy learning). For more general description of these two problems see the work of Blanzieri et. al [2004].

The structure of the SICS allows the system to find out relevant links from the observations and to discover relevant agents using the same mechanism. The SICS calculates the similarity between the community members in order to produce suggestions. Therefore, it personalizes user's web search to a certain extent. For more detailed description of the SICS module, we refer the reader to the paper of Blanzieri et. al [2004].

Agents use Agent Communication Language (ACL) and standard FIPA protocols for link and agent ID exchange. There is also a feedback protocol for the exchange of information about accepted/rejected links. A feedback from one agent to another is sent as the result of the user browsing behavior. We illustrate the use of communication protocols by the following short example. More detailed description of the message passing and communication between agents can be found in [Birukov *et al.*, 2005].

For instance, a user searches information about "train timetable" and asks his/her personal agent, *pagent*. *Pagent* starts the Google search, the Internal and the External searches. After the Google search has finished the user has information about the links (we consider only the first three links for this example) provided by Google: *www.nationalrail.co.uk/planmyjourney*, *www.thetrainline.com* and *www.railtrack.co.uk*. The Internal search is then started in which the SICS module uses data mining techniques to select agents that performed similar actions and then selects the link accepted for the keyword "train timetable" by the agent with the highest similarity. During the External search behavior the SICS module selects agents that performed similar actions and chooses an agent likely to propose a link that will be accepted by the user. Let us suppose that SICS suggested the link *www.fs-on-line.it* during the Internal search and another agent to contact, *agent1*, during the External search. The personal agent sends a request to *agent1* using FIPA Iterated Contract Net Protocol. *Agent1* receives the request from *pagent* and uses its SICS module in order to produce suggestions. Let us consider that the Internal search behavior of *agent1* produced the link *www.trenitalia.it* selected from the links accepted by the *agent1*'s user in the past. As a result, *pagent* receives the link *www.trenitalia.it* and shows it to the user. If the user accepts the link *www.trenitalia.it* then *pagent* stores the information that this link has been accepted and sends this information (using feedback protocol) to *agent1* because it provided *pagent* with *www.trenitalia.it*. When the user leaves *Implicit* or starts a new search all the unaccepted links are considered rejected and all the agents involved in the dialog receive the

communication. In our example, if the user does not accept *www.trenitalia.it* then *agent1* receives the message that this link is rejected. One of the benefits of our approach is that feedback is collected without any effort from the user, such as giving ratings to the items or specifying his/her interests.

It is possible to have some special agents in the platform. Although each agent encapsulates the ability of contacting the external search engine, it is also possible to use agents called wrappers for transferring the queries to other search engines like Yahoo! or Vivisimo. The Agent Resource Broker (ARB) is the special agent whose main purpose is to provide personal agents with the links to the services available on other platforms (wrappers for example). The system can use some sort of the locally available knowledge, e.g. "yellow pages" reference or bookmarks.

3 Experimental Results

In this section we present the experimental results obtained with the proposed platform. We also define the measures (precision and recall) estimating the quality of the recommendations produced by the SICS.

The aim of the experiment is to understand how the insertion of a new member into the community affects the relevance, in terms of precision and recall, of the links produced by the SICS. We also want to check the hypothesis that after a certain number of interactions, personal agents will be able to propose links accepted in previous searches.

In our experiment, interaction between agents and models of users replaces interaction between agents and actual users. A user model contains sequence of search keywords and results about link acceptance. The results are among the first m links provided by Google for each keyword and the rank of the list is adopted as an identifier. The links provided by Google for a certain keyword are reordered very quickly, therefore before the experiment we store the links in a dataset. During the simulation we use the dataset instead of contacting Google. User profile is a set of probabilities of choosing a specified link for a specified keyword. The profile is built using n keywords k_1, k_2, \dots, k_n and determining the probabilities $p(j|k_i)$ of choosing the j -th link, $j \in \{1, \dots, m\}$ while searching with the i -th keyword. We assume that the user accepts one and only one link during search for the keyword k_i , so $\sum_{j=1}^m p(j|k_i) = 1$. The user profile can be seen as a set of association rules with a probability of acceptance of a certain link for a given keyword search. In our experiment the number of keywords n is equal to 10, the number of the links provided by Google, m is equal to 10, the user profile is represented in Table 1.

We use the following performance-related notions in order to evaluate the quality of the suggestions:

- Link is considered to be **relevant** to a particular keyword if the probability of its acceptance, as specified in the user profile, is greater than some pre-defined relevance threshold.
- **Precision** is the ratio of the number of relevant links suggested to the total number of irrelevant and relevant links suggested.

Table 1: Basic profile. The probabilities of acceptance links for a set of keywords. Links are numbered 1..10.

keyword	Google rank of the link									
	1	2	3	4	5	6	7	8	9	10
tourism	0	0	0.05	0.4	0.05	0.2	0.1	0.05	0.1	0.05
football	0.05	0	0.1	0.3	0.3	0.1	0.1	0.05	0	0
java	0.35	0.3	0.05	0.05	0.05	0.05	0.05	0.1	0	0
oracle	0.1	0.1	0.45	0.2	0	0.05	0.05	0	0	0.05
weather	0	0.3	0	0	0.5	0	0	0.1	0.1	0
cars	0	0	0.05	0.4	0.05	0.2	0.1	0.05	0.1	0.05
dogs	0.05	0	0.1	0.3	0.3	0.1	0.1	0.05	0	0
music	0.35	0.3	0.05	0.05	0.05	0.05	0.05	0.1	0	0
maps	0.1	0.1	0.45	0.2	0	0.05	0.05	0	0	0.05
games	0	0.3	0	0	0.5	0	0	0.1	0.1	0

- **Recall** is the ratio of the number of relevant links proposed to the total number of relevant links.

We compute recall in a slightly different way. The total number of relevant links is adjusted by adding a number of relevant links proposed by the agents to a number of relevant links presented in the user profile. We do it despite the fact that in reality the links from the agents already exist in the user profile, because in this way model of interactions becomes more similar to a real-life situation, where users (and their agents as well) have different collections of links. However, with such an interpretation of recall, the quality of system suggestions is underestimated.

Assuming that all the users are members of the same community and have similar interests, the profile for each user is derived from the basic profile given in Table 1. In order to make the users different, we add noise uniformly distributed in $[0.00, \dots, 0.05]$ to each entry of the profile. Then we renormalize entries in order to keep the sum of each row equal to one. Following this procedure we generate five different profiles.

From our set of 10 keywords for each agent we generate 25 sequences of 25 keywords by extraction with repetition. Each sequence is used for a search session modelling the user query behavior. We also need to model the user acceptance behavior. Given a keyword in the sequence of keywords, an accepted result is generated randomly according to the distribution specified in the profile. Other links obtained from the agents are marked as rejected.

In a simulation we run 25 search sessions for each agent in the platform. At the end of each session the observation data were deleted. The search sessions were repeated several times in order to control the effect of the order of the keywords and link acceptance. We run five simulations for 1,2,3,4 and 5 agents. With one agent in the platform, the agent acts alone without interactions with the others. With five agents there is a small community where agents interact with each other. We set the relevance threshold used to determine the relevance of links equal to 0.1.

We compute precision and recall of the links proposed by the agents. In Figure 4, line 1 represents precision of the links produced by the personal agent only. The SICS module incorporated in the agent produces these links by analyzing stored observations. Line 2 represents precision of the links proposed by all the agents including the personal one. The agents were discovered at the External search stage or provided by the DF. In Figure 5 we have analogous curves for

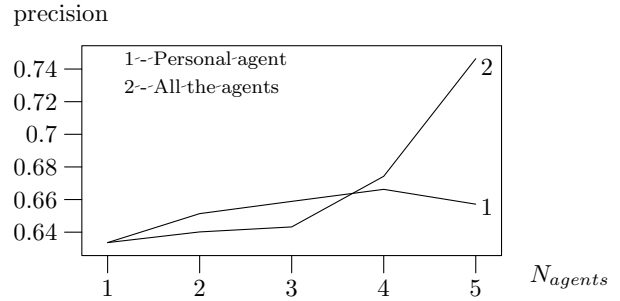


Figure 4: Average precision of 25 simulations with different number of agents.

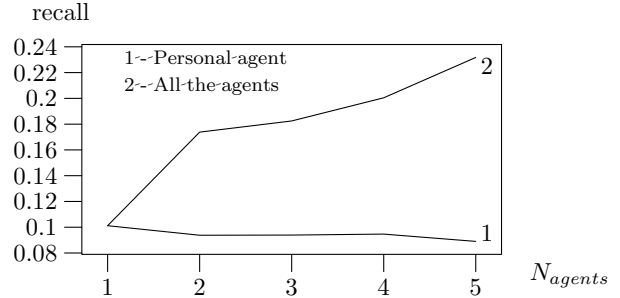


Figure 5: Average recall of 25 simulations with different number of agents.

recall.

From these figures we can note that the increase of community members causes the increase of the agents' recall. It is probably conditioned by the fact that when we have more agents we also have more interactions between them. The agents provide each other with only one link. The growth of the number of links provided by the agents during the search results in an increase of the percentage of relevant links proposed by the agents and causes an increase of recall. Moreover, the increase of recall appears without a decrease of precision and the precision keeps on a rather high level — from 0.63 to 0.75. The value of recall is also rather good and changes from 0.09 to 0.23. Because we limit ourselves to the small number of agents, the growing number of interactions does not really influence the characteristics. We also studied the statistical significance of the difference between agents with the same profile and in different simulations. We performed t -Tests with Bonferroni correction, namely dividing p -value by the number of tests we have performed, in order to control type I error. These tests prove that the average recall for 4 and 5 agents is consistently better ($p < 0.01$) than the average recall of the simulations with smaller number of agents. The results also prove the hypothesis that after a certain number of interactions, agents are able to propose links based on the past user actions.

In other words the obtained results prove that our method of complementing search engine with recommendations, produced as a result of collaboration, makes sense and allows a more qualitative web search.

For the moment we did not run yet any experiment for

a number of agents bigger than five. Therefore this paper contains only preliminary experimental results. We suppose, though we can not strongly claim that after a number of agents reaches a certain level, the increase of the community members causes only a moderate increment of the performance characteristics.

4 Related Work

In this section we briefly discuss related work.

A market-based recommender system is presented by Wei et. al [2003]. It is a multi-agent system where agent acts on behalf of its user and sells the sidebar space where recommendations can be displayed. Other agents participate in this auction in order to show their links on this sidebar. The agent-initiator of the auction chooses the most profitable offers and displays them to the user. Providers of the links accepted by the user receive reward. Agents adopt multiple heterogeneous recommendation methods and try to make better suggestions in order to increase their profit. The paper focuses more on the dynamic market behavior than on the recommendation quality evaluation.

A multi-agent recommender system is considered by Yu and Singh [2002]. MARS is a referral system for knowledge management that assigns software agent to each user. The agents interact in order to produce answers to the queries of their users. The agents are also able to give each other referrals to other users. There is a complex model of interactions in the system in a sense that it is important from who the query comes — there could be a different set of actions for the different agents. The system uses pre-determined ontologies, shared among all the agents, to facilitate knowledge sharing between them, while we emphasize the implicit support of knowledge by managing documents, links and references to people. Differently from our system, the agents do not answer all questions but only those related to their own user interests. The paper is focused more on knowledge (in general) search rather than on web search. Finally, the system is mail-based while *Implicit* is a web-based system that adopts FIPA standards and JADE platform.

Balabanović and Shoham present a recommender system Fab [Balabanović and Shoham, 1997] that combines collaborative and content-based filtering techniques. Personal selection agents analyze content of browsed web-pages and corresponding user ratings in order to maintain users profiles. Obtained profiles are compared using collaborative filtering algorithms and previously unseen items are recommended. Oppositely to using implicit feedback, the authors of this paper use explicit ratings, what requires a user to spend some time after browsing. Agents in Fab are divided into collection agents, who proactively gather pages relevant to a number of topics, and selection agents, who are dealing with discarding already browsed pages from the batch of the recommendations. The difference between Fab and *Implicit* is that our system filters not only links, but also agents. The framework we present is more general in a sense that different data mining algorithms can be implemented in order to produce recommendations. The ideas described in this paper can be deployed within different domains, e.g. Sarini et. al [Sarini

et al., 2004] describe application of Implicit Culture ideas to support the work of biologists in their laboratories. Yet another difference is that profiles in *Implicit* are not stored somewhere explicitly, but are spread around the agents and there is no explicit items ranking.

Freyne et. al [Freyne et al., 2004] describe I-SPY meta-search engine that re-rank search results by taking into account previous searches of the similar users. The system architecture differs from the architecture of *Implicit* significantly, but the goals and the techniques are very similar. The engine uses adapters in order to query several external search engines. These queries then pass through the component that re-rank search results according to the hit matrix of previous searches. The system tends to capture preferences of the users and therefore adapts to the community where it is deployed. While I-SPY has fully centralized architecture, *Implicit* is technically centralized, but conceptually it is distributed due to the fact that profiles are spread around the agents. It uses collaboration between the agents to improve results. We also focus more on an organizational community rather than on an emergent or online one.

A collaborative multi-agent web mining system “Collaborative Spiders” is given by Chau et. al [2003]. There are different types of agents responsible for retrieving web pages, performing post-retrieval analysis, interacting with users, sharing information about user search sessions, performing profile matching and carrying out retrieval and analysis tasks according to a schedule. Before a search the user has to specify the area of the interests and privacy or publicity of the search. One of the sufficient differences between this system and *Implicit* is that the user should analyze excessive output looking through a number of similar already finished search sessions.

Zhu et. al [2005] present WebICLite - a recommender system that uses behavior models to predict relevant web pages. They conceptualize web browsing as a search for a specific well-defined information need and make assumption that this need can be identified from the pages visited by the user and from the actions that he/she performs on the pages. Several specific algorithms for identifying information-need-revealing patterns are considered and compared. The algorithms are used in order to turn the inferences about the user information needs into the queries for a standard search engine which does the actual retrieval of recommended pages. The system is browser-integrated and reformulates a query of the user without any collaboration and communication between different users.

Macedo et. al [2003] apply a recommender system approach to assist and to augment the natural social process of asking for recommendations from other people. Web-Memex is a system that provides recommendations based on the browsing history of the people well-known to the users. To obtain the list of such users, a contact list from Yahoo Messenger is used. The system allows the user to keep privacy of web search by hiding his/her browsing for a certain time. The recommendations generated within the system are based on the links between the related documents visited by the users. On the server side there are no agents, but components that capture user behavior and generate recommendations.

5 Conclusion and Future Work

In this paper we have presented an agent-based recommender system that extracts implicit knowledge from user browsing behavior. The knowledge is necessary to suggest links or agents to a group of people and to their personal agents. Personal agents use universal mechanism to produce suggestions about links and agents IDs. Learning capabilities are used by agents to produce results even without an interaction. Interactions allow a user to use the already acquired experience of the members of his/her community. This increases the quality of the search. The process of collecting feedback and producing recommendations is completely hidden from the user and therefore does not require any kind of extra work from the user.

Implicit can be modified in several ways. It could be enhanced with the capability of analyzing content of visited web pages. In this way it would combine content-based and collaborative approaches. Classification of the users on “experts” and “novices” could also be implemented in order to take into account information about the author of the recommendation.

We use rather simple user model in this paper in order to test our system, and results presented here are preliminary. In the future, we plan to conduct some experiments with the participation of the real users.

6 Acknowledgements

This research is partially supported by COFIN Project “Integration between learning and peer-to-peer distributed architectures for web search (2003091149_004)”. The authors would like to thank anonymous reviewers for their helpful comments.

References

- [Balabanović and Shoham, 1997] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
- [Bellifemine *et al.*, 2001] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software - Practice and Experience*, 31(2):103–128, 2001.
- [Birukov *et al.*, 2005] Alexander Birukov, Enrico Blanzieri, and Paolo Giorgini. Implicit: An agent-based recommendation system for web search. In *Proceedings of the 4th International Conference on Autonomous Agents and Multi-Agent Systems*, 2005.
- [Blanzieri and Giorgini, 2000] Enrico Blanzieri and Paolo Giorgini. From collaborative filtering to implicit culture: a general agent-based framework. In *Proceedings of the Workshop on Agents and Recommender Systems*, Barcelona, 2000.
- [Blanzieri *et al.*, 2004] Enrico Blanzieri, Paolo Giorgini, Fausto Giunchiglia, and Claudio Zanon. Implicit culture-based personal agents for knowledge management. *Lecture Notes in Artificial Intelligence*, 2926:245–261, 2004.
- [Brin and Page, 1998] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.
- [Chau *et al.*, 2003] Michael Chau, Daniel Zeng, Hsinchun Chen, Michael Huang, and David Hendriawan. Design and evaluation of a multi-agent collaborative web mining system. *Decision Support Systems*, 35(1):167–183, 2003.
- [Freyne *et al.*, 2004] Jill Freyne, Barry Smyth, Maurice Coyle, Evelyn Balfe, and Peter Briggs. Further experiments on collaborative ranking in community-based web search. *Artificial Intelligence Review*, 21(3–4):229–252, 2004.
- [Gori and Witten, 2005] Marco Gori and Ian Witten. The bubble of web visibility. *Communications of the ACM*, 48(3):115–117, 2005.
- [Macedo *et al.*, 2003] Alessandra Alaniz Macedo, Khai N. Truong, Jose Antonio Camacho-Guerrero, and Maria da Graca Pimentel. Automatically sharing web experiences through a hyperdocument recommender system. In *HYPERTEXT '03: Proceedings of the fourteenth ACM conference on Hypertext and hypermedia*, pages 48–56, New York, NY, USA, 2003. ACM Press.
- [Sarini *et al.*, 2004] Marcello Sarini, Enrico Blanzieri, Paolo Giorgini, and Claudio Moser. From actions to suggestions: supporting the work of biologists through laboratory notebooks. In *Proceedings of 6th International Conference on the Design of Cooperative Systems (COOP2004)*, pages 131–146, French Riviera, France, 2004. IOSPress.
- [Wei *et al.*, 2003] Yan Zheng Wei, Luc Moreau, and Nicholas R. Jennings. Recommender systems: a market-based design. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 600–607, New York, NY, USA, 2003. ACM Press.
- [Yu and Singh, 2002] Bin Yu and Munindar P. Singh. An agent-based approach to knowledge management. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 642–644, New York, NY, USA, 2002. ACM Press.
- [Zhu *et al.*, 2005] Tingshao Zhu, Russ Greiner, Gerald Haubl, Bob Price, and Kevin Jewell. Behavior-based recommender systems for web content. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces. Workshop: Beyond Personalization 2005*, New York, NY, USA, 2005. ACM Press.

Trust and Bias in Multi-Agent Recommender Systems A Position Paper

Robin Burke and Bamshad Mobasher

School of Computer Science, Telecommunications and Information Systems

DePaul University

Chicago, IL USA

{rburke, mobasher}@cs.depaul.edu

1 Introduction

Recommendation systems are an increasingly important component of electronic commerce systems. Users have come to trust personalization and recommendation software to reduce the burden of navigating large information spaces and choosing from product catalogs. The preservation of this trust is important both for users and site owners and is dependent upon the perception of recommender systems as objective, unbiased and meaningful. However, because recommendation systems are dependent on external sources of information, they are vulnerable to attack. If a system generates recommendations collaboratively, that is by user-to-user comparison, hostile users can generate large numbers of biased user profiles for the purpose of distorting the system's recommendations. If on the other hand, the system uses properties associated with products, producers may label their items deceptively to generate more recommendations.

Consider a recommender system that identifies books that users might like to read based on a collaborative algorithm. Alice, having built up a profile from previous visits, returns for new recommendations. Unbeknownst to her, Eve has inserted a large number of biased profiles into the system, all of which give high ratings to a book she has written. If this so-called "push attack" is successful, Alice will get Eve's book as a recommendation, regardless of whether this is really the best suggestion for her or not. She may find the suggestion inappropriate, or worse, she may take the system's advice and then be disappointed by the delivered product.

This is the essence of the problem of secure recommendation. In fact, this particular example has currency: the Associated Press reported in February of 2004 that a security glitch at the online book store Amazon.com had caused the real identities of its book reviewers to be made public, and buyers found that many authors were indeed posting glowing reviews of their own work [AP, 2004]. How can a recommendation system, which is dependent on user-generated profiles, provide trustworthy advice? Recommender systems that are not protected against attack will have diminished value and will not achieve the goal of providing unbiased guidance.

Recent research has highlighted ways in which attackers may inject bias into collaborative filtering recom-

mender systems, causing them to produce particular desired results, for example to make a particular product more highly recommended [O'Mahony *et al.*, 2004] [Lam and Reidl, 2004] [Burke *et al.*, 2005] It is clear that other types of recommender system are also susceptible to similar forms of attack. Because of the possibility of bias, a multi-agent recommendation paradigm raises in particular the problem of trust: can a particular recommendation component be trusted? Our research is investigating the vulnerabilities of different recommendation algorithms (including hybrid ones) to different kinds of bias and examining how recommender systems can detect and compensate for such attacks.

2 Knowledge Sources

To examine this question further, we can consider the knowledge sources that recommender systems typically use in making recommendations. There are six knowledge sources that can be identified:

- R_c : the rating history of the current user C , a record of C 's preferences over the products being recommended;
- R : a database of all users' ratings: the set of R_u for all users u .
- P_c : demographic or other personal data about the current user.
- P : the demographic data for all users.
- D : the database of features associated with the products.
- K : a knowledge base about the products, their features and possibly the way that products satisfy those needs.

Different types of recommender systems use different knowledge sources. Classic collaborative filtering recommendation is performed with the use of R_c and R . Content-based recommendation requires R_c and D . Demographic recommendation uses P , D , and P_c . Knowledge-based recommendation uses D , K and sometimes R_c and P_c .

3 Bias

Bias in a recommender system is the distortion of the output of the system through the injection of false information. Bias may enter into the recommendation process from any one of these knowledge sources. The most studied form of recommender system bias is one in which false profiles are created, biasing R so that collaborative recommendations are affected. In domains where responsibility for labeling of products is distributed, it is possible for bias to creep into the feature database D . If we consider web search as a degenerate case of knowledge-based recommendation, the problem of "search engine spam", the labeling of documents with false or misleading terms, is a form of bias of this kind.

Bias can be caused by other knowledge sources as well. Users would seldom have reason to falsify their own rating histories, but it is often the case that more than one individuals' tastes may be represented by a rating history (as in the case of gift-giving or sharing accounts) and this constitutes a form of bias. As far as demographic data is concerned, it is well known that users frequently falsify their personal data when e-commerce sites request it. These forms of bias do not have the targeted quality of the attacks that are our primary concern.

It may be that an effective response to the problem of bias in one knowledge source is to build a hybrid recommender system [Burke, 2002] that combines multiple knowledge sources. For example, designers confronting a system with biased content may use collaborative information to dilute its impact. Much of the success of the Google search engine¹ can be attributed to its use of an authority measure (effectively a collaboratively-derived weight) in addition to standard content-based metrics of similarity in query processing [Brin and Page, 1998]. This hybrid technique means that a page that is misleadingly labeled is much less likely to be retrieved by Google than by a system that uses only content. Google is therefore an example of a hybrid approach to overcoming bias in D , defending against a biased content attack through the addition of collaborative information from R .

4 Multi-agent Recommendation

Consider a multi-agent recommendation scenario in which a given recommendation is computed based on input from a set of distributed agents, each of which may have its own data source. For example, agents may have disjoint sets of profiles, which together make up R , or the agents may specialize, for example, in a content-based recommender one agent might contain D and another R . Depending on the configuration and policy of the system, it may be that such an agent is less vulnerable to attack than a stand-alone system: an attacker might have to identify the distributed components that it uses and may have to attack several of them in order to achieve the desired result.

¹www.google.com

On the other hand, however, a new type of bias is introduced, namely the possibility of a hostile component. A compromised recommendation agent within the multi-agent system may be able to wreak considerable havoc, perhaps more than any bias injection attack. The establishment and maintenance of trust relationships between recommendation agents is therefore a key consideration for system design.²

In a more general sense, the recommender system and its users (including the attacker) can be considered together as a multi-agent system. Any practical attack would have to be carried out by one or more software agents interacting with the recommender to insert artificial profiles. The distributed agent perspective may prove to be the most fruitful way to analyze the impact of multiple attackers that may collude or compete to achieve their respective goals for the system.

5 Conclusion

Recommender systems (indeed open personalization systems in general) are vulnerable to attack by hostile users whose biased data cannot be easily distinguished from that originated by ordinary users. Recent research has established that the most widely-used algorithms are in fact quite easy to attack in this way. It appears that hybrid systems that contain components of different types may have security advantages – a finding that bolsters the case for distributed heterogeneous multi-agent recommendation. However, the possibility of hostile agents within the system itself cannot be ignored. A crucial factor in the robustness of multi-agent recommender systems will be the mechanisms for establishing and maintaining trust among the agents that collaborate to produce recommendations.

References

- [Brin and Page, 1998] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998.
- [Burke et al., 2005] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik. Identifying attack models for secure recommendation. In *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems Research*, pages 19-25. IUI, 2005.
- [Burke, 2002] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331-370, 2002.
- [Lam and Reidl, 2004] S. Lam and J. Reidl. Shilling recommender systems for fun and profit. In *Proceedings*

²The trust issue that we identify here is different from the sense of trust sometimes described in distributed recommendation contexts, where the question is who can be trusted with users' personal data [Ramakrishnan et al., 2001]. The question our work raises is the question of what recommendation components in a distributed system can be trusted.

of the 13th International WWW Conference, pages 393–402, 2004.

- [O’Mahony *et al.*, 2004] M. O’Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology*, 4(4):344–377, 2004.
- [AP, 2004] Associated Press. Computer glitch identifies anonymous amazon reviewers. eWeek, Feb. 14 2004.
- [Ramakrishnan *et al.*, 2001] N. Ramakrishnan, B. J. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62, 2001.

Language Games: Learning Shared Concepts Among Distributed Information Agents

Conor Hayes and Paolo Avesani and Marco Cova

ITC-IRST,

Via Sommarive 18 - Loc. Pantè, I-38050 Povo, Trento, Italy

hayes, avesani, cova@itc.it

Abstract

Early agent research recognised that co-operating agents require access to unambiguous, semantic description of the same concept, entity or object. In fact, agent-based research on this problem anticipates many of the current initiatives of the Semantic Web project. The proposed solution involves developing a domain-specific ontology that can be mapped to other ontologies as required. In this paper we describe an alternative approach which allows autonomous agents to index shared objects without requiring ex-ante agreement on an ontology. Using a process of distributed negotiation, each agent builds a lexicon of the problem-solving competences of other agents. We present an overview of our work using this approach in three domains: a web services scenario, a multi-case-based agent approach and finally, Tagsocratic, a blog-indexing service. We then describe our future work on several open issues related to this research.

1 Introduction

The problem of interoperability between autonomous, non-centralised software components has been an intrinsic feature of agent research. Although agent communication languages (ACLs) such as KQML [Finin *et al.*, 1994] and FIPA [FIPA, 2000] provide standard communication protocols, there is still the problem of agreement on a common language with which to allow agents to co-operate about content in their environment. One of the key incentives of the Semantic Web project was to introduce a semantic framework that would enable the processing of Internet resources by intelligent entities such as software agents. In pursuit of this, RDF and related initiatives such as DAML and OIL allow domain-specific ontologies to be specified [McGuinness *et al.*, 2002]. The usual strategy of these efforts consists in establishing a relationship between the local representations and a common reference encoding, namely a shared ontology. This approach requires two steps:

1. The definition of an ontology for the specific domain.
2. The definition of a mapping between a local representation and the shared ontology.

While intuitive, these approaches are often not effective in practice. The first step requires ex-ante agreement between all potential users of the ontology. Furthermore, the mapping step is generally far from trivial and often requires manual intervention. Indeed, in terms of take-up on the WWW, the top-down proposals of the Semantic Web have been less successful than simpler bottom-up protocols such as RSS,¹ which enable information providers to publish information in standard form quickly without having to agree on semantics in advance. The latter scenario, however, still implies ex-post agreement on semantics between locally defined representations. In terms of agent technology, this means allowing the agent complete autonomy in representing the information objects it expresses, but also requiring it to learn a mapping between alternative representations held by other agents.

In contrast, we present an alternative perspective in which autonomous, distributed agents negotiate using a technique called *language games* in order to develop a distributed indexing lexicon [Steels and McIntyre, 1999]. By learning this lexicon each agent builds a picture of the competences of external agents and can quickly request resources from these when it is not able to solve a problem locally. To illustrate the technique we present three scenarios in the area of distributed information retrieval. In the first, a web services plug-in allows distributed information providers to learn which topics they have in common. In the second, distributed case-based information agents learn the competences of each other so that information that is not available locally may be retrieved from other competent agents. Finally, we introduce Tagsocratic, a project in the blogging domain in which agents communicate in order to learn topic alignments between bloggers. In this scenario, the blogger is enabled to quickly find posts by other bloggers on the same or similar subjects.

In section 2, we locate this work in relation to the Semantic Web project and to previous work in agent research. Sections 3 and 4 describe the language games technique. We characterise this technique as having a two stages: an eager indexing stage and a problem-solving stage. In section 5, we describe our initial work in applying this technique in three domains. We describe open issues and future work in section 6.

¹blogs.law.harvard.edu/tech/rss

2 Background

Early agent research recognised that co-operating agents require access to unambiguous semantic description of the same concept, entity or object. For example, the DARPA Knowledge Sharing Effort (KSE) tackled the problem of interoperability between heterogeneous knowledge sources by dividing the problem into three layers: a language translation layer, a communications layer and an ontology layer in which semantic consistency is preserved within the domain in which the applications work [Neches *et al.*, 1991]. The key idea is that a shared domain-specific ontology is developed in advance from which agents can choose or extend the elements that best suit their own perspective. Knowledge can be shared because agents can translate between their own local representation and the shared ontology. Agent communication languages such as KQML and FIPA assume a shared ontology between communicating agents. In KQML, the message layer allows the agent to specify the ontology associated with the message, while FIPA has specified an ontology service to allow agents to reason about domain knowledge.

Indeed, one of the key objectives of the Semantic Web project is to enable processing of web resources by distributed, intelligent entities such as software agents [Berners-Lee *et al.*, 2001]. As such, this project has produced several specifications such as RDF, OIL and DAML, which allow domain-specific ontologies to be produced. The Semantic web proposals can be viewed as a top-down approach to the problem of semantic agreement: agreement is reached in advance on the formal relations between entities in a particular domain, after which agents or other intelligent software applications can reason about the objects in the domain. However, a difficulty in this approach is how agreement is reached on the correct knowledge representation for a particular domain. If two or more ontologies are used for a particular domain, agents who wish to communicate will require a translation service between ontologies. In the worst case scenario, where agreement is not reached, each agent uses a knowledge representation based on local semantics and communication between agents requires a translation service between each pair of agents.

Despite the obvious benefits of an agreed semantic framework, the take-up on Semantic Web proposals to date has been slow. Instead, simpler, ‘bottom-up’ initiatives have become much more successful. For example, RSS has become the standard means of allowing information providers to publish up-to-date data on the web without requiring explicit semantic mark-up. In the blogosphere, blog software enables bloggers to mark-up each of their posts with locally defined categories. Two key observations can be made here: the proliferation of these ‘bottom-up’ approaches appears to be stimulated by the lack of centralised co-ordination required for their deployment. This would seem to suggest that web content providers prefer minimal constraints on the local definition of semantics. Secondly, the issue of semantic alignment appears to be addressed by a second wave of low level initiatives such as RSS aggregators and category aggregators².

²<http://www.technorati.com/>

There are also a number of approaches to matching heterogeneous schemas based on machine learning [Rahm and Bernstein, 2001]. However, where agents have heterogeneous knowledge representations, such approaches require a mapping between representations for every pair of agents. In a similar spirit to this work, [Reed *et al.*, 2002] argue that the specification of the agent communication language (ACL) can be a run-time process where agents tailor their communication primitives to the circumstances in which they find themselves. In contrast, we are concerned with how distributed information agents can learn to refer to common objects in their environment without having to formally define or learn a particular semantic framework.

3 Introduction to Language Games

Our research is based on experiments on language evolution called *language games* [Steels and McIntyre, 1999]. Rather than requiring pairwise translation between heterogeneous knowledge representations, agents achieve a common language through a process of exchanging instances and feedback. The language games technique was developed originally to explore how autonomous, heterogeneous physical agents could develop a common language in order to communicate about objects in their environment. Agents are not constrained in terms of their internal representation about the objects. Instead, they learn a distributed common lexicon which allows them to refer to common objects in their environment although each agent has a different internal representation.

This technique has more recently been adapted to tackle the problem of aligning heterogeneous knowledge representations among information providers on the web [Avesani and Agostini, 2003; Avesani and Cova, 2005]. The problem scenario can be described as follows: The web is a distributed repository of web pages to which autonomous agents (users) add information in a random fashion. Although, many web pages describe related subject matter, the web is not organised in a topic-centric manner. We describe how distributed agents co-operate by game playing in order to produce a distributed index of topics that allows an agent to link its own topic definitions to those defined by agents with similar interests.

Our definition of an agent is quite varied. In the Moleski application, an agent is a server side add-on that learns the correspondence between ski-mountaineering trips being offered by other autonomously managed web sites [Avesani and Cova, 2005]. In the CBR agents initiative [Avesani *et al.*, 2005b], each agent filters and collects sites offering travel itineraries for a particular user interest group. A user can query the agent’s case base memory for a suitable trip description. If an adequate solution is not found, the agent contacts other agents with competence in the area. The key point is that each agent learns and indexes the competences of external agents using the language games technique. Finally, the Tagsocratic project examines the role for information agents in the Blogosphere [Avesani *et al.*, 2005a]. In this scenario, each blogger has an agent who learns which posts by other bloggers address similar topics to those of its blog master.

4 Indexing and Problem Solving

In this section we give a top level view of the language games methodology. For a more formal introduction to our methodology see [Avesani and Agostini, 2003]. The language games methodology can be viewed as being made up of 2 stages:

1. Indexing phase: An eager index learning stage in which agents communicate to assess similarity between their object representations. During this phase a shared distributed index emerges.
2. problem-solving phase: In this stage each agent can quickly retrieve relevant cases by consulting its indexing lexicon and issuing a label identifier to other agents.

4.1 Indexing phase

The indexing phase consists of a number of language games whereby a community of agents converge on a common set of labels for the objects in their world. The common set of labels constitutes a distributed global lookup table which is then used to reference similar cases in external case bases. For example, if we consider the CBR example previously mentioned and described in more detail later, the index learning phase involves each agent learning the correspondence between similar cases in other agents and assigning those cases a global identifying label by which those cases can be identified. The set of global labels constitutes an index of the global competence of the agent community. However, this index is not maintained centrally but is distributed among the agent community. After the indexing process, an agent can request similar cases to one of its own cases from another agent by simply sending it the appropriate global label. The receiving agent then looks up which case(s) in its case base corresponds to this label and returns it to the requesting agent.

Each language game involves an exchange between a pair of agents where one acts as a speaker and the other acts as a listener. As shown in Table 1, all agents maintain a table which records the degree of association between a global label (\mathcal{L}_p) and one of its cases (\mathcal{O}_p). In the table, u refers to the number of times the label has been used in different language games by this agent while a refers to how often it has been successfully used. Each game consists of a communications phase and an assessment phase (see Figure 1).

Communications

The *communications* phase proceeds as follows: The speaker agent chooses one of its object (case) representations and the corresponding best scoring label from the association table. In the example in Table 1, the label l_1 is the best performing label for case c_{j9} , having been successfully used for 8 out of 10 language games. Agent CB_j and agent CB_i agree to have a language game. Agent CB_j , acting as the speaker, selects case c_{j9} and (encodes it as) label l_1 . It sends the label to Agent CB_i . CB_i decodes label l_1 using its association table. If it finds that the label corresponds to one of its own cases, it selects the case and returns it to agent CB_j .

Assessment

The next phase of the game is the *assessment phase*. Agent CB_j must assess whether the case sent by agent CB_i is equivalent to case c_{j9} . In the Moleskiing application and the

\mathcal{O}_p	\mathcal{L}_p	u	a
c_{j9}	l_1	10	8
c_{j9}	l_2	3	0
c_{j5}	l_3	5	4
c_{j6}	l_4	8	1

Table 1: The lexicon of case base CB_j during the learning phase

CBR agents application we successfully use a bi-partite string matching algorithm to determine equivalence [Kuhn, 1955]. The cases in the speaker’s case base are ranked by similarity to the case received from the listener. If the top ranked case is the same as the case initially chosen by the speaker, the game is deemed to be a success and the speaker accordingly updates its table, increasing the fraction of times the label was successfully deployed (9 times out of an 11). The speaker sends positive feedback to the listener so that it too can update its table. If the game fails - the listener may return nothing or may return a mismatching case, the fraction of successes recorded by the label is reduced to 8 out of an 11. The speaker sends negative feedback to the listener and likewise it reduces the fraction of successful deployments for this label in its table.

It is important to remember that the assessment phase is domain dependent. For the Tagsocratic project, for instance, we use a naive Bayes classifier [Lewis, 1998] to determine whether the posts received from the listening blog agent are of the same class as the blog posts represented by the global label on the speaking blog agent. In this sense, the assessment phase in Tagsocratic is more difficult in that we have to assess whether exchanged posts belong to the same concept shared by both peers, rather than determining equivalence between object representations.

4.2 Problem-solving Stage

The problem-solving stage allows distributed CBR agents to quickly retrieve remote cases by issuing the learned index label. As similarity computation has been eagerly computed and indexed during the indexing stage, the bandwidth and computation overheads involved in sending and decoding a label are very low, entailing fast query time response.

Let us consider an example scenario in which CBR agents retrieve travel information based on input from a user. Each agent operates autonomously and has a case base made up of a set of travel itineraries. The solution part of each case description consists of reviews and comments posted by other users on the quality of the proposed travel package. Each case base is represented according to a locally defined schema. CBR agents may cooperate so that if a locally retrieved solution is not adequate it can contact agents with competence in the problem area to retrieve alternative solutions. In Figure 2 we illustrate the cycle just described. Each case base agent contains reviews that are pertinent to a particular interest group. While not delivering results to its user base, the agent is busy crawling the web for case material relevant to the interests of its user group. Let us consider the scenario where, after querying case base agent CB_i , the user provides

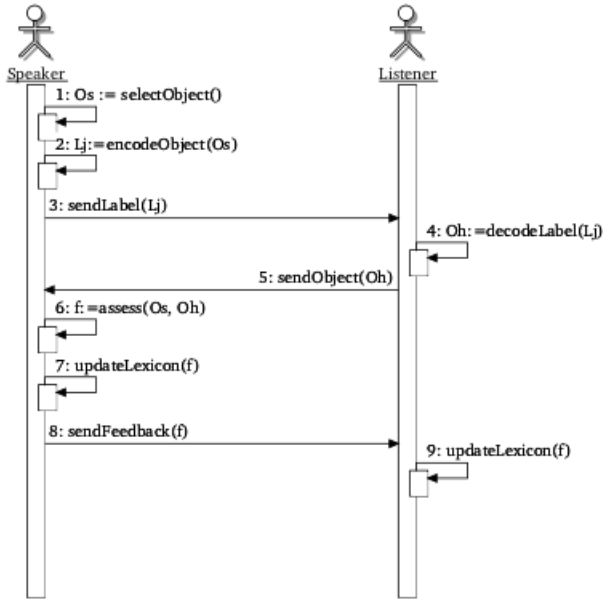


Figure 1: Language games interaction model.

relevance feedback to the agent that the solution of the retrieved case c_{i1} was not adequate: in our example we can see that the itinerary is not reviewed in sufficient detail. In response the agent looks up the case being inspected by the user in its association table. It finds that it corresponds to the lexical label l_1 . It then issues a request for solutions for l_1 to the other agents that contain cases indexed as l_1 . Each agent decodes the label l_1 and return cases associated with it. In the example shown in Figure 2 the solution for case c_{j9} from case base CB_j is returned to the user.

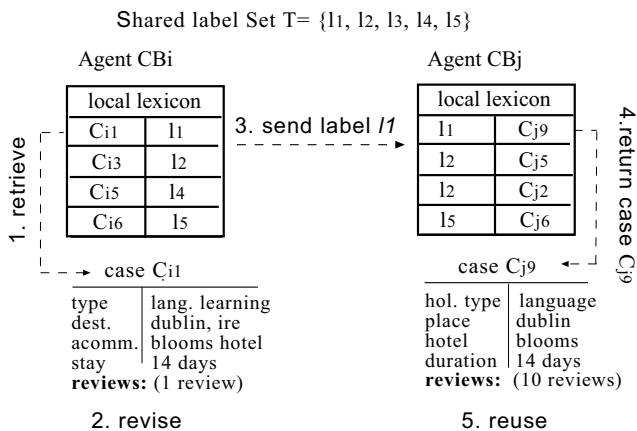


Figure 2: The retrieval model for distributed CBR agents.

5 Applications

5.1 Case-Based Information Agents

The previous section introduced the language game phenomenon as applied to a community of distributed case-based agents. A detailed discussion of how language games solve the vocabulary alignment problem in distributed CBR can be found in [Avesani *et al.*, 2005b]. Figure 3 illustrates the results from an evaluation where we used a real data set from the Harmonise project, a European project to align services from heterogeneous tourism service providers [Fodor *et al.*, 2002]. To enable interoperability between different representations of the same event, the Harmonise researchers propose manually mapping each vendor's schema to an intermediate schema. Our evaluation goal was to see whether we could automatically align the same events using the language games methodology. Our data set consisted of 6 events represented four different ways by four case-based agents. Figure 3 illustrates that 100% alignment was achieved soon after 800 pairwise games.

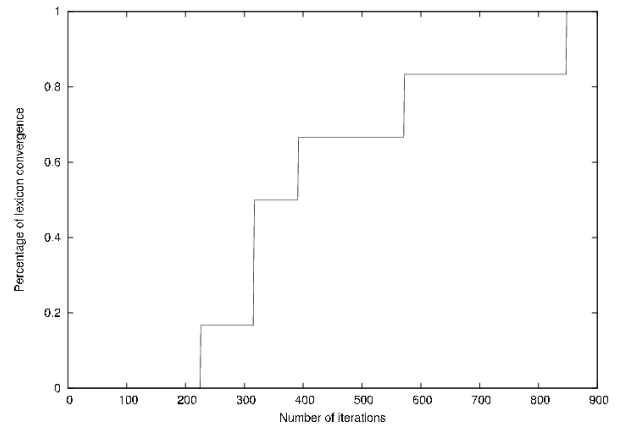


Figure 3: Formation of an indexing lexicon

5.2 Moleskiing

The Moleskiing initiative represents the authors' initial attempts at tackling the problem of the lack of implemented semantic standards on the web. In the sport of ski-mountaineering it is crucial for participants to have the most up to date reports on route conditions from other skiers. However, there are several ski-mountaineering portals to which alpine skiers can report conditions which tends to fragment the information available to other skiers. Moleskiing was designed as service to automatically reconcile the different schemas used by each portal to provide a single point of access on Alpine ski-mountaineering conditions. Three heterogeneous sources of ski-mountaineering information were used: Gulliver, Moleskiing and Skirando. Table 2 summarises the data from the three information providers that we used in our evaluation.

Figure 4 shows the plot of four sample game sessions. It shows the percentage of lexica convergence as a function of

	<i>gulliver</i>	<i>moleskiing</i>	<i>skirando</i>
Total items	38	179	69
gulliver overlap	-	22 (12%)	8 (11%)
moleskiing overlap	22 (57%)	-	51 (73%)
skirando overlap	8 (21%)	51 (28%)	-
Complete overlap	6 (15%)	6 (3%)	6 (8%)

Table 2: The lexicon of case base CB_j during the learning phase

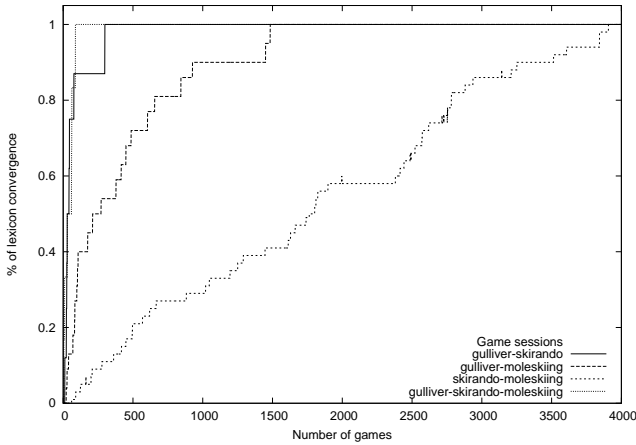


Figure 4: Evolution of common lexicon formation.

the number of games played between peer agents. 0% convergence means that there is no common associations among the peers: every peer is using different labels to encode the same object. Thus, the common lexicon is empty and inter-peer communication will fail. Conversely, 100% convergence indicates that peers have reached an agreement on how to reference all shared objects. The common lexicon contains one entry for each shared object and thus inter-peer communication is always successful. A full discussion of these results as well as a description of the service-oriented architecture for language games used in Moleskiing is given in [Avesani and Cova, 2005].

5.3 Tagsocratic

Weblogging has increasingly become an important part of the information economy found on the Internet [Nardi *et al.*, 2004; Schiano *et al.*, 2004]. Its great benefit is that it allows ordinary people to easily publish opinions based upon their experiences. This type of information, sometimes highly subjective, has great value for other Internet users who can make use of it to make decisions or simply to inform themselves. However, the blogging phenomenon exacerbates the problems posed by the lack of semantic protocols for the Internet. Although there is no constraint on what information can be posted, blogs often take the form of a series of annotations on topics of shared interest [Bar-Ilan, 2004]. As bloggers tend to publish their work independently, there is no standard way

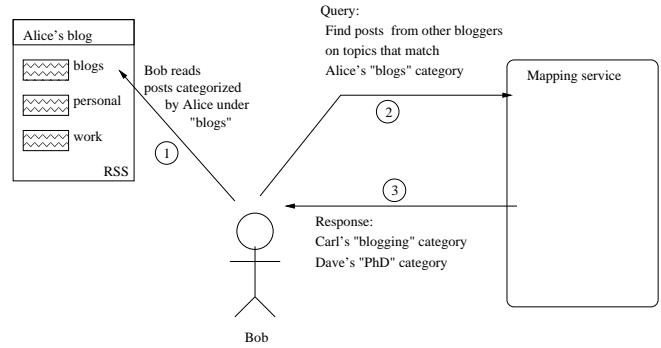


Figure 5: Tagsocratic use case.

of organising the blogosphere so that the posts that relate to a particular topic can be automatically indexed together. Most blog software allows users to define categories with which to label their posts. However, the semantics of the category are defined locally by the user rather than relating to a globally understood concept.

Clearly, there are benefits if these distributed information sources can be organised so that the reader (or blogger) can view related opinions on a single topic or concept. For example, the prominence given to user reviews on proprietary review sites like Amazon.com suggests their importance in providing sales advice to the potential customer. In Figure 5 we present a use case of the type of topic-centric service we require for the blogosphere. The objective is to provide an on-line mapping service for locally defined blog entry categories.

In the use case scenario depicted in Figure 5, Bob is visiting Alice's blog. He finds posts about the activity of blogging and notices that Alice categorises them under the category *blogs*. Bob would like to view other posts available in the blogosphere about the same topic. The problem, of course, is that other bloggers may use different categories to describe the *blogging* topic. Thus, Bob contacts the Tagsocratic service, requesting blog entries from categories mapped to Alice's *blogs* category; the mapping engine then returns a list of entries from categories aligned to the *blogs* category. The returned matches include entries from Carl under his *blogging* category and entries from Dave labelled *PhD* (Dave is doing a PhD on the effect of blogs on society). However, Bob does not receive any entries from Eve whose *blogs* category simply stores links to various blog engine web sites.

Thus, the objective of the Tagsocratic project is to provide an on-line matching service for local blog categories whilst respecting the autonomy of the blogger. Our goal is to allow a user to find posts categorised by other users under local categories that are semantically equivalent. In our approach, the semantics of other users' categories are automatically learnt by the system using the language games technique. The usage patterns of the user (which we call local context) are taken into account. This allows us, for example, to handle situations where two bloggers use the same category label with totally different meanings. From the functional point of view, Tagso-

cratic tackles the situation presented in the use case (see Figure 5). The issues involved in developing the Tagsocratic service are discussed in greater detail in [Avesani *et al.*, 2005a];

6 Future Work

We have begun examining how to reduce the number of game iterations required in order to converge on a stable lexicon. One area that can be improved is the initial period of game activity where the speaker sends a label which must be guessed by the listening agents. Clearly, many games must take place before the listener guesses correctly. One solution we have had initial success with is for the speaker to send an instance rather than a label. Secondly, we are examining the strategy used by the speaker in choosing labels or instances to send. Currently, these are chosen at random. However, it might be more efficient to send labels/instances that would provide more discriminating information to help in the formation of the local lexicon.

We have also begun to examine how different strategies can be employed by the peer agents. Currently, the strategy used by agents is naive: all agents are good potential game partners and partners are chosen at random for this purpose. However, we recognise that a more sophisticated strategy would be to choose partners that give the peer maximum information exposure (i.e. peer agents that service large communities) and partners that are consistently good sources of information. We describe some directions we are examining to allow agents to operate using more sophisticated strategies.

6.1 Mixed-initiative strategies

A mixed-initiative system is a system which allows more user interaction in the automated reasoning process. The key insight is that humans may be better equipped to assess critical points during the learning phase and should be enabled to contribute. By integrating the contributions from the user and system, we enable each to contribute what it does best. Moreover, flexible user-interaction policies would allow the system to adapt to differences in knowledge and preferences among users. In the context of language games research, we are investigating how a mixed-initiative strategy could be unobtrusively employed to speed up the convergence step by providing feedback on ambiguous lexical alignments during the learning phase. Furthermore, user interaction can help to narrow the scope of the game by selecting candidate players or barring further games with agents whose information services they distrust or dislike.

6.2 Trust/reputation strategies

The issue of trust and reputation on the Internet has become increasingly important, not just in terms of sales reliability but also in terms of implying consistency and authority [Richardson *et al.*, 2003]. For instance, Google's PageRank algorithm implicitly recognises highly linked pages as more likely to be authoritative sources of information [Page *et al.*, 1998]. Likewise, in the community of heterogeneous information agents we have described, certain agents are likely to emerge as authorities on certain topic areas. Thus, rather than choosing partners at random, an agent may have more success in linking his topic descriptions to those expressed by

such agents. We are interested in developing the protocols that allow reputation scores to be expressed and understood in a distributed environment.

6.3 Malicious recognition strategies

A related issue is how to recognise spam. At the present time, topic relevance is determined using classification and similarity matching techniques. However, spammers have shown ingenuity at gaming pattern recognition software and we would expect the determined spammer to be able to poison the distributed index with reference to non-relevant products and services. Our goal is to be able to detect spam early through a process of reputation metrics and pattern recognition.

6.4 Evaluation environment

A key issue is the evaluation of our system both in an off-line and on-line context. There are three aspects to this: Firstly, we recognise that there is some correspondence between learning a distributed topic index and typical unsupervised learning techniques. The language games approach is novel in that it can be viewed as an unsupervised learning approach where the training corpus is decentralised. In terms of learning efficacy we are examining how we can formalise the language games approach so that it can be compared with typical centralised approaches to clustering.

Secondly, we are interested in developing a games simulator where we can test the languages technique using a game theoretic approach. Thus we can assign agents differing strategies and observe which strategies perform better. At the moment our objective function of game success is a global one indicating the percentage of global convergence. We recognise that we need to develop a more fine-grained function to measure overall game outcomes for individual agents and communities of agents.

Finally, we need to test our applications in an on-line setting. This is particularly important for evaluating the mixed-initiative strategy.

7 Conclusions

In this paper we investigated the problem of building the distributed common reference systems needed to enrich current web applications and allow for their meaningful interoperability. We considered this problem from the perspective of a community of distributed agents. Whereas the Semantic Web proposes a top-down approach to Semantic interoperability, we suggest that agents can learn the competences of other agents in their community. We described a novel approach to this problem based on the language games technique. We introduced three prototype applications we have developed to test this methodology: CBR agents, Moleskiing and the Tagsocratic project.

There is wide scope for future work. The model underlying the language games technique is still fairly unsophisticated and we plan to use the experience gained from practical experimentation to improve it. Along the same line, we expect to design more refined strategies to guide the games, in order to improve the lexicon building process.

Acknowledgments

This work was partially funded by the DiCA project, thanks to a grant from the (Istituto Nazionale Ricerca Montagna) and by the PeRec project funded by the PAT (Provincia Autonoma di Trento).

References

- [Avesani and Agostini, 2003] P. Avesani and A. Agostini. A Peer-to-Peer Advertising Game. In M. Orlowksa, M. Papazoglou, S. Weerawarana, and J. Yang, editors, *First International Conference on Service Oriented Computing (ICSOC-03)*, pages 28–42, Berlin Heidelberg, 2003. Springer-Verlag LNCS 2910.
- [Avesani and Cova, 2005] P. Avesani and M. Cova. Shared lexicon for distributed annotations on the web. In *Fourteenth International World Wide Web Conference*, Chiba, Japan, May 2005. Association for Computing Machinery (ACM).
- [Avesani et al., 2005a] P. Avesani, M. Cova, C. Hayes, and P. Massa. Learning contextualised weblog topics. Chiba, Japan, May 2005. WWW 2005 2nd Annual Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics.
- [Avesani et al., 2005b] P. Avesani, C. Hayes, and M. Cova. Language games: Solving the vocabulary problem in multi-case-base reasoning. The Sixth International Conference on Case-Based Reasoning, Springer, August 2005.
- [Bar-Ilan, 2004] J. Bar-Ilan. An outsider’s view on ‘topic-oriented blogging’. In *Proceedings of the 13th International World Wide Web Conference*, 2004.
- [Berners-Lee et al., 2001] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 279, May 2001.
- [Finin et al., 1994] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In N. Adam, B. Bhargava, and Y. Yesha, editors, *Proceedings of the 3rd International Conference on Information and Knowledge Management (CIKM’94)*, pages 456–463, Gaithersburg, MD, USA, 1994. ACM Press.
- [FIPA, 2000] FIPA. Foundation for intelligent physical agents, 2000.
- [Fodor et al., 2002] O. Fodor, M. Dell’Erba, F. Ricci, and H. Werthner. Harmonise: a solution for data interoperability. In *Proceedings of the 2nd IFIP Conf.on E-Commerce, E-Business & E-Government*, Lisbon, Portugal, October 2002.
- [Kuhn, 1955] H.W. Kuhn. The hungarian method for the assignment problem. In *Naval Research Logistic Quarterly*, pages 83–97, 1955.
- [Lewis, 1998] D. D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *ECML ’98: Proceedings of the 10th European Conference on Machine Learning*, pages 4–15, London, UK, 1998. Springer-Verlag.
- [McGuinness et al., 2002] D. L. McGuinness, R. Fikes, J. Hendler, and L. A. Stein. Daml+oil: An ontology language for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, 2002.
- [Nardi et al., 2004] B. A. Nardi, D. J. Schiano, and M. Gumbrecht. Blogging as social activity, or, would you let 900 million people read your diary? In *CSCW ’04: Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 222–231. ACM Press, 2004.
- [Neches et al., 1991] R. Neches, R. Fikes, T. Finin, T. Gruber, R. Patil, T. Senator, and W. R. Swartout. Enabling technology for knowledge sharing. *AI Mag.*, 12(3):36–56, 1991.
- [Page et al., 1998] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.
- [Rahm and Bernstein, 2001] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.
- [Reed et al., 2002] C. Reed, T. J. Norman, and N. R. Jennings. Negotiating the semantics of agent communication languages. *Computational Intelligence*, 18 (2):229–252, 2002.
- [Richardson et al., 2003] M. Richardson, R. Agrawal, and P. Domingos. Trust management for the semantic web. In *International Semantic Web Conference*, pages 351–368, 2003.
- [Schiano et al., 2004] D. J. Schiano, B. A. Nardi, M. Gumbrecht, and L. Swartz. Blogging by the rest of us. In *Extended abstracts of the 2004 conference on Human factors and computing systems*, pages 1143–1146. ACM Press, 2004.
- [Steels and McIntyre, 1999] L. Steels and A. McIntyre. Spatially Distributed Naming Games. *Advances in Complex Systems*, 1(4):301–323, January 1999.

Relational Graphical Models for Collaborative Filtering and Recommendation of Computational Workflow Components

William H. Hsu

Laboratory for Knowledge Discovery in Databases, Kansas State University
234 Nichols Hall, Manhattan, KS 66506

bhsu@cis.ksu.edu

<http://www.kddresearch.org>

Abstract

This paper describes preliminary work on using graphical models to represent relational data in computational science portals such as *myGrid*. The objective is to provide a integrative *collaborative filtering* (CF) capability to users of data, metadata, source code, and experimental documentation in some domain of interest. Recent systems such as *ResearchIndex / CiteSeer* provide collaborative recommendation through citation indexing, and systems such as *SourceForge* and the Open Bioinformatics project provide similar tools such as content-based indexing of software. Our current research aims at learning *probabilistic relational models* (PRMs) from data in order to support intelligent retrieval of data, source code, and experimental records. We present a system design and a précis of a test bed under development that applies PRM structure learning and inference to CF in repositories of bioinformatics data and software.

Keywords: relational graphical models, collaborative information retrieval, explanation of recommendation, user profiling

1 INTRODUCTION

Collaborative filtering is the problem of analyzing the content of an information retrieval system and actions of its users, to predict additional topics or products a new user may find useful. Developing this capability poses several challenges to machine learning and reasoning under uncertainty. The research described in this summary addresses the problem of formulating tractable and efficient problem specifications for probabilistic learning and inference in this framework. It describes an approach that combines learning and inference algorithms for relational models of semi-structured data into a domain-specific collaborative filtering system. Recent systems such as *ResearchIndex / CiteSeer* have succeeded in providing some specialized but comprehensive indices of full documents. The collection of user data from such digital libraries provides a test bed for the underlying IR

technology, including learning and inference systems. The authors are therefore developing two research indices in the areas of bioinformatics (specifically, functional genomics) and software engineering (digital libraries of source codes for computational biology), to experiment with machine learning and probabilistic reasoning software recently published by the authors and a collaborative filtering system currently under development.

The overall goal of this research program is to develop new computational techniques for discovering *relational and constraint models* for domain-specific collaborative filtering from scientific data and source code repositories, as well as use cases for software and data sets retrieved from them. The focus of this project is on statistical evaluation and automatic tuning of algorithms for learning graphical models of uncertain domains from such data. These include probabilistic representations, such as *Bayesian networks* and *decision networks*, which have recently been applied to a wide variety of problems in intelligent information retrieval and filtering. The primary contribution of this research shall be the novel combination of algorithms for learning the structure of relational probabilistic models with existing techniques for constructing relational models of metadata about computational science experiments, data, and programs. The technical objectives center around statistical experiments to evaluate this approach on data from the domains of *gene expression modeling* and *indexing of bioinformatics repositories*.

1.1 Rationale

Recent systems such as *ResearchIndex / CiteSeer* [LGB99] have succeeded in providing cross-indexing and search features for specialized but comprehensive **citation** indices of full documents. The indexing technologies used by such systems, as well as the general-purpose algorithms such as *Google PageRank* [BP98] and *HITS* [K199], have several advantages: They use a *simple conceptual model* of document webs. They require little specialized knowledge to use, but organize and present hits in a way that allows a knowledgeable user to select relevant hits and build a collection of interrelated documents quickly. They are extremely popular,

encouraging users to submit sites to be archived and corrections to citations, annotations, links, and other content. Finally, some of their content can be automatically maintained.

Despite these benefits, systems such as *ResearchIndex* have limitations that hinder their direct application to IR from bioinformatics repositories:

- **Over-generality:** Citation indices and comprehensive web search engines are designed to retrieving all individual documents of interest, rather than collections of data sets, program source codes, models, and metadata that meet common thematic or functional specifications.
- **Over-selectivity:** Conversely, IR systems based on keyword or key phrase search may return fewer (or no) hits because they check titles, keywords, and tags rather than semi-structured content.
- **Lack of explanatory detail:** A typical user of an integrated collaborative filtering system has a specific experimental objective, whose requirements he or she may understand to varying degree depending upon his or her level of expertise. The system needs to be able to **explain relationships** among data, source codes, and models in the context of a bioinformatics experiment.

1.2 Objectives and Hypothesis

How can we achieve the appropriate balance of generality and selectivity? How can we represent inferred relationships among data entities and programs, and explain them to the user? Our thesis is:

Probabilistic representation, learning, and reasoning are appropriate tools for providing domain-specific collaborative filtering capability to users of a scientific computing repository, such as one containing bioinformatics data, metadata, experimental documentation, and source codes.

Toward this end, we are developing *DESCRIBER*, a research index for consolidated repositories of **computational genomics resources**, along with machine learning and probabilistic reasoning algorithms to refine its data models and implement collaborative filtering. The unifying goal of this research is to advance the automated extraction of **graphical models of use cases** for computational science resources, to serve a user base of researchers and developers who work with genome data and models. We present current work in progress and survey results from related research that suggest how this can be achieved through a novel combination of probabilistic representation, algorithms, and high-performance data mining not previously applied to collaborative filtering in bioinformatics. Our project shall also directly advance gene expression modeling and

intelligent, search-driven reuse in distributed software libraries.

2 CF IN COMPUTATIONAL SCIENCES

2.1 Collaborative Filtering Objectives

We seek to take existing ontologies and minimum information standards for computational genomics and create a refined and elaborated data model for decision support in retrieving data, metadata, and source codes to serve researchers. A typical collaborative filtering scenario using a domain-specific research index or portal is depicted in Figure 1. We now survey background material briefly to explain this scenario, then discuss the methodological basis of our research: development of learning and inference components that take records of use cases and queries (from web server logs and forms) and produce decision support models for the CF performance element.

As a motivating example of a computational genomics experiments, we use gene expression modeling from microarray data. DNA hybridization *microarrays*, also referred to as *gene chips*, are experimental tools in the life sciences that make it possible to model interrelationships among genes, which encode instructions for production of proteins including the *transcription factors* of other genes. Microarrays simultaneously measure the expression level of thousands of genes to provide a “snapshot” of protein production processes in the cell. Computational biologists use them in order to compare snapshots taken from organisms under a control condition and an alternative (e.g., *pathogenic*) condition. A microarray is typically a glass or plastic slide, upon which DNA molecules are attached at up to tens of thousands of fixed locations, or *spots*. Microarray data (and source code for programs that operate upon them) proliferate rapidly due to recent availability of chip makers and scanners.

A major challenge in bioinformatics is to discover gene/protein interactions and key features of a cellular system by analyzing these snapshots. Our recent projects in computational genomics focus on the problem of automatically extracting gene regulatory dependencies from microarray data, with the ultimate goal of building simulation models of an organism under external conditions such as temperature, cell cycle timing (in the yeast cell), photoperiod (in plants), etc. Genomes of model organisms, such as *S. cerevisiae* (yeast), *A. thaliana* (mouse ear cress or *weed*), *O. sativa* (rice), *C. elegans* (nematode worm), and *D. melanogaster* (fruit fly), have been fully sequenced. These have also been annotated with the *promoter* regions that contain binding sites of *transcription factors* that regulate gene

expression. Public repositories of microarray data such as the *Saccaromyces* Genome Database (SGD) for yeast have been used to develop a comprehensive catalog of genes that meet analytical criteria for certain characteristics of interest, such as *cell cycle regulation* in yeast. We are using SGD data and a synthesis of existing and new algorithms for learning Bayesian networks from data to build robust models of regulatory relationships among genes from this catalog. Most data resources we plan to use in developing *DESCRIBER* are in the public domain, while some are part of collaborative work with the UK *myGrid* project [ZGSB04].

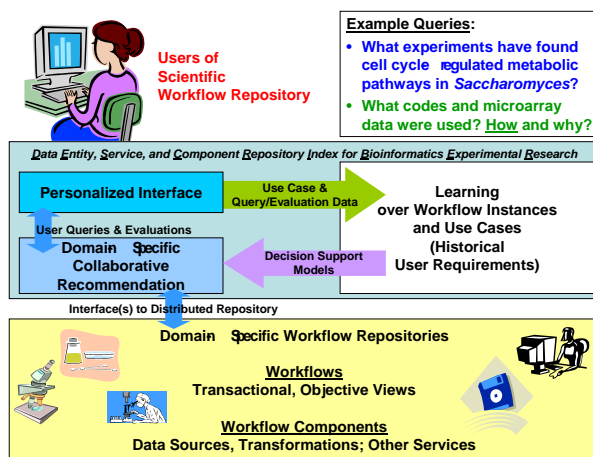


Figure 1. Design overview of *DESCRIBER*.

The next two figures depict our design for *DESCRIBER*. Figure 2 is the block diagram for the overall system, while Figure 3 elaborates Module 1 as shown in the lower left hand corner of Figure 2. Our current and continuing research focuses on algorithms that perform the learning, validation, and change of representation (inductive bias) denoted by Modules 2 and 4. We choose probabilistic relational models as a representation because they can express constraints (cf. Figure 1) and capture uncertainty about relations and entities. We hypothesize that this will provide more flexible generalization over use cases. We have recently developed a system for Bayesian network structure learning that improves upon the *K2* [CH92] and *Sparse Candidate* [FLNP00] algorithms by using combinatorial optimization (by a genetic algorithm) to find good topological orderings of variables. Similar optimization wrappers have been used to adapt problem representation in supervised inductive learning.

Other relevant work includes *BioIR*, a digital library for bioinformatics and medical informatics whose content is much broader than that of this test bed for genome analysis. *BioIR* emphasizes phrase browsing and cross-indexing of text and data repositories rather than experimental metadata and source codes. Other systems such as *CANIS*, *SPIDER*, and *OBIWAN* also address intelligent search and IR from bioinformatics digital

libraries, emphasizing categorization of text documents. We view the technologies in these systems as complementary and orthogonal to our work because of this chief difference.

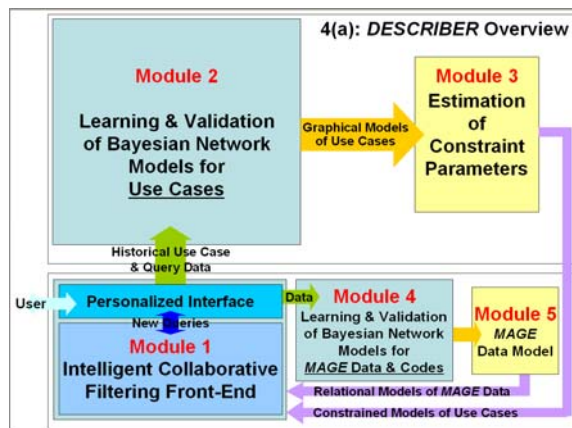


Figure 2. *DESCRIBER* system.

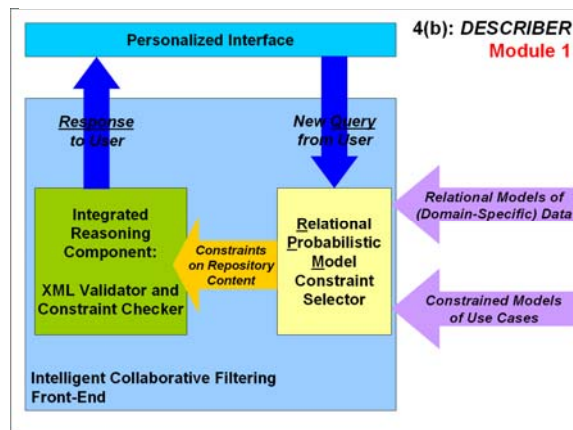


Figure 3. Collaborative filtering component.

3 NEW RGM MODELS

3.1 Managing Decisions under Uncertainty

Decision-theoretic intelligent agents must function under uncertainty and be able to reason and learn about objects and relations in the context of action and utility. This section presents a new relational graphical model (RGM), analogous to the probabilistic relational model (PRM), for representation of decisions under uncertainty. It first analyzes some basic properties of the representation and gives an adaptation of several decision network inference algorithms to these relational decision networks. It then describes some early experimentation with algorithms learning link structure in PRMs, discussing how these can be adapted to learning in decision networks. Finally, it considers the problem of representing dynamic relations in decision networks and sketches an extension of the dynamic PRM representation to include choice and utility.

Uncertainty is a common feature of decision problems for which the decision network or influence diagram is currently one of the most widely-used graphical models. Decision networks represent the state of the world as a set of variables, and model probabilistic dependencies, action, and utility. Though they provide a synthesis of probability and utility theory, decision networks are still unable to compactly represent many real-world domains, a limitation shared by other propositional graphical models such as flat Bayesian networks and dynamic Bayesian networks. Decision domains can contain multiple objects and classes of objects, as well as multiple kinds of relations among them.

Meanwhile, objects, relations, choices, and valuations can change over time. Capturing such a domain in a decision network would require not only an exhaustive representing of all possible objects and relations among them, but also a combinatorially fast-growing space of choices and valuations. This raises two problems. The first one is that the inference using such a dynamic decision network would likely exhibit near-pathological complexity, making the computational cost prohibitive. The second is that reducing the rich structure of domains such as workflow management to very large, “flat” decision networks would make it much more difficult for human beings to comprehend. This paper addresses these two problems by introducing an extension of decision networks that captures the relational structure of some decision domains, and by adapting methods for efficient inference in this representation.

3.2 Extending Decision Networks

The representation we introduce in this paper extends PRMs to decision problems in the same way that the decision networks extend Bayesian networks. We therefore call it the *relational decision network* or RDN. We develop two inference procedures for RDNs: the first based upon the traditional variable elimination algorithm developed by Shenoy and Cowell, the second a more efficient one based upon an adaptive importance sampling-based algorithm.

3.2.1 Probabilistic Relational Models

First-order formalisms that can represent objects and relations, as opposed to just variables have a long history in AI. Recently, significant progress has been made in combining them with a principled treatment of uncertainty. In particular, probabilistic relational models, or PRMs, are an extension of Bayesian networks that allows reasoning with classes, objects, and relations.

Probabilistic relational models (PRMs) [GFKT02, SDW03] extend the flat (propositional) representation of the variables and the conditional dependencies among them to an object-relational representation. Before proceeding to discussion the decision network analogues of PRMs, we briefly review the PRM family and the relevant components of a PRM specification. As an

example extending the DEC-Asia decision network above, the Patient schema might be used in to represent partial or total patient records, with classes corresponding to information about a patient's pulmonary medical history, smoking history, travel itinerary, and groups of people contacted. The propositional attributes of the medical history include the patient's age, previous contagious pulmonary diseases contracted, and currently extant diseases; the relational attributes might include the patient's membership in a list of quarantine subjects and *links* between patients denoting specific exposure incidents and contexts. Note that some of these are static and some, such as clusters of at-risk destinations and groups of people, may be dynamic relational attributes.

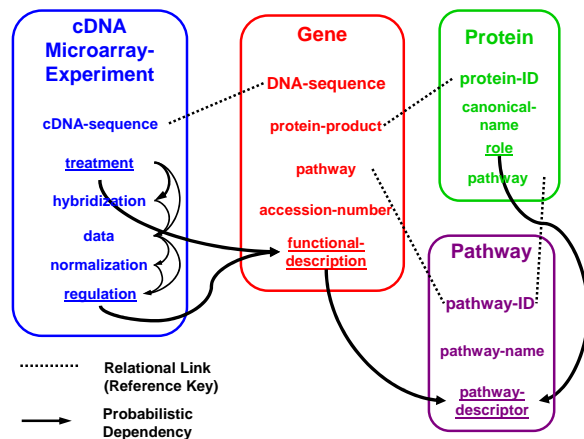


Figure 4. PRM for the *DESCRIBER* domain.

As a further example, Figure 4 depicts a PRM for the domain of computational genomics, particular gene expression modeling from DNA hybridization microarrays. Slot chains can be traced using the reference keys (dotted lines). This PRM contains tables for individual microarrays or gene chips (admitting aggregation of chip objects into classes), putative gene function (where known or hypothesized), putative pathway membership (where known or hypothesized), and protein production (a target aspect of discovered function).

This allows a PRM to be flattened into a large Bayesian network containing ground (propositional) chance nodes, with one variable for every attribute of every object in the relational skeleton of Π and belief functions (usually deterministic) for the aggregation operations. The latter are open-ended in form.

As Getoor *et al.* [GFKT02] and Sanghai *et al.* [SDW03] note, the most general case currently amenable to learning is where an object skeleton is provided and structure and parameter learning problems must be solved in order to specify a distribution over relational attributes. In the epidemiology domain, a PRM might specify a distribution over possible transmission vectors of an infected person (the itinerary, locale of contamination, and set of persons contacted).

3.2.2 Relational Decision Networks

Decision networks are extendible to relational representations using a simple and straightforward synthesis of decision network and PRM specifications.

Thus the relational attributes can include distinguished member *action identifiers* and *outcome identifiers* specifying a representation for equivalence classes of decisions and outcomes. Note that the range of actions may be continuous (e.g., in intelligent control or continuous decision problems) and the range of *utilities* may also be continuous.

When the decision network's object skeleton is not known (i.e., the set of decisions and outcomes is not fully pre-specified), the RDN includes boolean *existence variables* for propositional attributes of the relational tables, and boolean *reference slot variables* for relational attributes.

Inference algorithms that can be used with RDNs include two based on stochastic sampling: Likelihood Weighting and Adaptive Importance Sampling (AIS). For brevity, we refer interested readers to Cheng and Druzdzal [CD00] for detailed descriptions of these algorithms.

A desired joint probability distribution function $P(X)$ can be computed using the chain rule for Bayesian networks, given above. The most probable explanation (MPE) is a truth assignment, or more generally, value assignment, to a query $Q = X \setminus E$ with maximal posterior probability given evidence e . Finding the MPE directly using enumeration requires iteration over exponentially many explanations. Instead, a family of exact inference algorithms known as *clique-tree* propagation (also called *join tree* or *junction tree* propagation) is typically used in probabilistic reasoning applications. Although exact inference is important in that it provides the only completely accurate baseline for the fitness function f , the problem for general BNs is $\#P$ -complete (thus, deciding whether a particular truth instantiation is the MPE is NP-complete).

Approximate inference refers to approximation of the posterior probabilities given evidence. One stochastic approximation method called importance sampling [CD00] estimates the evidence marginal by sampling query node instantiations.

4 CONTINUING WORK

Our current research focuses on structure learning of relational models by adapting traditional score-based search algorithms for flat graphical models [Pe03] and constrain-based structure search over hierarchical models.

Entity and reference slot uncertainty present new challenges to PRM structure learning. Three of the questions that we are looking into are:

1. *How much relational data is needed?* How can we estimate the sample complexity of PRMs under specified assumptions about entity existence and reference slot distributions?

2. *What constraint-based approaches can be used?* Learning reference slot and entity structure in PRMs presents a task beyond flat structure learning.
3. *Can this cut down on the amount of data to learn the low-level model (versus the flat version)?* How can we establish and test sufficient conditions for conditional independence, and context-specific independence, in PRMs?

5 References

- [BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117.
- [CD00] J. Cheng and M. J. Druzdzal. AIS-BN: An adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks. *Journal of Artificial Intelligence Research*, 13:155-188, 2000.
- [CH92] G. F. Cooper and E. Herskovits. A Bayesian Method for the Induction of Probabilistic Networks from Data. *Machine Learning*, 9(4):309-347, 1992.
- [FLNP00] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (RECOMB 2000)*, ACM-SIGACT, April 2000.
- [GFKT02] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning Probabilistic Models of Link Structure. *Journal of Machine Learning Research*, 2002.
- [KI99] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604-632.
- [LGB99] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67-71.
- [Pe03] B. B. Perry. *A Genetic Algorithm for Learning Bayesian Network Adjacency Matrices from Data*. M.S. thesis, Department of Computing and Information Sciences, Kansas State University, 2003.
- [SDW03] S. Sanghai, D. Weld, and P. Domingos. Dynamic Probabilistic Relational Models. In *Proceedings of IJCAI-2003*.
- [ZGSB04] J. Zhao, C. Goble, R. Stevens, and S. Bechhofer. Semantically Linking and Browsing Provenance Logs for E-science. In *LNCS 3226, Semantics of a Networked World: Semantics for Grid Databases*. Berlin: Springer-Verlag.

Negotiation for task allocation among agents in case-base recommender systems: a swarm-intelligence approach

Fabiana Lorenzi^{1,2}, Daniela Scherer dos Santos¹ and Ana L. C. Bazzan¹

¹Instituto de Informática, UFRGS

Caixa Postal 15064 91.501-970 Porto Alegre, RS, Brasil

²Universidade Luterana do Brasil

Av. Farroupilha, 8001 Canoas, RS, Brasil

lorenzi@ulbra.tche.br, dsradtk@inf.ufrgs.br

Abstract

Planning a trip is a complex task to the travel agencies because it involves dynamic information and the travel agent's experience. This paper presents the use of swarm intelligence in the task allocation among cooperative agents applied to a case-based recommender system to help in the process of planning a trip.

1 Introduction

In the tourism branch, travel agencies have a hard task when planning a trip because it depends on several information that change over time. Besides, this information is distributed in the internet (in sites of flight companies, hotels, city's attractions). Another difficulty is that, many times, the trip destination is not fixed. Finally, it is hard to plan it according to the user's preferences without all the information needed.

Thus, a tool is necessary to retain the knowledge about the most popular destinations (e.g. within a season) and the travel agency experience. To solve those questions we propose an approach that combine case-based recommender systems and agents. The overall system has two different layers (with different goals): to search the case base and to search new information in the internet. This paper focusses on the former, proposing a metaphor from swarm intelligence to help the negotiation process among agents.

A Multi-Agent System (MAS) needs to manage the problem dynamic, such as variation in the number of agents, changes in environment and in the system's goals. These organizational issues depend on the system's goals, the perceived environment, and the relationships among agent's activities, as well as their interactions. In MAS, there have been many propositions to tackle these issues, whose discussion is outside the scope of this paper. However, a swarm-based approach to adapt organizations in MAS was proposed in [7] and is further discussed here in Section 2. This approach is useful because it deals with dynamic organizations, the same kind of problem posed by the tourism scenario just described.

The next section provides a brief introduction to case-based recommender systems, swarm intelligence, and negotiation among agents. Section 3 discusses the tourism domain and the problems found in the travel agencies. Section 4 shows

the proposed approach to the described scenario, while section 5 presents and discusses the results of the experiments performed to validate the approach. Finally section 6 shows the conclusions and the future work.

2 Related Work

2.1 Case-Based Recommender Systems

Case-Based Reasoning (CBR) is a problem solving methodology that deals with a new problem by first retrieving a past, already solved similar case, and then reusing that case for solving the new problem [1]. In a CBR recommender system (CBR-RS) a set of suggested products is retrieved from the case base by searching for cases similar to a case described by the user [4].

In the simplest application of CBR to recommend, the user is supposed to look for some product to purchase. He/she inputs some requirements about the product and the system searches the case base for similar products (by means of a similarity metric) that match the user requirements. A set of cases is retrieved from the case base and these cases can be recommender to the user. If the user is not satisfied with the recommendation he/she can modify the requirements, i.e. build another query, and a new cycle of the recommendation process is started.

The case retrieval is typically the main step of the CBR cycle and the majority of CBR recommender systems can be described as sophisticated retrieval engines. For example, in the Order-Based Retrieval [3] the system uses special operators to retrieve a partially-ordered set, or in the Compromise-Driven Retrieval [10] the system retrieves similar cases from the case base but also groups the cases, putting together those offering to the user the same compromise, and presents to the user just a representative case for each group.

2.2 Negotiation among agents and swarm intelligence

In a MAS it is necessary that agents coordinate their actions because they do not have a global view, and so the goals and knowledge are local, making it difficult to cooperate. Besides, the system should be able to deal with global constraints, which are not perceived by the individual agents' local view, and with inter-agents dependencies.

In a more general way, coordination increases the MAS performance. There are many ways to coordinate agents in a MAS, classified by Nwana et al. [11] in four categories: organizational structuring, contracting, multi-agent planning and negotiation.

Negotiation is a process where two or more parties make a joint decision [15]. The negotiation can be classified as competitive or cooperative. We are interested here in the cooperative negotiation that can be viewed as a distributed search process where the agents try to reach the maximum global utility.

The negotiation process starts when the agent realizes that it is not the best option to perform the task. This can happen for many reasons: the agent can be overloaded, the agent does not have capacity to perform the task or maybe another agent can perform the task with more quality. When the agent has to perform the task it validates the task's utility (cost and time) and decides if it is necessary to start the negotiation process.

One way to active cooperation without explicit communication is to use swarm intelligence. The use of the social insect metaphor to solve computer problems such as combinatorial optimization, communications networks or robotics is increasing [2].

Social insects living in colonies, e.g. ants, termites, bees, and wasps [14] distinguish themselves by their organization skill without any centralized control [5; 9]. Organization emerges from interactions among individuals, between the individuals and the environment, and from behaviors of the individuals themselves [2]. In [7] the authors present a swarm-based approach applied to task allocation in dynamic environments where the agents adapt to changes in the organization, just as social insects do.

Another example of self-organization is found in the bees. Among these, the colony selects the best nectar source available through simple behavioral rules. The process of dispatching bees into the surrounding countryside to gather the colony's food is called foraging. Bees travel up to 10km from the hive to collect nectar. They return with nectar and information about the nectar source [5].

The bee has three behavior options in the foraging process [5]:

1. to share the nectar source information by dancing, a behavior in which a bee communicates to other bees the direction, distance, and desirability of the food source, trying to recruit new bees to that food source.
2. to continue foraging without recruiting other bees.
3. to abandon the food source and go to the area inside the hive called the dance floor to observe dancing bees and select its next food source.

In [13] the authors showed that the foraging decision of each bee is based on very limited information of its own particular visited source. This simple behavior allows the colony to select the best quality source of nectar. Colonies select the best quality source through the rate of dancing and abandonment based upon nectar source quality. Camazine and Sneyd [6] developed a mathematical model which demonstrates how the properties of the system emerge automatically from the dynamic interactions among the constituent components.

3 Scenario

Tourism is the second most important social economic activity in Brazil. It involves many actors: the traveler (customer), the travel agencies, the clerk, flight companies, hotels, the tour operator and others. This reality makes the sector grows every year.

Normally, the traveler goes to a travel agency to get help to plan her/his trip. Such a planning is not an easy task because it involves dates and times of flights, booking hotels, timetable from attractions and further information.

Most of the travel agencies do not have information systems to help the customer to plan a trip. Few systems only search for packages in a catalogue pre-determined by the tour operators (e.g. Panrotas - www.panrotas.com.br). The existing systems for travel planning have unsolved problems such as:

- Data is not integrated: the flight companies, hotels and tourism operators do not use a unique system. For example, American Airlines works with SABRE (www.sabreairlinesolutions.com) and VARIG (Viacao Aerea Riograndense) works with AMADEUS (www.amadeusbrasil.com.br). The clerk must have both systems to search for flights to the customer;
- Sometimes the clerk does not have information about the customer desired destination(s). It is difficult to plan a travel if he/she does not know about all destination details, such as, what time the museum is open, how can the traveler go from one place to another one;
- The tourism operators give the travel agency a file of pre-defined tourist destinations (packages/catalogues). These packages include: flight tickets, hotel, city tour and transfers in-out. Some of them include more than one destination but they are predefined and it is not possible to change them or to adapt them to the customers preferences. In these cases, the clerk has to create a new package from scratch (the so-called forfeit) and he/she needs to search for each piece information to complete the whole travel.

These problems make planning a trip a complex problem solving activity for the clerk of the travel agency. Recommender systems seem to be a good solution for them. The two most successful recommender system technologies, which have been recently applied to travel applications, trying to solve some of these problems, are supplied by Triplehop (www.triplehop.com) and VacationCoach Inc (www.vacationcoachinc.com). Both recommenders try to cope with the added complexity simulating the interactivity observed in traditional counseling sessions with the clerk of the travel agency.

Taking into account that the information needed to plan a trip is distributed into the internet, using agents to search this information is an appropriate technique. Next section shows our proposal of a case-base recommender system using agents in two layers: to recommend new travels using past cases and to search new information in the web (when the cases are not updated).

4 Approach

Many researches are generating new solutions to show how e-commerce could be used for travel and tourism, trying to reproduce the traditional travel agents advices. In [12], the authors describe an intelligent recommender system developed with Case-Based Reasoning approach that helps a traveler to select a tourist destination, building a set of products and composing a plan for the travel. Dietorecs [8] is also a recommender system that incorporates a human decision model that stresses individual differences in decision styles.

We propose the development of a case-based recommender system to the forfeit's planning task using agents to recommend and also to search updated information into the web.

The first layer of agents is responsible for the recommendation. In this process the metaphor of the honey bees is used to select the cases. The Camazine and Sneyd's mathematical model [6] was adapted and used in the negotiation process.

Each agent represents a bee with the following features: probability of abandoning the nectar source (P_X^i); probability of recruiting more bees by dancing (P_D^i) and probability of keep following the source (P_F^i).

There is a case base with past travels and the agents search in this case base for possible travel to recommend according to the users' preferences or the user's profile. Each case is viewed as a nectar source for the agents bees. Given a new user query, the metaphor calls for honey bees leaving the nest looking for nectar sources.

According to Camazine and Sneyd's model, each bee's choice is calculated by using probabilities. The most important one is the abandon where the bee checks the quality of the nectar source and decides whether or not to continue in this source. In our system the original model was modified and the bee can forage for several different sources (not only two as in the original model).

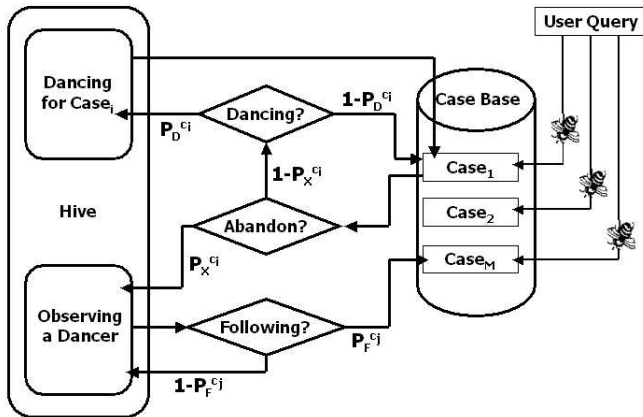


Figure 1: The metaphor – adapted from Camazine and Sneyd

Figure 1 shows the adapted model with the hive having two compartments (dancing and observing), the case base (representing the available nectar sources), and the diamonds representing the bee's decisions possibilities.

senting the available nectar sources), and the diamonds representing the bee's decisions possibilities.

In a traditional case-based recommender system, the new query is compared with all cases in the case base and the most similar is shown to the user. In this approach, the similarity is used to help the bee to make its decision to continue in the nectar source or not. The bee visits the case base, comparing the query with each case and it calculates the (P_X^i). The cases with the smallest distance to the new query are the most similar, i.e. they represent the best nectar sources. c_i is the case the bee is visiting and c_j is the case visited by the dancer bee, which the observing bee will decide to follow it or not.

The black diamonds shown in figure 1 represent the bee's possibilities. It can decide to continue foraging at the case, or either visit the compartment *Dancing for Case* to perform recruitment's dance before returning to its case or continue to forage at the case without recruiting other bees. Another possibility is to abandon the case and then visit the compartment *Observing a dancer* (that has bees dancing for different cases). Here the bee decides which dancer bee it will follow. It means that it will visit the bee dancer case and the process will restart. In each one of these possibilities the probabilities are calculated.

As shown in algorithm 1 when the user informs his/her preferences the negotiation process is called. Several parameters are initialized in the beginning of the algorithm such as the total number of bees and the number of bees that will be distributed in case base (K).

The advantage of using this metaphor in the negotiation process is that the bees always return something to recommend (in this first layer). Despite the controversy that sometimes is better not recommend instead of recommend wrong products, this is not true in the tourism context, especially in the frequent case where the client does not know exactly where to go/what to do. Our approach always return some recommendation, which is specific for the current season, given the dynamic nature of the approach (bees adapting to the environment pressure which is here the travel demands of the current season). This can be seen as a way of dynamically clustering the cases.

Searching first in the case base makes the recommendation more efficient and reduces the time spent searching for information. On the other hand, the cases can be old and not valuable anymore to the recommendation.

In order to solve this problem we propose a component that controls the validation of each case. For example, a trip done last year maybe has different timetables and it is not worth it anymore. So, this *useful component* helps the agents to validate if the case is still useful. If not, the case is removed from the case base and the second layer of agents is called. These agents have to search updated information in the web and plan the new trip to recommend.

The second layer is responsible for search updated information into the web. Here there are agents that need to update information and return it to the recommendation's agents.

5 Results

To validate the negotiation algorithm we have done some experiments with different users' queries and also different parameters such as number of bees and number of iterations. This section shows two of these experiments and their results.

In the first experiment, we used 2500 agents (which is considered a high number) and the system was allowed to run until the best case emerges. Figure 2 shows the results of this first experiment. We plot only the most visited cases during the simulation in order to avoid a dense graphic. Among these, only three have high similarity with the user query. The figure shows that, in the beginning, the agents consider various cases because they have little information about them (as it is the case with real nectar sources which are visited randomly).

By time $t=20$ the best case emerges as *case 04* and this is in fact the best package the user can have (in the case base, *case 04* is the one with the highest similarity to the user's query).

Algorithm 1 Recommendation's algorithm

{N is the number of bees}
 {K is the number of bees that will be distributed in the case base}
 {M is the number of cases (sources)}
 {CB is the case base - $CB = c_1, c_2, \dots, c_M$ }
 {MAX is number of iterations}
 {q is the user's query}

Procedure Negotiation (q, CB)

```

for each  $c_i \in CB$  do
   $Bdc \leftarrow (K * N) / M$  {Bees' distribution in each case.
  The number of bees that will visit the case}
end for
 $Bdo \leftarrow (N - K)$  {Bees' distribution in the observing
dancer's compartment}
repeat
  for  $b = 1, \dots, N$  do
    {only the bees visiting the case base}
     $P_X^{c_i} \leftarrow \text{distance}(q, c_i)$ 
    if ( $P_X^{c_i}$  not met) then
       $P_D^{c_i} \leftarrow 1 - P_X^{c_i}$ 
      if ( $P_D^{c_i}$  met) then
        visit the compartment Dancing for Case
      else
        keep foraging
      end if
    else
      visit the compartment Observing a dancer
      repeat
        choose a random dancer bee
         $P_F^{c_j} \leftarrow D_{c_j} / \sum_{x=1}^N D_x$  { $D_{c_j}$  is the number of
        bees dancing for  $c_j$ }
        if ( $P_F^{c_j}$  met) then
           $c_i \leftarrow c_j$  {Now the bee will forage in  $c_j$ }
        end if
      until ( $c_i = c_j$ )
    end if
  end for
until (cycles' number = MAX)
return Mostvisitedcase {The case with the highest number
of bees in the end of the iterations}
  
```

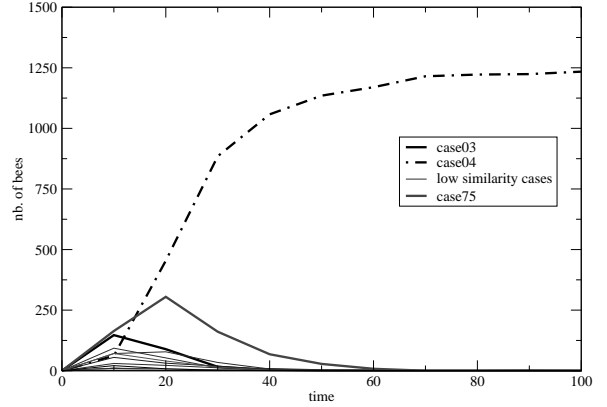


Figure 2: Number of agents (bees) visiting each case along time: experiment with 2500 bees

In the next experiment shown in figure 3 we used the same user query but we reduced the number of agents to 500, allowing the simulation to run up to 500 iterations. Here, an interesting situation arises: agents started to visit *case 69* because it had the highest similarity *in the beginning*. However, as soon as *case 04* was founded as a better case (around $t=150$), the agents start abandoning visiting *case 69* and prefer to visit *case 04*. This shows that the approach is effective.

Moreover, the approach is also robust to dynamic changes in the queries: during the iterations, it is possible to notice that the bees' behavior is automatically clustered around cases according to the user's preferences. For example, in the tourism scenario, users' preferences usually change with seasons of the year: during summer beaches are favorite spots, whereas in winter people normally look for skiing or tropical resorts. Once the user's query changes, so does the clustering of bees.

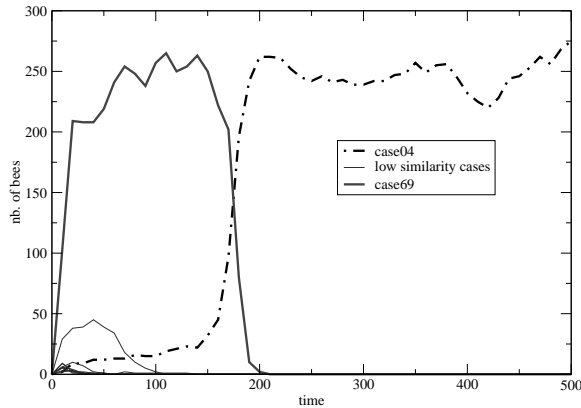


Figure 3: Number of agents (bees) visiting each case along time: experiment with 500 bees

6 Conclusions and Future Work

This paper presented the approach of agents (inspired by social insects) applied to a case-based recommender system in the tourism domain. This use is justified by the success agents (and also social insect) have in dynamic changing environments.

The planning travel task needs dynamic information that is not always available to the travel agent. The use of agents combined to case-base recommender systems helped the travel agent in this task.

Our experiment' results have shown that using this metaphor the system always return some travel to recommend to the user, avoiding the user's disappointment with the recommender system.

In the future we intend to improve the case representation, saving the previous queries as cases. Depending on the similarity of the new query the agents can start not completely randomly but using information gathered regarding the previous query.

As another future work we want to dedicate more to the useful component to validate the cases and the development of the agents that search the update information in the internet.

References

[1] A. Aamodt and E. Plaza. Case-based reasoning: foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, 1994.

[2] E. Bonabeau, G. Thraulaz, and M. Dorigo. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford Univ Press, 1999.

[3] D. Bridge and A. Ferguson. Diverse product recommendations using an expressive language for case retrieval. In S. Craw and A. Preece, editors, *Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR*

2002, pages 43–57, Aberdeen, Scotland, 4 - 7 September 2002. Springer Verlag.

[4] R. Burke. Knowledge-based recommender systems. In J. E. Daily, A. Kent, and H. Lancour, editors, *Encyclopedia of Library and Information Science*, volume 69. Marcel Dekker, 2000.

[5] S. Camazine, J. D. Deneubourg, N. R. Franks, J. Sneyd, G. Thraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton, 2003.

[6] S. Camazine and J. Sneyd. A model of collective nectar source selection by honey bees: Self-organization through simple rules. *Journal of Theoretical Biology*, 149(4):547–571, April 1991.

[7] P. R. Ferreira Jr., D. Oliveira, and A. C. Bazzan. A swarm based approach to adapt the structural dimension of agents' organizations. *Journal of Brazilian Computer Society (JBACS) Special Issue on Agents Organizations*, 2005.

[8] D. R. Fesenmaier, F. Ricci, E. Schaumlechner, K. Wöber, and C. Zanella. DIETORECS: Travel advisory for multiple decision styles. In A. J. Frew, M. Hitz, and P. O'Connors, editors, *Information and Communication Technologies in Tourism 2003*, pages 232–241. Springer, 2003.

[9] D. Gordon. The organization of work in social insect colonies. *Nature*, 380:121–124, 1996.

[10] D. McSherry. Similarity and compromise. In A. Aamodt, D. Bridge, and K. Ashley, editors, *ICCB 2003, the 5th International Conference on Case-Based Reasoning*, pages 291–305, Trondheim, Norway, June 23–26 2003.

[11] H. S. Nwana, L. C. Lee, and N. R. Jennings. Coordination in software agent systems. *The British Telecom Technical Journal*, 14(4):79–88, 1996.

[12] F. Ricci and H. Werthner. Case-based querying for travel planning recommendation. *Information Technology and Tourism*, 4(3/4):215–226, 2002.

[13] D. Seeley, S. Camazine, and J. Sneyd. Collective decision-making in honey bees: How colonies choose nectar sources. *Behavioral Ecology and Sociobiology*, 28:277–290, 1991.

[14] E. O. Wilson. *Sociobiology: The New Synthesis*. Harvard Univ Press, 2000.

[15] X. Zhang, V. Lesser, and R. Podorozhny. Multi-dimensional, multistep negotiation for task allocation in a cooperative system. *Autonomous Agentes and Multi-Agent Systems*, 10:5–40, 2005.

Making Use of Fipa Multiagent Architecture to Develop Animated Pedagogical Agents Inside Intelligent Learning Environments

Joel Pinho Lucas; Beatriz Wilges; Ricardo Azambuja Silveira

Universidade Federal de Pelotas

Department of Informatics

(+55) 53-3275 7449

{joelpl; beaw; rsilv}@ufpel.edu.br

Abstract

Pedagogical Agents are fundamental entities in learning environments modeled using the MAIDE framework, which models distributed intelligent learning environments (DILE). This paper shows a special class of pedagogical agent: the Animated Pedagogical Agent (APA). We propose to use this type of agent in multi-agent systems for learning by means of the Foundation for Intelligent Physical Agents (FIPA) standards for agent development, which provides many functionalities in the communication among agents through the use of the FIPA-ACL communication language. Learning is likely to be more efficient in distance learning environments if an APA be used, because APA presents functionalities, such as human-like behavior, that may increase interaction between a student and the environment. Finally, this paper presents a simple environment, developed for the MAIDE project, that shows the viability of developing APA in multi-agent systems for learning.

1 Introduction

The development of computing instruments for learning has been researched since the fifty years. Firstly, the purpose was to have software based on the educational model that has the teacher as the main character of the learning and the student as the passive one. So that, the student had to understand the subjects supplied by the teacher and after that, answer questions related to these subjects.

However, according to Pianesso et. al. [2003], educational software, which are based in this model, have some critical shortcomings, because the student is managed by the software and then does not play an active role in the learning. In this model, the student is instigated to act only if he/she is encouraged. Thus, there is no concerning with the student's reasoning and learning. Therefore, the educational software does not encourage the student to learn.

The Distributed Artificial Intelligence (DAI) came out in the beginning of the seventies due to the convergence of Artificial Intelligence (AI) and Distributed Systems areas. The major goal of DAI is try to solve problems in a cooperative way. One of DAI's sub-areas is the Multi-Agent Systems (MAS). In these systems, the cooperation among agents is focused, the agents have to cooperate with each other to reach a common goal.

One of the major shortcomings troubles of traditional computer based learning systems is how to provide adaptive teaching, suitable to each student. As the student's performance is rarely consistent, and it is impossible to preview the entire set of student's behavior, recent advances in the field of Intelligent Learning Environments have proposed the use of agent's society based architectures. The principles of multi-agent systems have shown a very adequate potential in the development of teaching systems, due to the fact that the nature of teaching-learning problems is more easily solved in a cooperative way. Following this purpose, a variety of learning environments [Cheikes, 1995; Johnson et. al., 1997] uses this kind of architecture, implementing Pedagogical Agents as a specialized class of agents with the necessary abilities to deal with teaching strategies.

According to Ferber [1999], a MAS is a distributed computing application composed of a set of autonomous, heterogeneous, asynchronous and intelligent processes named "agents". They are able to cooperate with each other to solve complex problems that are superior than their individual abilities. Thus, a MAS tries to coordinate an intelligent behavior among a set of autonomous agents.

In this context, the Intelligent Distributed Learning Environments Modeling (MAIDE – in Portuguese) project extends the scope of Java Agent framework for Distance learning Environments (JADE) project [Silveira, 2002]. JADE proposes an infrastructure of project, development and implementation of Distributed Intelligent Learning Environments (DILE), based on the approach of multi-agent architecture towards distance education, for multiple domains.

Having the MAIDE project as background, in this work we describe the use of Animated Pedagogical Agents (APAs) inside MAS for learning, by means of an example of a MAS containing an APA, which will be described in section 5.

2 The MAIDE Project

The MAIDE project proposes a framework to project, develop and implement DILE, based on the approach of FIPA (Foundation for Intelligent Physical Agents) FIPA [2002] Multi-Agent Architecture.

The MAIDE architecture encompasses a very small family of agents composed of just three types of agents (see Figure 1): an agent responsible for the student interface, one responsible for the student model, and a set of agents responsible for tasks related to teaching tactics (Pedagogical Agents). Each agent may have its tasks specified according to its goal. All these agents are implemented according to FIPA specifications. The FIPA foundation developed a set of documents, including a communication language named "FIPA-ACL". FIPA-ACL will be discussed in Section 4.

In order to ensure the FIPA compliance, the FIPA-OS [2002] platform was used. The FIPA-OS framework provides several Java classes for FIPA compliant agents' construction, an agent runtime environment and a set of services for FIPA message interchange. The entire message exchange, through the FIPA-ACL language, among FIPA agents is managed by FIPA-OS.

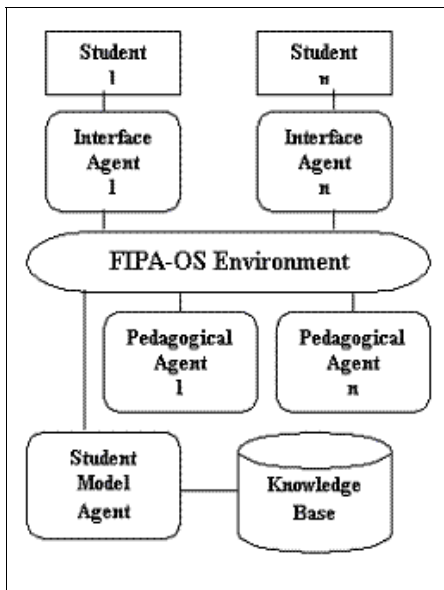


Figure 1 – MAIDE architecture

3 Pedagogical Agents

According to Bradshaw [1997], an agent is described as a software entity that works in a continuous and autonomous way in a particular environment, generally inhabited by other agents and able to interact in that environment in a flexible and intelligent way, not requiring human

intervention or guidance. Ideally, an agent who works continuously, for a long time, must be able to learn through experience and, if the environment inhabits other agents, the agent must be able to communicate and cooperate with them, sharing a common world.

The pedagogical agents, according to Giraffa et. al. [2001], are those based on the paradigm of agents developed for educational purpose, acting as virtual tutor, virtual student or virtual assistant of learning, in order to help students during the learning.

According to these definitions, the MAIDE project aims at developing pedagogical agents inside DILEs. In next sections it will be shown, through the implementation of a MAS containing an APA, the advantages and the viability of using APAs inside distance learning environments.

3.1 Animated Pedagogical Agents

According to Tebarth et. al. [2000], an APA can respond to the learner with a human-like combination of verbal communication and non-verbal gestures such as gaze, pointing, body stance, and head nods, and it is also able to convey emotions such as surprise, approval, or disappointment. Thus, the agent may adapt itself to the student needs, being able to interact and help any student.

Johnson et. al. [2000] and Jaques et. al. [2003], argue that APAs are the new model to learning environments. By means of complex tasks' demonstration using locomotion, gestures and expressing emotive answers, they are able to keep the student concerned in essential aspects of the tasks. They are responsible for attending the student in his/her interaction with the learning environment, being able to guide the student's actions. Therefore, it is argued that the learning may be more efficient.

The APA shown in this paper is represented by a character named Pedagogical and Affective Tutor (PAT) who was used by Jaques et. al. [2003].

4 Communication among FIPA agents

In order to show and exemplify the communication among software agents inside learning environments, the Section 5 presents a simple MAS composed of three agents developed according to FIPA standards. One of these agents is the APA, who is also used in the MAIDE project.

Due to the use of FIPA standards, it was possible to make system's agents communicate constantly, what contributed to the behaviour expressed by the APA, because due to this communication, the APA was able to be aware of the student's behaviour in the use of the system.

In order to make easier the development of agents suitable to FIPA standards, a platform for agents development was used. The FIPA-OS platform was the one chosen to be used in the MAIDE project.

The FIPA-OS platform was developed in Java programming language. It presents a set of classes and an environment containing all resources needed to perform communication among agents. In order to develop a FIPA agent, the class "FIPAOSAgent" has to be extended.

Considering that the MAIDE project had used FIPA-ACL communication language, the implementation of the example described in Section 5 also used FIPA-ACL.

5 Example of an APA being part of a MAS

This system aims at helping primary school students to learn more efficiently the fundamental mathematical properties of multiplication (commutative, zero element and identity element) and addition (commutative and identity element). The system is composed of a calculator owning only the addition and multiplication arithmetic operations. For each student's action on the calculator, the APA displays some message for him/her. This message aids and also encourages the student to learn the mathematical properties. Thus, this agent accomplishes an interface between the student and the calculator.

The system's life cycle starts when a screen containing the definition of the mathematical properties, as well as a tutorial explaining how apply them, is shown to the student. Secondly, the APA appears on the screen displaying a welcome message and after that, the calculator is shown on the screen. Finally, the student is able to accomplish the first operation. After the accomplishment of this operation, the APA informs, according to the operation accomplished previously, the student what mathematical properties can be applied and encourages the student to accomplish the next operation. After the accomplishment of the last (second) operation, the APA appears again on the screen to inform the student if any mathematical property was applied successfully or not. If the student was not successful on applying any mathematical property, he/she is informed of which were applied incorrectly, as well as the mistakes committed. If at least one property was applied successfully, the APA will just congratulate and inform the properties that were applied successfully.

The pictures above show the two elements of the system interface: the calculator on figure 2 and the APA on figure 3.

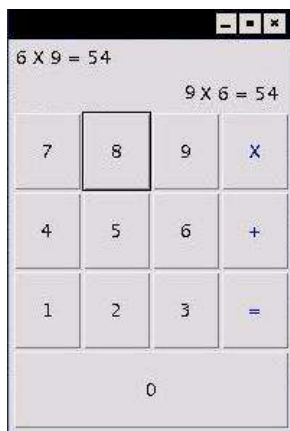


Figure 2 – The Calculator



With this operation you can demonstrate the commutat...

Figure 3 – The APA

5.1. The System's Agent Society

The agent society of the system is composed of three agents, all of them were implemented in Java programming language and in accordance to FIPA specifications. The first agent was named as "InitialAgent". InitialAgent is instantiated by the "Agent-Loader" owned by FIPA-OS.

This agent has the shortest life cycle of the System's Agent Society, because he/she has only the task of: instantiate other agents of the system and send a message to each of them, requesting to start performing their tasks.

The APA was named as "ApAgent". ApAgent is responsible for performing an interface between the student and the calculator. This agent is also responsible for the evaluation reasoning on the operations accomplished by the student and for the texts' formulation reasoning, which will be shown for the student. Both reasoning are based on rules.

The agent who holds the calculator's graphical interface was named as "CalcAgent". This agent plays the role of an informative of "ApAgent", because he/she is responsible for sending the operations accomplished by the student.

5.2. The Agent Communication Architecture

The cooperation model used on the system was the peer-to-peer message passing, making use of FIPA-ACL communication language. Considering that this system is composed of only three agents, the implementation of a coordination model and of a mediator agent would not be worthy.

Since a MAS has the viewpoint of solving problems in a distributed form, the expected method to be employed in this type of system is the "Divide and Conquer". The use of this method in a MAS targets to divide the global task of the system in sub-tasks distributed among agents that compose the system. In this method, the agents have to cooperate with each other to achieve their tasks. This cooperation is achieved through passing messages among agents.

In the system described in this section, the tasks' allocation among the system's agents is described bellow. Initial owns the following task:

- **Welcome:** it is achieved when the life cycle of the system starts. It is responsible for instantiating "CalcAgent" and "ApAgent".

CalcAgent owns the following task:

- SendResult: informs “ApAgent” the operation accomplished by the student.

ApAgent owns the following tasks:

- ResultReasoning: do the reasoning on the first operation accomplished by the student and then get the mathematical properties that may be applied. After that, it instantiates the task “InstructionReasoning”;
- EvaluationReasoning: do the reasoning to evaluate what mathematical properties were applied successfully. If non was, the mistakes committed are defined;
- InstructionReasoning: the role played by this task depends on the source of the message who achieved it. If the message has come from the task “ResultReasoning”, the APA displays, on its dialog box, an instruction stating what mathematical properties may be applied. However, if the message has come from the task “EvaluationReasoning”, the instruction states what properties were applied successfully, but if no one was, the instruction states the mistakes committed.

The picture below shows the protocol of messages’ flow among the system’s agents:

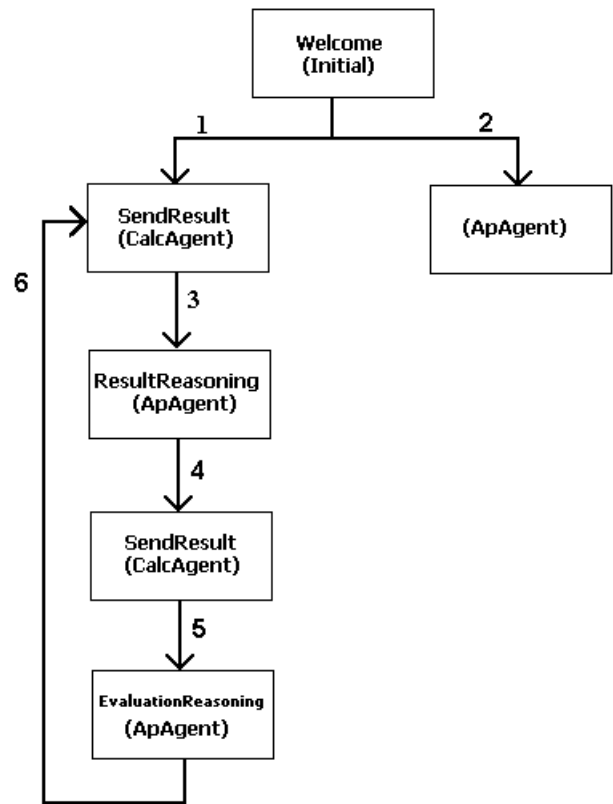


Figure 4 – The message’s flow Protocol

- Message 1: “InitialAgent” sends a request (through the task “Welcome”) to “CalcAgent” initialize the calculator;
- Message 2: “InitialAgent” sends a request (through the task “Welcome”) to “ApAgent” start displaying a welcome message for the student;
- Message 3: “CalcAgent” informs “ApAgent” (through the task “SendResult”) the first operation accomplished by the student;
- Message 4: “ApAgent” sends a request (through the task “ResultReasoning”) to “CalcAgent” unlock the calculator’s keyboard;
- Message 5: “CalcAgent” informs “ApAgent” (through the task “SendResult”) the second operation accomplished by the student;
- Message 6: “ApAgent” sends a request (through the task “EvaluationReasoning”) to “CalcAgent” unlock the calculator’s keyboard in order to restart the process.

6 Conclusion

In the example presented in Section 5, the APA has an active interaction with the student. For each action of the student on the calculator, the APA is capable of providing a

suitable message to the student. This interaction builds more reality to the environment, and as a consequence, the student feels that he/she is the active character of the learning. In addition, the student is likely to pay more attention on learning tasks supplied by the environment, because the human-like character of APA is similar to the student's reality.

This real-time interaction only was accomplished due to FIPA-ACL communication language provided by FIPA. FIPA-ACL enabled the communication between the calculator and the tasks of "ApAgent". Therefore, modeling MAS composed of APAs provides a powerful instrument for learning.

7 References

- [Bradshaw, 1997] J. M. Bradshaw. An introduction to software agents In: *BRADSHAW, J. M. Ed. Software Agents*. Massachusetts: MIT Press, 1997.
- [Cheikes, 1995] B. A. G. Cheikes.: Agent Based Architecture for Intelligent Tutoring Systems. In: *Proceedings of The Cikm Workshop On Intelligent Information Agents*, 1995.
- [Ferber, 1999] J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley. 1999.
- [FIPA, 2002] FIPA: *The foundation for Intelligent Physical Agents*. Specifications. Available by HTTP in <<http://www.fipa.org>> March, 2002.
- [FIPA-OS, 2002] FIPA-OS: *FIPA Open Source*. Available by HTTP in:<fipa-os.sourceforge.net> Dec, 2002.
- [Giraffa et. al., 2001] L. M. M. Giraffa, M.C. Móra, A. O. Zamberlam. Working on student models (really) based on mental states. In: *4th International Conference On Computational Intelligence And Multimedia Applications*. Los Alamitos, California. 2001.
- [Jaques et. al., 2003].Patricia A. Jaques, S. Pesty, Rosa M. Vicari. An Animated Pedagogical Agent that Interacts Affectively with the Student. In *AIED 2003, shaping the future of learning through intelligent technologies*. Sydney, Australia. 2003.
- [Johnson et. al., 2000] W. L. Johnson, J. Rickel, J. Lester. Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments. In: *The International Journal of Artificial Intelligence in Education*, 11. pp. 47-78. 2000.
- [Johnson et. al., 1997] W. L. Johnson, Erin Shaw. Using agents to overcome deficiencies in web-based courseware. In: *Proceedings of the World Conference On Artificial Intelligence In Education, Ai-Ed*, 8., 1997. Available by HTTP in: <www.isi.edu/isd/johnson.html> Mar, 2004.
- [Pianesso et. al., 2003] A. C. F. Pianesso, C. L. O. Castanho, J. Schreider.: Design of Intelligent Tutorial Systems (in Portuguese). Available by HTTP in <http://www.edit.info.ufsc.br:1998/aluno2000/artigo.doc> November, 2003.
- [Silveira , 2002] Ricardo A. Silveira, Rosa M. Vicari. Developing Distributed Intelligent Learning Environment with JADE - Java Agents for distance Education Framework. *Proceedings of ITS 2002 Lecture Notes in Computer Science*, Biarritz, v. 2363, p. 105-118, 2002.
- [Tebarth, 2000] H. Tebarth, Y. Mohamad, M. Pieper. Cognitive Training by Animated Pedagogical Agents (TAPA) Development of a Tele-Medical System for Memory Improvement in Children with Epilepsy. *6th ERCIM Workshop "User Interfaces for All"*. Florence, Italy. 2000.

Conversational CBR in Multi-Agent Recommendation

David McSherry

School of Computing and Information Engineering
University of Ulster, Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulster.ac.uk

Abstract

In case-based reasoning (CBR) approaches to product recommendation, a common cause of recommendation failure is that none of the recommended cases is acceptable to the user and the system has no means of accessing more acceptable cases that might be available elsewhere. To address this issue, we propose a multi-agent CBR approach in which a recommendation failure triggers a *referral* of the user's query to other recommender agents in search of additional cases to be considered in the recommendation process. The agent initiating referral of an unsuccessful query is responsible for elicitation of any additional user preferences needed to discriminate between cases suggested by other agents, delivery of the final recommendation, and explanation of the recommendation process.

1 Introduction

In a CBR recommender system, descriptions of available products are retrieved from a *case base* in response to a query representing the user's requirements. In approaches related to interactive/conversational CBR [e.g., Aha *et al.*, 2001; McSherry, 2001], a query is incrementally elicited in a recommendation *dialogue*, often with the aim of minimizing the number of questions the user is asked before an acceptable product is retrieved [e.g., Kohlmaier *et al.*, 2001; McSherry, 2003a].

Recent research has also highlighted the importance of recommender systems having the ability to explain their reasoning [Herlocker *et al.*, 2000; McSherry, 2004a; Shimazu, 2002] and recognize when the dialogue can be safely terminated without loss of solution quality [McSherry, 2003a]. Incremental nearest neighbor (iNN) is a conversational CBR approach that aims to address these issues in a mixed-initiative recommender system called *Top Case* [McSherry, 2004a]. It combines a *goal-driven* approach to selecting the most useful question at each stage of the recommendation dialogue with a simple mechanism

for ensuring that the dialogue is terminated only when it is certain that the recommendation will be the same no matter how the user chooses to extend her query [McSherry, 2003a]. Another important benefit is the ease with which the recommendation process can be explained. For example, Top Case can explain the relevance of any question the user is asked in terms of its ability to discriminate between competing cases. Recommendations based on incomplete queries can also be justified on the grounds that any unknown preferences of the user cannot affect the outcome.

Sørmo and Cassens [2004] distinguish between different types of explanation in CBR according to the *goals* they support, such as explaining why a question is relevant or how the system reached a proposed solution. In recommender systems that treat some or all of the user's requirements as constraints that must be satisfied, explanation can also play an important role in recovery from the *retrieval* failures that occur when there is no exact match for the user's requirements [e.g., Hammond *et al.*, 1996; McSherry, 2004b].

In comparison with recommendation strategies that insist on exact matching, an advantage of nearest-neighbor retrieval (and iNN) is that retrieval failures can never occur. In the absence of a case that exactly matches the user's query, she can be shown the case that is *most similar* to her query. That is not to say, however, that *recommendation* failures cannot occur. A recognized limitation of similarity-based retrieval, and one to which iNN is not immune, is that the most similar case may not be the one that is most acceptable to the user. Strategies for reducing the risk of recommendation failure include ensuring that the alternatives the user is offered are representative of *compromises* that the user may be prepared to accept [McSherry, 2003b] and reasonably diverse [e.g., Smyth and McClave, 2001]. However, these strategies cannot help if *none* of the cases available to the system is acceptable to the user. Typically in this situation, the user's only option is to look elsewhere for the product she is seeking.

Unfortunately, maintaining a product case base with sufficient coverage to ensure that recommendation failures can never occur is unlikely to be feasible in practice.

Moreover, balancing the trade-off between coverage and retrieval efficiency is one good reason for limiting case-base size and tailoring a CBR system to the needs of its most frequent users [e.g., Leake and Sooriamurthi, 2001]. As we show in Section 3, limiting the size of a product case base may also be beneficial in terms of the length of recommendation dialogues.

In this paper, we propose a multi-agent CBR approach to recommendation that assumes the existence of a collection of homogeneous recommender agents specializing in different areas of the product space (e.g., budget and luxury holidays). Each agent has its own case base and is capable of recommending products to its users with or without the cooperation of other agents. A recommendation failure is deemed to have occurred if none of the cases recommended by the agent to which a user submits her initial query is acceptable to the user. Such a failure triggers a *referral* of the user’s query to other recommender agents in search of additional cases to be considered as candidates in the recommendation process.

The agent initiating referral of an unsuccessful query is responsible for elicitation of any additional user preferences needed to discriminate between cases suggested by other agents, delivery of the final recommendation, and explanation of the recommendation process. Throughout the recommendation process, the user interacts solely and directly with the local agent in a recommendation dialogue driven by iNN, the conversational CBR approach used in Top Case [McSherry, 2003a; 2004a]. As we show in Section 5, the techniques presented can also be applied in a single recommender agent with access to multiple case bases.

In Section 2, we use a well-known case base in the travel domain to illustrate the iNN process in Top Case. In Section 3, we examine some of the trade-offs involved in dividing a product case base into smaller, more specialized case bases. In Sections 4 and 5, we present our multi-agent approach to recommendation and demonstrate the ability of a new version of Top Case to recover from recommendation failures when provided with an alternative source of cases. Related work is discussed in Section 6 and our conclusions are presented in Section 7.

2 Recommendation in Top Case

An initial query, if any, entered by the user is incrementally *extended* in Top Case by asking the user about her preferences with respect to attributes not mentioned in her initial query. An important role in the goal-driven approach to attribute selection that characterizes iNN is played by the concept of *case dominance* [McSherry, 2003a]. In the following definition, *Sim* is the similarity measure on which retrieval is based, and a given query is considered to be an extension of itself.

Definition 1. A given case C_1 dominates another case C_2 with respect to a query Q if $Sim(C_1, Q^*) > Sim(C_2, Q^*)$ for all possible extensions Q^* of Q .

One reason for the importance of case dominance is that is any case that is dominated by another case with respect to the user’s current query can be eliminated; clearly it can never emerge as the most similar case regardless of how the user chooses to extend her query. It can also be seen that if any case dominates all the other cases, then there is no need for the user’s query to be further extended, as the recommendation is certain to be the same regardless of the user’s preferences with respect to any remaining attributes.

McSherry [2003a] uses the triangle inequality to show that a given case C_1 dominates another case C_2 with respect to a query Q over a subset A_Q of the case attributes A if and only if:

$$Sim(C_1, Q) - Sim(C_2, Q) > \sum_{a \in A - A_Q} w_a (1 - sim_a(C_1, C_2))$$

where for each $a \in A$, w_a is the importance weight assigned to a , and $sim_a(C_1, C_2)$ is a local measure of the similarity between C_1 and C_2 with respect to a . The cases dominated by a given case with respect to a partially-elicited query can thus be determined with a computational effort that increases only linearly with the size of the case base.

At each stage of the recommendation dialogue, the attribute selected by Top Case is the one with the potential to maximize the number of cases dominated by the case it has selected as a *target* case. Starting with the case that is most similar to the user’s initial query, or a randomly selected case if there is no initial query, the target case is continually revised as the user’s query is extended.

The recommendation dialogue continues until it is certain that the outcome will be the same no matter how the user chooses to extend her query. At this stage, the user is informed that the target case has been confirmed as the recommended case. Any cases that equal the current similarity of the target case are also recommended.

The example case base that we use to demonstrate the recommendation process is a subset of the *Travel* case base (www.ai-cbr.org), a standard benchmark containing the descriptions of over 1,000 holidays. Case attributes and weights assigned to them in our experiments are price (8), month (7), location (6), persons (5), duration (4), type (3), accommodation (2), and transport (1). In our experiments, we divide the original case base into two smaller case bases called *Budget Travel* (523 cases) and *Luxury Travel* (501 cases). Budget Travel consists of all cases in the original case base with price ≤ 1500 , while Luxury Travel consists of all cases with price > 1500 .

Figure 1 shows an example recommendation dialogue in Top Case based on the Budget Travel case base. Minor editing in the interest of clarity includes the format of the user’s initial query.

Top Case: Please enter your query:
User: type = walking, month = may
Top Case: The target case is:
 Case 567: 764, may, alps, 3, 7, walking, flat, car (0.28)
 Other competing cases (493) include:
 Case 710: 807, may, giant mountains, 3, 7, walking, 2 star, coach (0.28)
 Case 699: 758, may, giant mountains, 2, 7, walking, 2 star, car (0.28)
 What is the preferred location?
User: **why**
Top Case: *Because if location = alps this will eliminate 470 cases, including Case 710*
 What is the preferred location?
User: **any**
Top Case: What is the preferred number of persons?
User: 2
Top Case: The target case is:
 Case 152: 449, may, black forest, 2, 7, walking, flat, train (0.5)
 Other competing cases (30) include:
 Case 699: 758, may, giant mountains, 2, 7, walking, 2 star, car (0.5)
 Case 501: 738, may, tyrol, 2, 7, walking, 4 star, car (0.5)
 What is the preferred accommodation?
User: 4 star
Top Case: The target case is:
 Case 501: 738, may, tyrol, 2, 7, walking, 4 star, car, (0.57)
 Other competing cases (3) include:
 Case 455: 778, may, salzbergerland, 2, 14, walking, 2 star, car (0.53)
 Case 435: 1008, may, carinthia, 2, 14, walking, 2 star, car (0.53)
 What is the preferred duration?
User: **why**
Top Case: *Because if duration = 7 this will confirm Case 501 as the recommended case*
 What is the preferred duration?
User: 14
Top Case: The target case is:
 Case 455: 778, may, salzbergerland, 2, 14, walking, 2 star, car (0.67)
 The only competing case is:
 Case 435: 1008, may, carinthia, 2, 14, walking, 2 star, car (0.67)
 What is the preferred price?
User: **any**
Top Case: The recommended cases are:
 Case 455: 778, may, salzbergerland, 2, 14, walking, 2 star, car (0.91)
 Case 435: 1008, may, carinthia, 2, 14, walking, 2 star, car (0.91)
User: **why** 455
Top Case: *Case 455 differs from your query only in accommodation and is highly recommended*
no matter what transport you prefer

Figure 1. Example recommendation dialogue in Top Case.

In response to her initial query, the user is shown Case 567, selected by Top Case as the target case, and the two most similar of the 493 cases that are not currently dominated by the target case. Note that cases in the Budget Travel case base retain their serial numbers from the original Travel case base.

Top Case now selects location as the most useful attribute for confirming the target case and asks the user what location she prefers. When the user asks why this question is relevant, Top Case explains that if the preferred location is *Alps*, this will eliminate 470 cases, including one of the cases the user was shown in the previous cycle.

Top Case now repeats the question and the user indicates her indifference to location by telling the system that *any* location is acceptable. Note that the current similarity of each retrieved case, shown in brackets, is normalized by the sum of the importance weights of all the case attributes apart from any attributes to which the user is indifferent.

Later in the example dialogue, the target case has changed to Case 501 in light of the user's preferences for a two-person holiday and 4-star accommodation (and her indifference to location). The target case now dominates all but three other cases in the Budget Travel case base. It can also be seen from the explanation that Top Case provides when asked to explain the relevance of duration that a preference for a one-week holiday would be enough to confirm Case 501 as the recommended case regardless of the user's preferences with respect to price and transport.

However, in light of the user's preference for a two-week holiday, the target case changes to Case 455. The user's indifference to price means that the target case cannot be discriminated from Case 435, which is equally similar to the user's query and offers the same means of transport.

As the recommendation is certain to be the same whatever the user's preferred means of transport, the dialogue can be safely terminated with Cases 455 and 435 as the recommended cases. When the user asks why Case 455 is recommended, Top Case points out that it differs from her query only in accommodation and is highly recommended no matter what form of transport she prefers.

Of course, it is possible that neither of the recommended cases, nor any of the other cases she was shown, may be acceptable to the user. In this case, the system has failed to deliver a successful recommendation. One possible approach to recovery from recommendation failure in this situation would be to explore *compromises* that the user might be prepared to make as in compromise-driven retrieval [McSherry, 2003b]. In Section 4, we present a multi-agent approach to recovery that is less dependent on the user's willingness to compromise.

3 One Case Base or Two?

As described in Section 2, we divide the Travel case base into two smaller case bases of roughly equal size called Budget Travel (price ≤ 1500) and Luxury Travel (price > 1500). In this section, we examine some of the trade-offs that may be involved if the two smaller and more specialized case bases are deployed in separate recommender systems. The original Travel case base provides a baseline for our evaluation of recommender performance.

One of the performance measures of interest is the average length of recommendation dialogues in the smaller case bases compared to that for the original Travel case base. We are also interested to know how often a typical user of the Budget Travel recommender would be better served by the Luxury Travel recommender and *vice versa*. Our analysis is based on the assumption that there is no cooperation between the separate recommenders. We also assume that users seeking a budget holiday will consult the Budget Travel recommender while users seeking a luxury holiday will consult the Luxury Travel recommender.

3.1 Recommendation Efficiency

Our empirical comparison of recommendation efficiency in the three case bases is based on the attribute selection strategy and dialogue termination criteria from iNN, the conversational CBR approach used in Top Case [McSherry, 2003a; 2004a]. Our experimental method is based on a *leave-one-out* approach. For example, we temporarily remove each case from the Travel case base, use its description to represent the preferences of a simulated user in a recommendation dialogue based on the remaining cases in the Travel case base, and record the number of questions asked before a recommendation is made. We repeat the same process for the Budget Travel and Luxury Travel case bases.

Observed dialogue lengths in the three case bases are summarized in Figure 2. A striking feature of the results is that dividing the original case base into smaller case bases has reduced average dialogue length in both of the smaller case bases. While dialogue length is never less than 4 in Travel, a recommendation is sometimes reached after as few as 3 questions have been asked in the smaller case bases. This benefit is particularly noticeable in Budget Travel, with an average dialogue length of only 5.2 compared to 5.7 in Travel. This amounts to a reduction in average dialogue length of 10% relative to the original case base, and a 35% reduction in query length relative to retrieval based on full-length queries involving all eight attributes in the domain.

However, it is worth noting that partitioning a case base into two smaller case bases cannot always be guaranteed to reduce average dialogue length in both partitions. In one experiment, we found that a reduction in

one partition was balanced by a similar increase in the other. The relationship between dialogue length and case-base size is one of the issues that we propose to investigate more fully in future work.

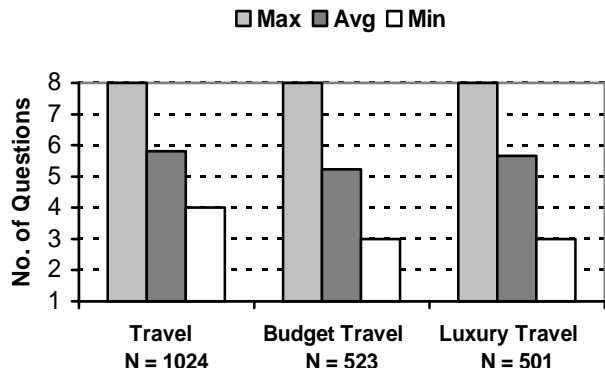


Figure 2. Lengths of recommendation dialogues in the Travel, Budget Travel, and Luxury Travel case bases.

3.2 Solution Quality

We have seen that one benefit of dividing a product case base into two smaller and more specialized case bases may be a reduction in the length of recommendation dialogues in both case bases. A more obvious benefit, particularly if the product case base is very large, is the increase in *speed* of retrieval that is bound to result from the system having fewer cases to process.

With no cooperation between the recommender systems in which the smaller cases bases are deployed, an expected trade-off for these benefits is a reduction in solution quality. Our second experiment aims to determine the extent of this trade-off as measured by the number of queries for which the most similar case in one of the smaller case bases is less similar than the most similar case in the other.

Again we use a *leave-one-out* approach, now applying NN retrieval to full-length queries as a basis for the similarity comparisons. First we temporarily remove each case from the Budget Travel case base, and use its description to simulate a query for a user seeking a budget holiday. For each such query, we apply NN retrieval to the Budget and Luxury case bases and note the location of the most similar case. We repeat the same process for the Luxury Travel case base, now using each left-out case to simulate a query for a user seeking a luxury holiday. For one such simulated query, an equally similar case was retrieved from each of the Budget and Luxury case bases. The results are summarized in Table 1.

A more similar case was retrieved from the Luxury Travel case base for only 13% of the 523 simulated queries originating from Budget Travel. This is an important result as it suggests that while access to a smaller case base is likely to have little impact on solution

quality for most users seeking a budget holiday, a minority of such users may benefit from referral of their queries to the Luxury Travel case base. The potential impact on solution quality appears to be greater for users of the Luxury Travel case base, with a more similar case being retrieved from Budget Travel for 21% of simulated queries.

In Section 4, we present a multi-agent approach to recommendation with support for referral of unsuccessful queries between recommender agents.

Table 1. Locations of most similar cases for simulated queries.

Query Origin	Location of Most Similar Case		
	Budget	Luxury	Both
Budget	453	70	0
Luxury	103	397	1

4 Multi-Agent Recommendation

Our multi-agent approach to recommendation assumes the existence of a collection of homogeneous recommender agents specializing in different areas of the product space (e.g., budget and luxury holidays). Each agent has its own case base and is capable of recommending products to its users with or without the cooperation of other agents. In either case, the user interacts solely and directly with the local agent to which she submits her initial query. Figure 3 shows a possible configuration with recommender agents RA1 and RA2 and case bases CB1 and CB2.

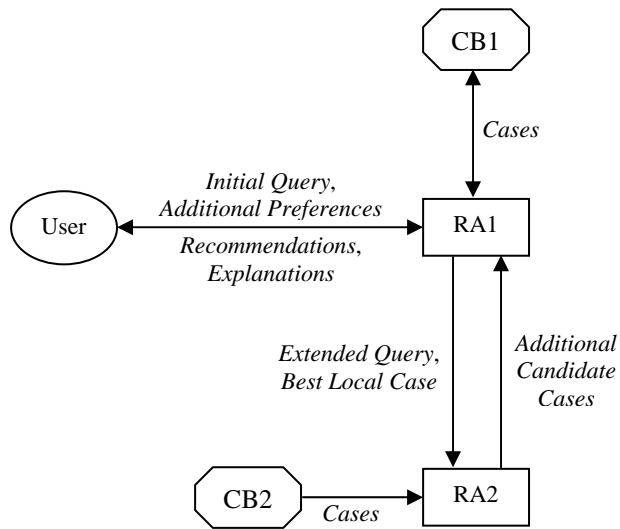


Figure 3. Referral of unsuccessful queries in multi-agent recommendation.

The user's initial query, if any, is incrementally extended by the local agent in a recommendation dialogue

driven by iNN, the conversational CBR approach used in Top Case [McSherry, 2003a; 2004a]. If none of the cases recommended by the local agent is acceptable to the user, this triggers a *referral* of the user's partially extended query, together with the "best" local case, to other recommender agents in search of additional candidate cases to be considered in the recommendation process.

The concept of case dominance on which attribute selection is based in iNN plays an important role in a remote agent's assessment of its ability to suggest cases that may be more acceptable than the best local case. As the user's query may be incomplete, the remote agent's task is not simply a matter of suggesting cases that are more similar to the current query than the best local case. It follows from the definition of case dominance in Section 2 that any case in the remote case base that is not dominated by the best local case has the potential to emerge as the most similar case following the elicitation of additional user preferences.

In summary, the cases suggested by the remote agent are those cases in the remote case base that are not dominated by the best local case. Of course, it is not unlikely that all cases in the remote case base are dominated by the best local case; in this case, the remote agent simply informs the local agent that it is unable to suggest any additional cases for consideration.

On completion of the referral process, the local agent discards any duplicates among the cases suggested by other agents and temporarily inserts the remaining candidates into the local case base. It may now be necessary for the user's query to be further extended to discriminate between cases suggested by other agents and the best local case. Any additional user preferences required for this purpose are now elicited by the local agent in a seamless extension of the recommendation dialogue. The local agent is also responsible for delivery of the final recommendation and explanation of the recommendation process.

5 Recommendation in Top Case 2

Our approach to recovery from recommendation failure can also be applied in a single recommender agent with access to multiple case bases. In this case, a recommendation failure in the local/default case base triggers the retrieval of cases from the other case bases that are not dominated by the most similar local case. We now use our Budget and Luxury versions of the Travel case base to demonstrate the ability of Top Case 2, an extended version of Top Case, to recover from recommendation failures in one case base when provided with access to the other.

To illustrate the recovery process, we return to our example in Section 2 of a user seeking a two-week walking holiday in May with 4-star accommodation for two persons. With Budget Travel as the default case base

in Top Case 2, and type = walking, month = may as the user's initial query, the first part of the recommendation dialogue is the same as in Figure 1. In Figure 4, we pick up the dialogue at the point where Cases 455 and 435 are presented as the recommended cases.

Suppose now that neither of the recommended cases is acceptable to the user, for example because she does not wish to travel by car and/or is not prepared to settle for 2-star accommodation. As shown in Figure 4, the user can now ask for her query to be referred to the Luxury Travel case base. In response to this request, Top Case 2 retrieves the only three cases that are not dominated by Case 455, the first of the initially recommended cases, from the Luxury Travel case base and temporarily inserts them into the Budget Travel case base.

As the new cases are more similar to the user's query than the initially recommended cases, one of them is selected as a new target case. Although the initially recommended cases are dominated by the target case, the user's query must now be further extended to discriminate between the cases borrowed from the Luxury Travel case base. Top Case 2 now asks the user which method of transport she prefers and explains that if she prefers to travel by car, this will eliminate Case 218. But as the user prefers to travel by plane, Case 218 is instead confirmed as the recommended case.

Given its similarity to the user's final query, and the user's indifference to price (at least as indicated in her initial dialogue with the Budget Travel recommender), Case 218 seems likely to be more acceptable than the initially recommended cases. As explained by Top Case 2 when asked why it is now recommending this case, it differs from the user's query only in accommodation.

Currently in Top Case 2, a query is referred only at the user's request. Of course, such a request does not necessarily mean that the system has failed to deliver a successful recommendation; the user may simply be curious to see what alternatives are available elsewhere. In future research, we propose to investigate a variety of alternative referral strategies and their impact on referral frequency and success. For example, referral could be automatically triggered by a low similarity score for the best local case.

6 Related Work

Giampapa and Sycara [2001] describe the use of Aha *et al.*'s [2001] NaCoDAE, a generic tool for conversational CBR, to generate plan objectives for team agents in their multi-agent system RETSINA. The authors demonstrate NaCoDAE's ability to proactively dialogue with other agents in a goal-directed approach to gathering relevant information in a command and control scenario. In contrast to our work, the role of the case-based reasoner is to converse directly with other software agents and not with the human user.

Top Case 2: The recommended cases are:
 Case 455: 778, may, salzbergerland, 2, 14, walking, 2 star, car (0.91)
 Case 435: 1008, may, carinthia, 2, 14, walking, 2 star, car (0.91)

User: **why** 455

Top Case 2: *Case 455 differs from your query only in accommodation and is highly recommended no matter what transport you prefer*

User: **refer**

Top Case 2: Now referring your query (type = walking, month = may, location = any, persons = 2, accommodation = 4 star, duration = 14, price = any) to the Luxury Travel case base.
 The target case is:
 Case 108: 3768, may, harz, 2, 14, walking, 5 star, car (0.93)
 The only competing cases are:
 Case 218: 2578, may, corsica, 2, 14, walking, 3 star, plane (0.93)
 Case 89: 1828, may, erzgerbirge, 2, 14, walking, 3 star, car (0.93)
 What is the preferred transport?

User: **why**

Top Case 2: *Because if transport = car this will eliminate Case 218*
 What is the preferred transport?

User: plane

Top Case 2: The recommended case is:
 Case 218: 2578, may, corsica, 2, 14, walking, 3 star, plane (0.98)

User: **why** 218

Top Case 2: *Case 218 differs from your query only in accommodation*

Figure 4. Recommendation failure and recovery in Top Case 2.

McGinty and Smyth [2001] propose an approach to personalized route planning in which problem-solving experience is distributed among multiple CBR agents. As in our approach to multi-agent recommendation, collaboration between agents is essentially *failure driven* in that individual agents rely on assistance from other agents only when confronted with route-planning problems in unfamiliar territory that they cannot easily solve. As user preferences are not explicitly modeled in the approach, and a target problem is simply defined by a start and goal location, there is no requirement for incremental elicitation of user preferences as in multi-agent recommendation.

Multi-case-base reasoning (MCBR) extends CBR to draw on multiple case bases that may address somewhat different tasks [e.g., Leake and Sooriamurthi, 2001; 2003]. It reasons about issues such as whether to solve a problem locally or dispatch it to an external case base, and how to adapt the solution obtained to account for differences in tasks and/or task environments. Similar requirements for *cross-case-base* adaptation are likely to arise in multi-agent

recommendation if the participating agents are not homogeneous, for example with products priced in different currencies in different case bases.

Leake and Sooriamurthi [2003] show that MCBR applied to two case bases that differ in task environments can sometimes improve solution quality compared to merging the separate case bases into a single case base. They also show that MCBR is potentially useful as a basis for guiding selective case addition; that is, cases are added only when required to fill competence gaps that affect the current performance of the CBR system.

Selective retention of cases is another potential benefit of multi-agent recommendation that we propose to investigate in future research. Our mechanism for referral of unsuccessful queries to other recommender agents, together with the most similar local case, provides an opportunity for retention of both referred and suggested cases that may be mutually beneficial to the participating agents, thus providing an *incentive* for cooperation that might otherwise be lacking.

7 Conclusions

We have presented an approach to multi-agent recommendation which combines the benefits of iNN, the conversational CBR approach used in Top Case [McSherry, 2004a], with the benefits of access to a potentially unlimited range of products. One such benefit is a simple recovery mechanism in which a local recommendation failure triggers a referral of the user's query, together with the "best" local case, to other agents in search of additional candidate cases to be considered in the recommendation process. The concept of case dominance on which attribute selection is based in iNN [McSherry, 2003a] also plays an important role in a collaborating agent's assessment of its ability to suggest cases that may be more acceptable than the best local case.

The techniques presented can also be applied in a single recommender agent with access to multiple case bases. We have also demonstrated the ability of an extended version of Top Case to recover from recommendation failures when provided with access to an alternative source of cases. Other potential benefits of multi-agent recommendation that we propose to investigate in future research include the opportunity that the referral process provides for selective retention of referred and suggested cases, and the incentive this provides for cooperation between agents.

References

- [Aha *et al.*, 2001] David Aha, Leonard Breslow, and Héctor Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14:9-32, 2001.
- [Giampapa and Sycara, 2001] Joseph Giampapa and Katia Sycara. Conversational case-based planning for agent team coordination. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 189-203, Vancouver, BC, Canada, 2001. Springer-Verlag.
- [Hammond *et al.*, 1996] Kristian Hammond, Robin Burke, and Kathryn Schmitt. A case-based approach to knowledge navigation. In David Leake, editor. *Case-Based Reasoning: Experiences, Lessons & Future Directions*, pages 125-136. AAAI Press, Cambridge, Massachusetts, 1996.
- [Herlocker *et al.*, 2000] Jonathan Herlocker, Joseph Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 241-250, Philadelphia, Pennsylvania, USA, 2000. ACM Press.
- [Kohlmaier *et al.*, 2001] Andreas Kohlmaier, Sasha Schmitt, and Ralph Bergmann. A similarity-based approach to attribute selection in user-adaptive sales dialogues. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 306-320, Vancouver, BC, Canada, 2001. Springer-Verlag.
- [Leake and Sooriamurthi, 2001] David Leake and Raja Sooriamurthi. When two case bases are better than one: exploiting multiple case bases. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 321-335, Vancouver, BC, Canada, 2001. Springer-Verlag.
- [Leake and Sooriamurthi, 2003] David Leake and Raja Sooriamurthi. Dispatching cases versus merging case bases: when MCBR matters. In *Proceedings of the Sixteenth Florida Artificial Intelligence Research Society Conference*, pages 129-133, St. Augustine, Florida, 2003. AAAI Press.
- [McGinty and Smyth, 2001] Lorraine McGinty and Barry Smyth. Collaborative case-based reasoning: applications in personalized route planning. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 362-376, Vancouver, BC, Canada, 2001. Springer-Verlag.
- [McSherry, 2001] David McSherry. Interactive case-based reasoning in sequential diagnosis. *Applied Intelligence*, 14:65-76, 2001.
- [McSherry, 2003a] David McSherry. Increasing dialogue efficiency in case-based reasoning without loss of solution quality. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 121-126, Acapulco, Mexico, 2003. Morgan Kaufmann.
- [McSherry, 2003b] David McSherry. Similarity and compromise. In *Proceedings of the Fifth International Conference on Case-Based Reasoning*, pages 291-305, Trondheim, Norway, 2003. Springer-Verlag.
- [McSherry, 2004a] David McSherry. Explanation in recommender systems. In *Proceedings of the ECCBR 2004 Workshops*, pages 125-134, Madrid, Spain. Universidad Complutense Madrid.
- [McSherry, 2004b] David McSherry. Incremental relaxation of unsuccessful queries. In *Proceedings of the Seventh European Conference on Case-Based Reasoning*, pages 331-345, Madrid, Spain, 2004. Springer-Verlag.
- [Shimazu, 2002] Hideo Shimazu. ExpertClerk: A conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, 18: 223-244, 2002.
- [Sørmo and Cassens, 2004] Frode Sørmo and Jörg Cassens. Explanation goals in case-based reasoning. In *Proceedings of the ECCBR 2004 Workshops*, pages 165-174, Madrid, Spain. Universidad Complutense Madrid.
- [Smyth and McClave, 2001] Barry Smyth and Paul McClave. Similarity vs. diversity. In *Proceedings of the Fourth International Conference on Case-Based Reasoning*, pages 347-361, Vancouver, BC, Canada, 2001. Springer-Verlag.

Utilizing Multiple Agents In Dynamic Product Recommendations

Timothy A. Musgrove and Robin H. Walsh

CNET Networks, Inc.

Artificial Intelligence Research Group

235 Second Street, San Francisco, CA 94105 USA

{tim.musgrove, robin.walsh}@cnet.com

Abstract

One application of recommendation systems is the “upgrade-or-replace” scenario, wherein a user faces the decision of whether (and how) to improve an existing system with add-ons, versus replacing the system altogether. This paper describes a project, now entering its mature phases at CNET Networks, for explaining to users, in natural language, how an automated multi-agent recommender has addressed multiple goals or values that can apply to such a situation. The solution invokes four agents, which interact with one another: a product value scoring agent, a product functionality evaluation agent, a competitive product discovery agent, and an accessory discovery agent. The examples chosen relate to personal computers and digital photography systems, where some partial results from live tests on CNET’s websites have been obtained.

1 Introduction

Any long-term utilization of a system, i.e. an apparatus with numerous modules, will inevitably present administrators and/or users with the tasks of system maintenance, enhancement, upgrade, and eventual replacement of the system. In many domains, there exists a wide variety of available systems and component modules, such that it is possible (and prudent) to compare and “shop around” for either a new system, or for upgrades and add-ons to one’s existing system. This description would apply, for example, to the automobile one drives, the barbeque one uses to cook food outdoors, and the notebook computer one hauls back and forth between office and work. As the word is used here, each of these things is a “system” because it consists of some interchangeable (and independently configurable) components working together. Because the task of finding an upgrade or replacement for these things is so complex, people often consult various sources of advice when considering procurement.

The primary categories of such advice are those answering the two questions of “what to buy” and “where to

buy.” Both of these questions are approached usually from an absolutist point of view, which is to say, a non-contextual standpoint; the question of what to buy is equated with the question of what is objectively the best product or the best “deal” (if indeed such a judgment can be made objectively), and where to buy is equated to which vendor has the lowest price. This approach does not produce the best end-result for the user, whose aim usually is to compare the item(s) in question with what he or she already owns and uses. Or in another case, which turns out to be analytically very similar, the user has a “leaning” toward a product he or she wants to buy, but wishes to explore other options in order either to confirm or disconfirm this tentative choice. In either case, the user needs to see contextually relevant comparisons with respect to existing products that mark their starting point. Often this starting point is more relevant to users’ needs than is, say, their history or profile of past purchases or activity with the website, the likes of which are taken as the primary input of some other recommender systems, e.g. Ghani and Fano [2002].

Such a user needs to determine first whether a given system is “obsolete” to the degree that an alternative is recommended at all, and secondly, whether it is not better to upgrade that system rather than to replace it completely, and thirdly, where (or from whom) to make such acquisition. Marketers of course are eager to rush the user to the third of these steps. Our present interest however is in the first two steps.

Our multi-agent recommendation system involves four interdependent agents who operate upon a shared knowledge base. The knowledge base blends some rankings and ratings from domain experts with many data, secondary attributes and statistical functions representing the ever-changing marketplace of available products. All four agents draw from and contribute to the knowledge base, each with its own distinctive interest or goal. The first agent has the goal of determining whether a product or system offers good features relative to its price; the second, whether a system or a product offers robust and useful features compared to what is standard for that type of product; a third determines for any given product, what competitive products exist that can offer material advantage

in terms of either cost or features; and a fourth discovers a diverse assortment of validated accessories, parts, and add-ons for a product or system, as an alternative to replacing that system.

We call these “agents” in respect of their following behaviors: (1) each one passes evaluative judgment, positive or negative, upon a candidate object; (2) each one re-considers and changes its judgments dynamically, in the face of an ever-changing data set; (3) after having had the initial input of certain rankings and preferences from humans, each one makes its judgments without further human intervention; (4) each one makes judgments that are not wholly determined by the inputs given it by humans. This paper discusses each of these agents in turn, and how they inter-operate.

Although these recommendation agents are being applied to computer and electronics products for consumers, their underlying methods are applicable to many other types of products and systems (e.g. barbecues, jet skis, food processors, etc.), and one could readily apply some of their methods even to things that are not durable goods (such as martial arts programs or vacation packages).¹

2 Automated product cost evaluation agent

This agent’s purpose is to compare a product’s features to its typical cost, in order to evaluate whether the features provide more, or less value than one should expect at that price point. The most important feature of this agent is its “competitive ranking” method. Competitive rank of a product’s attribute value for a particular attribute, is a function not only of the attribute value’s place within the scale of possible values for that attribute, but also of the distribution of such values among competing products. As such, the curve of competitive ranks of all products reflects what the market says about the commonness or scarcity of an attribute value. Consequently, it will not follow the linear scale for the attribute in question. This is a positive feature of the agent, as it represents that some brackets within a value scale are more meaningful to consumers than others (e.g. the difference in value between a 1 and 2 megapixel camera is greater than the difference between a 5 and 6 megapixel camera).

It is worthwhile to explain how the competitive rank function is calculated, and why it is used instead of the more common percentile rank.

2. 1 Competitive rank vs. percentile rank

One of the most commonly used ranking functions in statistics packages today is “percentile rank,” defined as the percentage of items in the data set which are below the value in question. Two problems arise with the use of this measure for analyzing the feature-to-price relations of products.

First, by definition, the average percentile rank of a set of competing products is not guaranteed to be 0.50, and in fact will move around as the data changes. This means it is difficult to build a balanced index from percentile-ranked values without continual re-normalization of the index.

Second, percentile rank does not depict a sound estimation of top and bottom values in situations where the number of possible values is small, e.g when the values range merely from the integers 0 to 3. Specifically, the percentile rank is *always* zero for the bottom value, regardless of how common or how rare it is for an item to have such value. For example, if nearly all cameras in the market lack a certain feature (have 0 value for that feature) versus only one camera lacking that feature -- either way, the percentile rank is the same (i.e., zero) for a camera lacking that feature.

However, it is a commonplace of economics that a product’s lacking a feature (or “value” or “utility”) which nearly all competitors have, is usually perceived by consumers as *worse* than lacking a feature that is hard to come by. Competitive rank accounts for this by setting each rank as the center-percentile of the bracket of a particular value’s distribution, as illustrated in this chart:

Table 1. Illustration of the difference between percentile rank and competitive rank

Percentile vs. Competitive Ranks	Values listed from high to low (total =10)	Percentile rank	Competitive rank
Values	3	0.9	0.95
	2	0.7	0.8
	2	0.7	0.8
	1	0.3	0.5
	1	0.3	0.5
	1	0.3	0.5
	1	0.3	0.5
	0	0	0.15
	0	0	0.15
	0	0	0.15
Average	1.1	0.35	0.5

¹ The various software components described herein are the subjects of patents pending with the USPTO on behalf of CNET Networks, Inc.

In the above example, because the bracket of value “0” extends from the 0th to the 30th percentile, its competitive rank is $(.30-0)/2 = .15$; whereas the bracket for value “1” extends from the 30th to the 70th percentile, so its competitive rank is $(.70-.30)/2 = .50$; and so on. Note that the average is 0.5 – and always will be (after rounding). Also note that, according to this formula, if, instead of 3 products (out of 10) having “0” value, 6 of them did, then the competitive rank of “0” would change from 0.15 to 0.30. These are the features which percentile rank lacks, but which competitive rank provides. It is a numerical way of expressing that if most products in a category lack attribute X (have a “0” value for X), then the rank of “0” is not as bad as it would be, were it the case that most products in the category had a non-zero value for X.

2.2 Automatically measuring “bang-for-the-buck”

Once the competitive ranks are calculated, the price offset of each product from the average price of all products is distributed in proportion to the difference in competitive ranks (after weighting according to importance levels, e.g. that the weight of a notebook computer is far more important than, say, the length of its power cord) of all the attribute values of that product. Once this has been done for each product, we effectively have attribute-based price-differentials for each product, e.g. if a camera costs \$120 dollars more than average, we will have attributed, say, +\$48 of that differential to the camera’s having a 0.08 better-than-average competitive rank on its “zoom” attribute. In some cases, these will be negative instead of positive (e.g. the product may be attributed -\$12.50 for having a shorter-than-average battery life).

Once these assignments are made to each attribute of every product, they are averaged across all products for each attribute, giving us the average dollars-per-competitive-rank-point for each attribute. E.g., “zoom” may be worth \$12.80 per each 0.01 of competitive rank difference, meaning that when a camera is better than average (above 0.50) on its zoom competitive rank, it will be worth an additional \$12.80 for every 0.01 that it is above 0.50. This will differ for each attribute.

Finally, these are applied back to each product according to the competitive rank of all its features. The result is a “value price” – the theoretical price that the product is worth in the market. From the value price we readily derive the “value score” by spreading any differential between the value price and the actual street price across a 1-10 scale, which is divided by increments of standard deviation. Value scores will average 5.0, and those above 5.0 will mean “more features for the dollar” than the average product in the market; those with scores below 5.0 have “less features for the dollar” than the average product in the market. Because rarely a day goes by without new products entering the market, or old products leaving, or prices

changing on existing products, the value scoring agent is constantly analyzing things in the background, in order to react to these changes and re-calibrate the competitive ranks of the entire data space, assigning an adjusted score to every product on a daily basis.

We ran tests to compare the value scoring agent with the competitive rank described above, versus using the conventional percentile rank. Using notebook computers as our test category, we checked the Cronbach alphas, and also the Pearson correlation between the agent’s value scores and those of a manually created value index that has been accepted by the industry over a number of years.² As shown in Table 2, the Cronbach alpha improved from 0.58 to 0.76 moving from percentile rank to competitive rank; the Pearson correlation (with acceptable r-values) improved from 0.66 to 0.84. These results encouraged us to test the agent’s output with real users.

Table 2. Ranking Function Comparison

<i>Ranking Function</i>	<i>Cronbach</i>	<i>Pearson</i>
Percentile Rank	0.58	0.66
Competitive Rank	0.76	0.84

Early usability tests showed that while some users appreciated the meaningfulness of our 1-10 scale numbers, many did not, until after the numbers were explained to them in depth. To remedy this, a natural language component was added, so that the agent now actually produces a short paragraph (typically two or three sentences in length), in place of the actual number. This paragraph explains the general value score of the product in a more “common sense” way and picks out the top attribute or two that contributed most heavily to that score (details and an example to follow).

3 Product functionality evaluations

The product functionality evaluator agent utilizes the competitive ranks in a different manner; instead of relating them to price, it looks at the distance from 0.50 of a product’s competitive ranks for the more important features, and produces natural language evaluative statements – essentially the pros and cons of the product compared to what is typical or standard within the current market.

Naturally there is not just one way to evaluate a product, so the agent relies upon a knowledge base that provides several alternative ranking schemata grouped into “usage

² Specifically, the CVI (CNET Value Index) compiled by Russ Gilchrist of Computer Shopper Magazine, and made available to advertisers in that magazine.

scenarios,” so called because they reflect different emphases of usage for the same type of product. Examples of usage scenarios for notebook computers would be mainstream (the default usage scenario), ultra-mobile, desktop-replacement, and entertainment-oriented. An example of how the rankings can vary, is that a larger screen size is ranked with high importance in the desktop-replacement scenario, while weight is of less importance; exactly the reverse holds for the ultra-mobile scenario. Subject matter experts (SME’s) were consulted on the rankings of the scenarios.

3. 1 NL presentation and user reaction

The product functionality evaluation agent also uses a natural language generator to create paragraphs explaining the pros and cons of the product. In some cases, the agent will cite an example of how another product in the market offers a superior feature. The natural language generator employs some randomly alternating sentence templates with slots filled in dynamically via shallow linguistics and micro-grammar, so as to produce several thematically defined paragraphs. It uses some familiar elements of NLG, such as a knowledge source, a set of well-defined communicative goals, a running discourse history, and a distinction between depth and surface grammar – much like some of the systems described by Reiter and Dale [2000]. (On top of this, we took pains to allow human editors the ability to set rules forcing the inclusion of a few “canned phrases” along with the dynamically generated segments of sentences.) An example set of paragraphs is shown here:

Description: The Fujifilm FinePix S3100 is a 4-megapixel, point-and-shoot digital camera. Compared to other point-and-shoot cameras on the market, it is relatively inexpensive at around \$270. Includes ImageMixer VCD2 and FinePix viewer software.

Pros: Has a 6x optical zoom, which is considerably more powerful than some similarly priced cameras. By way of comparison, the typical cost of a product with this high an optical zoom is \$455.

Cons: Has only a 1.5 inch viewscreen, which is smaller than some similarly priced cameras. Also, this camera weighs in at 9.6 ounces, which is somewhat heavier than some similarly priced cameras. The 39mm minimum focal length of this camera will make it difficult to photograph landscapes or architecture.

Suitability: Suited for the average, casual picture-taker. The 234mm maximum focal length of this product makes it suitable for taking pictures of wildlife, candid portraits, and other long-distance photography. A

recommended use, based on key features, is framed photos for the home. The largest recommended print size, based on the camera's resolution, is 8x10.

Value: In general, this camera's features are in line with its price. However, the average price of point-and-shoot cameras with similar resolution is considerably more, around \$310.

Note that if the pricing or features change among other cameras in the market place, any sentence in the above paragraphs could potentially change. Such textual change is a result of the interaction of the agents on an automated basis, and is not supervised by humans. This is one of the most important results of the multi-agent system: the content is kept “evergreen” to a degree that would be unthinkable to do manually on a weekly or daily basis. As of this writing, more than 20,000 such agent-generated recommendation texts are published on CNET’s websites, refreshed weekly; this compares to about 5,000 evaluative texts written by humans, updated or replaced every several months.

Impact on user behavior has been fairly dramatic. “Page stickiness” (non-abandonment of a product evaluation page by users) has improved at least 12%, and up to 20% where such texts have been used on the web site, and positive user feedback during a 12-month beta cycle has outnumbered negative feedback by about a 4:1 ratio. The number of paragraphs containing errors is estimated to be much lower than 1%. From these and other metrics, CNET has concluded that the agents here described are producing texts deemed useful by most users, bearing all the features we have found to obtain for value-added content in an advice-giving environment. Most site visitors do not realize, despite the presence of all the appropriate disclaimers, that humans do not write these texts.

4 Accessory discovery agent

This agent uses as its primary resource the thousands of cross-sell items from product pages on the websites of several hundred online merchants. The premise in using these data as starting points, is that since marketers aim to use their limited amount of “screen real estate” only for products that “move” (and that are not frequently returned by users after purchase), they effectively fine-tune these placements with respect to what users find most relevant. Therefore, when a sufficiently broad network of merchants shows a pattern of cross-selling the same accessory for the same product, chances are good that it is a compatible, popular, and useful accessory.

The noisy data that accompanies this, however, is not to be taken lightly. Merchants also rotate random cross-sell

items that are unrelated to, and sometimes even incompatible with the products they are placed against. They also push some accessories that are not the best fit, but have the highest profit margin, or the greatest surplus inventory that the merchant wants to unload. These factors tend to be cancelled out when there is a sufficiently large pool of merchant offers from which to draw data. That is why the agent looks for multiple attestations of accessory relations among several merchants, before concluding that an accessory is of genuine value.

By aiming this “collaborative” approach to *merchant* data, instead of to the more commonly used *user* data, the agent can handle cases where there is a sparsity of user data (i.e., accessories not yet clicked on by very many of our users) – a problem which takes considerable effort to overcome in user-centric collaborative filtering mechanisms, as illustrated in Huang *et al.* [2004].

The agent has the ability to compare language in the descriptions of these products, against its knowledge base of known features (which includes an index of variant language for describing the same feature) to determine approximate sameness of accessories. This is to prevent the agent from recommending a top bundle of accessories that is functionally redundant, such as, for a digital camera, three memory cards of different sizes or brands. Such redundancy could initially occur because those three memory cards were the items most broadly and strongly attested within the merchant network. However, the agent will recognize their similarity, and include only one of them, moving on to include perhaps a telephoto lens and water-resistant case in its top recommendation set. With the elimination of similar accessories ensuring a diverse package of add-ons, users can see how their camera’s functionality could be expanded, as opposed to replacing it.

We have found that users respond well to having the accessories shown in a list format. Nonetheless, as the different agents are connected, the product function evaluation agent has the capacity to “borrow” the knowledge of the accessory agent, and thereby can generate sentences about the capacity to augment a product with add-ons, such as in the example, referring to a handheld personal digital assistant (PDA):

Suggestions: (1) If you plan to enter large amounts of data, consider purchasing an [external portable keyboard](#) for your PDA. (2) If you want to synchronize data and documents with Microsoft office, consider a 3rd party application such as Documents To Go.

5 Automated discovery of competing products

The competing products discovery agent behaves similarly to the product functionality evaluation agent, except that it

focuses on competitive products, in relation to a given product, to find those that offer one type of advantage or another. First the agent identifies qualified candidates, construed as those products that are roughly similar in their descriptions and specifications. The similarity measure is somewhat like that described by Fleischman and Hovy [2003], with the difference that some numerical attribute value sets have been scalarized by subject matter experts (e.g., that higher RAM counts as better but lower weight counts as better). Next the agent identifies, from among the qualified candidates, which ones exhibit a compelling advantage. This could mean a product with better features at around the same price, or with the same features at a lower price, or with a slightly higher price but much better features, or with the same features and price but in a different form factor, and so on. The following is an example of this agent’s auto-generated text:

Better features, lower price

To help decide if the Fujifilm FinePix S3100 is your best choice in a camera, you may want to check out the HP Photosmart M307, as it will provide you certain additional benefits at a lower price. For example, it weighs just 5.1 ounces, which is relatively lighter than the Fujifilm FinePix S3100 (9.6 ounces). The HP Photosmart M307 also has a 1.8 inch viewscreen, which is larger than that of the Fujifilm model (1.5 inch). Finally, the HP Photosmart M307 usually goes for about \$165, compared to \$270 for the Fujifilm FinePix S3100.

Next the system will endeavor to generate additional such texts for other scenarios, e.g. superior features in a moderately higher priced model, or similar features in a different form factor, etc. Each alternative scenario includes a compound of common user preference types (e.g. price, features, brand, form factor, etc.) Finally, several of these alternative scenarios are offered to users as successive paragraphs on a Web page. This “menu” presentation is an alternative way of addressing users’ variant preferences, as against the “conversational” approach employed by Smyth *et al.* [2004]. (We pursued this alternative after our focus group studies showed that only a small percentage of users would pursue a dialogue-based approach.)

6 Conclusion: using multiple agents to create a total advice context

The four agents working together create a multi-faceted advice-bearing web page, wherein the pros, cons, competing products, and suggested accessories are always contextually relevant, and are explained in terms of how their specific

features compare to the original product. A “healthy tension” exists between and among the four agents, much to the user’s benefit. One agent seeks to maximize features relative to cost; while another seeks to make sure one’s product is “up to par” on features, even if that would require more cost. A third agent seeks to find the best package of accessories and add-ons that could enhance a product or system, and thereby forestall the need for its replacement. A fourth agent seeks to show, for any given product, which competing products have some superior advantage to offer.

Each of the agents contributes in some way to the generation of natural-reading text. Together, they constitute a scalable advice system that offers recommendations on many thousands more products than would be feasible to review manually. In the future, CNET’s AI Research Group will experiment with additional recommender agents sharing the same knowledge base, such as agents to summarize user opinions, explain the differences between brands, or summarize recent industry trends.

References

- [Fleischman and Hovy, 2003] Michael Fleischman and Eduard Hovy. Recommendations without user preferences: a natural language processing approach. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 242–244, Miami, Florida, USA, 2003. ACM Press.
- [Ghani and Fano, 2002] Rayid Ghani and Andrew Fano. Building recommender systems using a knowledge base of product semantics. *Workshop on Recommendation and Personalization in ECommerce (RPEC 2002)*, Malaga, Spain, 2002.
- [Huang et al., 2004] Zan Huang, Hsinchun Chen, Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. In *ACI Transactions on Information Systems (TOIS)*, Volume 22, Issue 1 (January 2004), pages 116-142, New York, NY, USA, 2004. ACM Press.
- [Reiter and Dale, 2000] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems (Studies in Natural Language Processing)*, Cambridge, England, 2000. Cambridge University Press.
- [Smyth et al., 2004] Barry Smyth, Lorraine McGinty, James Reilly, Kevin McCarthy. Compound critiques for conversational recommender systems. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI’04)*, pages 145-151, Beijing, China, 2004, IEEE Computer Society.

Discovery of Business Opportunities on the Internet with Information Extraction

François Paradis and Jian-Yun Nie
Université de Montréal
Québec, Canada
{paradif,nie}@iro.umontreal.ca

Arman Tajarobi
Nstein Technologies
Québec, Canada
arman.tajarobi@nstein.com

Abstract

In this paper we describe a tool for the discovery of business opportunities on the Web. The aim of our system is to help a user decide which call for tenders should be examined further. The project and its goals will be presented. We then focus on one aspect, *named entity* extraction, and its use to improve classification and user navigation. We show that filtering contents improves the classification results, but so far we did not get a significant improvement by combining named entities and extraction. However there are benefits to the interface to facilitate query refinement and collection browsing.

1 Introduction

Finding and selecting business opportunities is a crucial activity for businesses, as evidenced by the recent interest in Business Intelligence [2]. A key functionality is the ability for suppliers to find Call for Tenders (CFT) relevant to their businesses, based on their domain of expertise, the geographical location, the submission/execution dates, the contract size, etc. Sometimes the only information freely available is the solicitation notice: a fee must be paid to obtain the full documentation. Thus it is important to select the right candidate CFTs.

Large organisations can have staff dedicated to the purpose of finding CFT. For smaller organisations, or for secondary suppliers that might not respond directly to CFT but want to monitor the activity in their sector, tendering services are offered. Typically a professional will have a portfolio of enterprises and send them digests of candidate CFTs on a routine basis. Many electronic tendering sites are also available, usually covering an economic zone (e.g. “TED” for the European Union, “SourceCan” for Canada) or sector of activity (e.g. “NetMetal” for the metal industry in Canada). While these sites do increase the accessibility to the data, an organisation trying to go outside its usual sphere of activity will still have difficulties. Firstly it has to deal with country/domain specific standards, such as regulations and classification codes. Secondly it might be missing some contextual information, such as the recurring nature of the tender, the tendency of the contracting authority to select local suppliers, etc.

Our solution is to augment the original data with extracted information from CFT and other supporting documents, and to cross-reference this information in such a way that the user can discover the patterns of a company or an industry, or relate to information she is familiar with. For example, multi-classification can provide more than one angle to a CFT (for instance, a code for a product-oriented schema, and a code for a service-oriented schema), or an alternative schema when the user is not familiar with the schema provided in the CFT. By combining information from CFTs and awards, a pattern of the activity in a sector, or the business relationships between organisations, can be presented to the user.

Two challenges are: how to discover new CFT and related documents on the Web, and how to extract information from these documents, knowing that the Web offers no guarantee on the structure and stability of those documents. There is a much work lately on Intelligent Web Robots [4] and topic-focused Web crawling [1]. In our case, however, we are not just crawling for a “topic”, but rather for certain types of documents. Moreover, the typical “crawl” strategy might not be adequate here, since the information is often not linked directly (for example, a company site will not offer links to its competitors). To extract information, *wrappers* [13], i.e. tools that can recognise textual and/or structural patterns, will have limited success because of the diversity and volatility of Web documents. Free-text techniques such as named entity extraction [12], do not assume a particular document model or structure, but have lower accuracy.

In this paper we describe a tool to help the discovery of business opportunities on the Web. In particular we focus on Information Extraction, and its use to improve classification and user navigation. We first present the general framework of the project and the test collection we built to evaluate our results. We then discuss the application of Information Extraction in the project. We present some results of filtering contents based on named entities. We also show how these named entities are used to aid the user refining her queries and browse the information space.

2 The MBOI Project

The MBOI project (Matching Business Opportunities on the Internet) deals with the discovery of business opportunities on the Internet. In the first phase of the project we have implemented a tool to aid a user in this process. This first

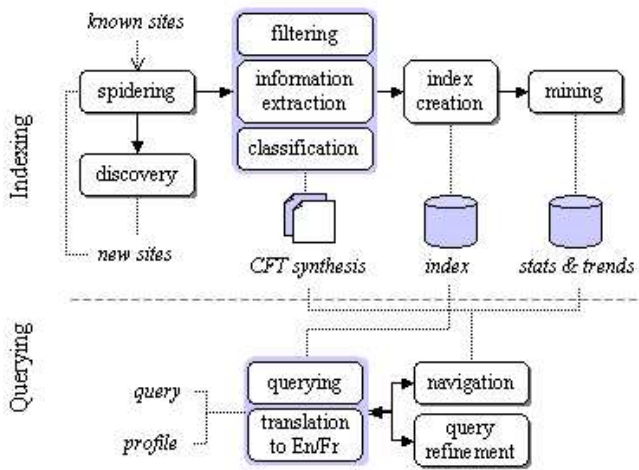


Figure 1: System Architecture

system, called CERVO was developed by CIRANO (Inter-university Centre of Research on the ANalysis of Organizations), and was used by SDTI (Information Technology Development Service) and its professionals to help businesses located in the Ste-Hyacinthe area (Quebec) in the search for CFTs. Other applications followed, notably a business portal for the metal industry in Canada, NetMetal. In the latter, rather than defining profiles for each organisation, some key profiles were defined corresponding to simple themes; users could then express complex queries by combining these profiles. Finally, it should be noted that CERVO was also applied to another domain, namely the tourism industry, in a project with UQAM (University of Quebec in Montreal). There again, although the information needs are typically less precise and more changing, the basic process is very similar to a business watch.

Figure 1 shows the system architecture for the second phase of the project. The various processes for collecting the information, extracting information, index and mining, querying and navigation, are discussed below.

2.1 Collecting documents

The first step of indexing is *spidering*, i.e. to collect documents from Web sites. In the earlier work with CERVO, a list of 40 sites was defined. These include *aggregator* sites, such as SourceCan, which collect information from other Web sites, *government* sites, such as FedBizOpps (Federal Business Opportunities – i.e. U.S. agencies), which maintain a central database of solicitations sent to them, or finally regional or organisation sites, which publish the tenders informally. At the moment our robot needs to be told where to find the documents: it is supplied with a URL seed or pattern, and a username/password if access is restricted. It can also fill out forms if CFTs can only be accessed through search forms.

The discovery of new sites has not been implemented yet. It will consist of following promising links from the known sites, using a model for business relationships, and the ex-

tracted information.

Eventually we would like to collect a wide range of documents: press releases, solicitation notices, awards, quarterly reports, etc. At the moment however we collect only solicitation notices and *awards* i.e. details about the winning bid to the tender. Although the solicitation notice and its award should share a common identifier, it is not always straightforward to pair them up, because they can exist on different sites, and the publisher can substitute their own identifier to the contracting authority. Different versions of the solicitation can exist (e.g. French and English). Moreover, amendments can be published.

Figure 2 (top) shows a simplified pre-solicitation notice for the office supplies of the Saskatchewan government. The notice was published on the Merx site, along with an amendment to change the date of the delivery from December 5, 2003 to January 5, 2004. The two documents share a common identifier (CFAB4).

2.2 Information extraction

The next step is the extraction of information and classification. The extracted information is put in a *CFT synthesis*, which is an XML document inspired from xCBL (Common Business Language) and OASIS UBL (Universal Business Language) [5].

Business-related documents, in particular CFTs, are typically classified according to an industry standard, for example, NAICS (North American Industry Classification System) or CPV (Common Procurement Vocabulary, for the European Union). Some CFTs are manually classified with these codes, whereas some others are not. However we note that some of the contents of the CFT is not relevant for classification because it is not indicative of the topic. For example, in figure 2, only the first sentence is deemed relevant. The second sentence states the delivery date and contractual obligation, and the last sentence provides a contact for more information. We consider this *procedural language* as noise, and try to remove it before classification by *filtering*. At a more technical level, filtering could also be simply removing HTML layout tags.

Figure 2 shows an example of a CFT synthesis, which combines information from the presolicitation notice and its amendment (represented as two `published-document` elements). It is identified by the Saskatchewan Government with number 031021-5, and by Merx with number CFAB4. The synthesis includes the free text description of the CFT (`description` element) and the various extracted elements: for example the contracting authority (Saskatchewan government), contact person (Bernie Juneau), classification code (NAICS 418210), etc.

Elements in the synthesis can be repeated if they apply more than once to the CFT. In our example there is a French and an English title because the notice on Merx was bilingual. Similarly there could be multiple classification codes because the CFT was classified under several schemas. Elements can also have an associated confidence measure, to represent the accuracy of information extraction. In our example there are two execution dates, but the amended date is given more confidence.

Presolicitation (on Merx):

Reference Number: CFAB4
Source ID: PV.MN.SA.213412
Published: 2003/10/08
Closing: 2003/10/28 02:00PM
Organisation Name: Saskatchewan Government
Title (English): Office Supplies
Title (French): Fournitures de Bureau
Description: The Government of Saskatchewan invites tenders to provide office supplies to its offices in Regina. The supplier is expected to start delivery on December 5, 2003, and enter an agreement of at least 2 years.
Contact: Bernie Juneau, (306) 321-1542

Amendment (on Merx):

Reference Number: CFAB4
Description: The start delivery date has been revised to January 5, 2004.

Synthesis (inferred in MBOI):

```
<call-for-tender>
<contracting-authority-solicitation-id>
  031021-5</...>
<title xml:lang="en">Office Supplies</title>
<title xml:lang="fr">Fournitures de Bureau</...>
<description xml:lang="en">The Government of
Saskatchewan
  invites tenders...</description>
<date-closing>2003-10-28</date-closing>
<execution-date-start confidence="0.1">
  2003-12-05</...>
<execution-date-start confidence="0.9">
  2004-01-05</...>

<classification>
  <classification-system>NAICS</...>
  <code>418210</code>
</classification>

<published-document>
  <publisher-id>Merx</publisher-id>
  <publisher-solicitation-id>CFAB4</...>
  <original-url>...</original-url>
  <cached-url>...</cached-url>
  <date-published>2003-10-08</...>
  <date-cached>2003-10-09</date-cached>
  <language>eng</language>
  <format>text/html</format>
  <document-type>presol</document-type>
</published-document>
<published-document>...
  document-type>amendment</document-type>
</published-document>
<contracting-authority>
  <name>Saskatchewan Government</name>
  <authority-type>provincial government</...>
</contracting-authority>
<contact><name>Bernie</name>
  <surname>Juneau</surname>
  <phone>(306) 321-1542</phone>
</contact>
</call-for-tender>
```

Figure 2: A Call for Tender

2.3 Indexing and mining

Indexing is the process of building an *inverted file* or index to support querying. In our case the index is partitioned into *fields*, corresponding to elements in the CFT synthesis, so that we can combine free text search over all contents, with search for specific dates, locations, classification codes, etc. The retrieval engine we are using is *lucene*. Dates and money amounts are represented in a special format to accommodate the range queries supported in *lucene*, e.g. to search for money amounts between \$50,000 and \$100,000.

Other information is also pre-computed and stored in a database (*data mining*). This information is inferred from CFT synthesis; it is used to provide statistics about CFTs or information about organisations. At the moment the inference is quite simple, but in the future it is possible to envisage technique akin to business intelligence.

2.4 Querying and navigation

There are two possible uses of *profile* in our system. One is to have the user define a recurring need (query), so that the system can send results automatically whenever CFT are found. Another is to save contextual or background information and use it transparently to affect querying. The first was implemented in CERVO, and we are currently working on the second.

Our system includes CFTs in French and English. In order to allow a searcher to find documents in another language, we have implemented translation of queries using CLIR (Cross-Language Information Retrieval) techniques [11]. To this end, we have build a statistical translation dictionary from a collection of 100,000 pairs of documents of the TED site (Tenders Electronic Daily - for the European community).

3 Test collection

It is well-known that the results of information extraction and classification techniques vary according to the domain and collection. Since no CFT collection was available, we defined our own, using pseudo-XML documents from FedBizOpps (FBO). The appeal of this collection is that some information is already tagged, and therefore can ease the evaluation.

To eliminate duplicates, we kept only one document per CFT, i.e. chose a document amongst and pre-solicitations and amendments. We only considered documents with two classification codes, FCS and NAICS¹, so that we can later measure the effectiveness of conversion. We obtained 21945 documents, covering the period from September 2000 to October 2003. Finally, we splitted this collection in two: 60% for training, and 40% for testing.

The NAICS codes are hierarchical: every digit of a six-digit code corresponds to a level of the hierarchy. For example, in figure 2, the industry code is 418120 (Recyclable Paper and Paperboard Wholesaler-Distributors) and the sector code, 418 (Miscellaneous Wholesaler-Distributors). Each of

¹Since the NAICS codes were not tagged in XML at the time (as they now are), they were extracted from the free text description. To ensure 100% precision, very strict patterns were used: e.g. "The NAICS code is XXXXXX".

the three participating countries, the U.S., Canada and Mexico, have their own version of the standard, which mostly differ at the level of industry codes (5th or 6th digit). However, there are exceptions, as demonstrated by our example: the equivalent code in the American version would be 424120 (Stationery and Office Supplies Merchant Wholesalers) and the sector code, 424 (Merchant Wholesalers, Nondurable Goods).

We reduced the category space by considering only the first three digits, i.e. the corresponding sector. This resulted in 92 categories (vs. 101 for FCS). We did not normalise for the uneven distribution of categories: for NAICS, 34% of documents are in the top two categories, and for FCS, 33% are in the top five.

As mentioned above, filtering can play an important role in the classification of CFTs. We thus built a subset of the collection in order to experiment with filtering techniques. We manually labeled 1000 sentences from 41 documents. The label was "positive" if the sentence was indicative of the tender's subject, or "negative" if not. Sentences with descriptive contents were labeled positive, while sentences about submission procedure, rules to follow, delivery dates, etc. were labeled negative. In the example of figure 2, only the first sentence would be labeled positive. Overall, almost a quarter of the sentences (243) were judged positive.

4 Information Extraction

Although CFTs do not follow a standard format, they usually contain the following information: contracting authority, opening and closing date, legal notices on the conditions of submission, etc. These pieces of information are often present but mixed up in a semi-structured text. For example the location can be tagged as "`<td>Location</td><td>Montreal</td>`", identified with "Location: Montreal", or simply appear in the description body: "This contract is to be performed in Montreal."

This information is called *named entities* [7] in the literature, and usually refers to names of people, organisations, locations, time or quantities. Their extraction proceeds from a syntactic or lexical analysis, possibly with the help of a dictionary. In our experiment we use NStein *NFinder*, which uses a combination of lexical rules and dictionary.

4.1 Filtering

In a previous study [9] we have examined the performance of named entities extraction on a small sample of 40 documents from our collection. Note surprisingly we found that money extraction, a relatively simple task, obtained the highest score, while organisation and person scored the lowest. Some of the problems we identified were: four-digit SIC (Standard Industrial Classification) codes incorrectly identified as years, acronyms erroneously identified as organisations (e.g. "frequency domain helicopter electromagnetic (HEM)"), two capitalised words identified as a person (e.g. "Space Flight", "Will Result"), missing US state abbreviations (e.g. TX, FL), etc.

This time we are interested to see if those entities can be used predict the relevance of a sentence in a call for tender. We have considered the following entities:

- geographical location. In a call for tender, this can be an execution or delivery location. A location can also be part of an address for a point of contact or the contracting authority (although these are preferably tagged as meta-data in FBO, they often appear in the text body).
- organisation. Most often the organisation will be the contracting authority or one its affiliates. For pre-determined contracts it can be the contractor.
- date. This can be a delivery date or execution date (opening and closing dates are preferably tagged as meta-data).
- time. A time limit on the delivery date, or business hours for a point of contact.
- money. The minimum/maximum contract value, or the business size of the contractor.
- URL. The Web site of the contracting authority or a regulatory site (e.g. CCR).
- person. A point of contact in the contracting agency.
- email, phone number. The details of a point of contact.

The entities above have a particular use in our collection. However they are generic in the sense that they also apply to many other domains. We have also considered the following entities, specific to our collection:

- FAR (Federal Acquisition Rules). These are tendering rules for U.S. government agencies. A call for tender may refer to an applicable paragraph in the FAR (e.g. "FAR Subpart 13.5").
- CLIN (Contract Line Item Number). The line item define a part or sub-contract of the tender. Line items usually appear as a list (e.g. "CLIN 0001: ...").
- dimensions. In the context of a tender, a dimension almost always refers to the physical characteristics of a product to deliver (e.g. "240MM x 120MM").

All entities except CLIN and dimensions are negative indicators: their presence is an indication of a negative passage or sentence, i.e. not relevant to the subject of the tender. CLIN and dimensions on the other hand are positive indicators, since they introduce details about the contract or product.

We have extracted entities using NStein *NFinder* for the top four entities, and simple regular expressions for the others. Table 1 summarises the results by entity type. The first column gives the total frequency of the entity in the FBO collection. The second column shows the accuracy of the entities as positive/negative indicators on the 1000 sentences subcollection of FBO. For example, phone number (a negative indicator) appeared in 40 sentences, 39 of which were labeled negative. Dimensions (a positive indicator) appeared in 8 sentences, all of which were labeled positive.

Locations and organisations are the most problematic entities, with very low accuracy. That is partly because they often

Table 1: Named entities in FBO documents

type	Freq. in FBO	Accuracy as a predictor
location	123344	50% (66/132)
person	48469	N/A
date & time	170525	96% (101/105)
money	30606	100% (18/18)
URL & email	29177	100% (38/38)
phone number	25938	98% (39/40)
FAR	142762	100% (56/56)
CLIN	10364	80% (4/5)
dimensions	5290	100% (8/8)

appear along with the subject in an introductory sentence. For example the first sentence in our example CFT contains an organisation (Government of Saskatchewan), the subject (office supplies) and a location (Regina).

4.2 Classification

CFTs are often classified by industry type, according to the several existing standards: SIC (Standard Industrial Classification), NAICS (North American Industry Classification System), FCS (Federal Supply Codes), CPV (Common Procurement Vocabulary), etc. The classification codes are not always included in documents, and even when they are, it is interesting to classify the same CFT according to other standards. For instance, an American user will probably be familiar with NAICS, but maybe not with CPV (the European Union standards). Furthermore, these standardss are regularly updated, and the different codes between two versions can be the source of errors. We can make these conversions explicit by classifying CFTs according to several standards: e.g. CPV, NAICS version 1997/Canada, NAICS version 2002, etc.

Table 2 shows results of the classification of NAICS codes on our FBO collection, using the title and description fields. The first result is our baseline: a Naive Bayes classifier [6] where the 8000 top terms were selected according to their InfoGain score. The following thresholds were applied: a rank cut of 1 (rcut), a fixed weight cut of 0.001 (wcut), and a category cut learnt after cross-sampling 50test set over 10 iterations (scut). More details about these thresholding techniques can be found in [14; 15]. Our results were obtained with *rainbow* [8].

The second line, “patterns”, is the classification result after replacing named entities in the collection with generic names. For example, “\$42.00” with “CU” (for currency). This will force the classifier to consider some terms which would normally not be indexed (such as numbers), and diminish the impact of some other terms (such as locations). Not surprisingly it does not affect results much, because these entities are pretty much uniformly distributed over the classes. We tried different combinations of entities, with little difference in the results. A micro-F1 score of .5466 was obtained with the most accurate entities: date, time, money, URL, email, phone number, FAR and dimensions.

We trained a Naive Bayes classifier on the 1000 sentences subcollection of FBO, for the positive and negative classes. The task seems to be relatively simple, since when we tested

Table 2: Classification on FBO

method	macro-F1	micro-F1
baseline	.3297	.5498
patterns	.3199	.5466
sent.filt.	.3223	.5918 (+7.6%)
sent.filt.patt.	.3497 (+6.1%)	.5939 (+8%)

the classifier on a 40/60 split we obtained a micro-F1 measure of 85%. We thus filtered the whole collection with this classifier, keeping only the positive sentences. The collection size went from around 600,000 sentences to 96,811. The new, filtered documents were then classified with another Naive Bayes classifier. The results, reported in the table as “sent.filt.” show a strong increase of the micro-F1 measure (+7.6%).

Finally the last line in table 2 combines sentence filtering with patterns, i.e. the sentence filtering classifier was re-trained with the patterns in the sentences. We expected a good increase since the classifier could now generalise better from its small training set, but the results were somewhat disappointing. The macro-F1 shows the strongest increase, +6.1% over the baseline, or +8% over the non-pattern filtering. This tend to indicate that the use of patterns did not change the overall measure (micro-F1), but provided more stability in the classes results (macro-F1).

4.3 Query Refinement

Our index supports precise queries over named entities. For example, the simple keyword query “bush” will return all documents where the word occurs, including documents about bush trimming and president Bush. However the query “person:Bush” will only return documents about (president) Bush. We provide an interface for query refinement, where extracted information is shown and can be added to the query. This can be used to disambiguate terms (e.g. starting with keyword “bush” and then adding person “Bush”) or to add criteria such as location or classification code.

Figure 3 shows a query and its results in our system. Here the query “snow removal” was entered. The bottom right part of the screen displays the results, in the usual manner, i.e. call for tenders are listed by order of relevance. Each document has a small excerpt (from its filtered contents) where the query keywords are highlighted, as well as some extracted information (here, the classification codes).

The boxes on the left represent information extracted from the top 100 result documents. *Concepts* are phrases extracted by Nstein *Nconcept* tool, which represent the salient ideas of the document. Organisations, locations (*lieux géographiques*) and categories are the named entities discussed above.

The user can refine her query by selecting an entity and adding it to the query with “+”. She can also require the absence of an entity (i.e. the NOT operator) with “-”. Figure 4 shows a “refined” query, where the location “Illinois” has been added to the keywords “snow removal”. More constraints can be added on the dates, awarded contract amount (*montant du contrat*), or awardee (*fournisseur*). Finally the

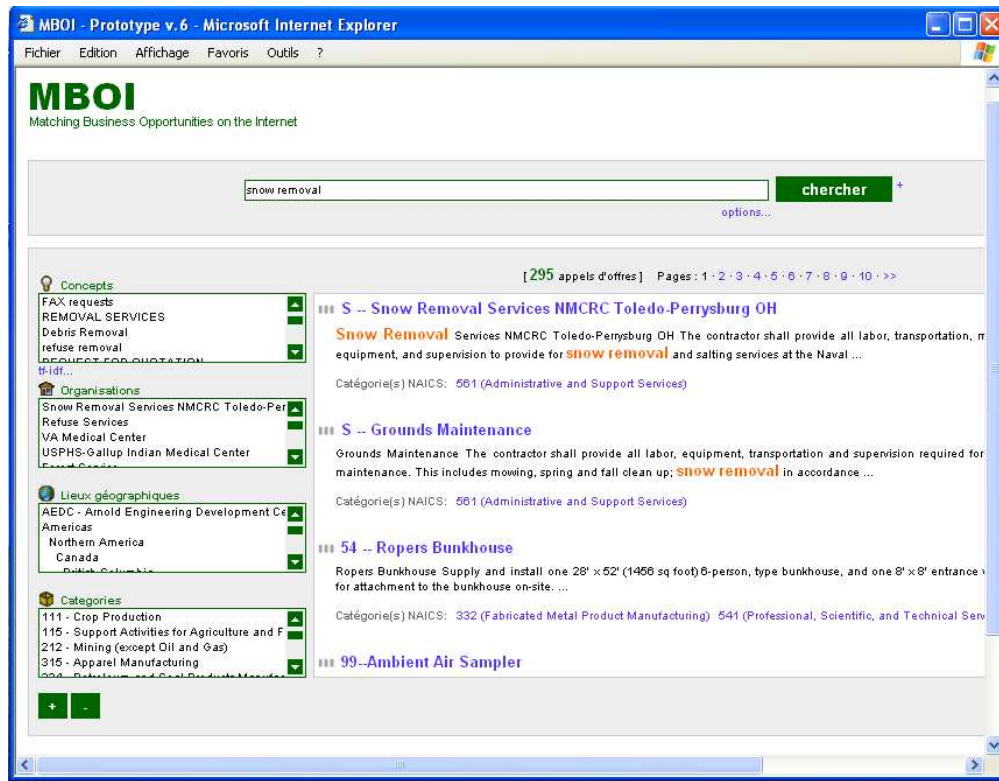


Figure 3: Querying in MBOI

Query refinement form for 'snow removal' in Illinois. The form includes fields for 'Date de publication' (with 'Du' and 'Au' date pickers), 'Montant du contrat' (with 'Entre' and 'Et' value pickers), and 'Fournisseur'. At the bottom, there are radio buttons for 'Traduction': 'Pas de traduction' (selected), 'Française', and 'Anglaise'.

Figure 4: Query Refinement

query can be translated (*traduction*) to French (*Française*) or English (*Anglaise*).

4.4 Navigation

Another aspect of our use of information extraction is to let the user browse the information space starting from the named entities.

For example figure 6 shows the top entities or “hot list” (*palmarès*) for a given period.

The first block shows the top categories (*activités*), i.e. categories that had the most number of CFTs (*appels d’offres*). Selecting a category sends a query to retrieve the associated CFTs. Similarly the classification tree can be navigated, and the relevant CFTs displayed at each node.

The second block shows the top contracting authorities (*organismes adjudicateurs*), this time based on the amount of the

Palmarès (du 2003/08/01 au 2003/08/15)		
Activités	Appels d'offres	
1. 541 - Professional, Scientific, and Technical Services	2557	
2. 334 - Computer and Electronic Product Manufacturing	2482	
3. 336 - Transportation Equipment Manufacturing	1236	
4. 561 - Administrative and Support Services	474	
5. 333 - Machinery Manufacturing	249	
Organismes adjudicateurs	Appels d'offres	Montant
1. Department of the Air Force	5	\$1067515
2. Department of the Navy	6	\$306996
3. General Services Administration	1	\$125000
4. Department of Agriculture	1	\$44219
5. Department of Energy	1	\$37740

Figure 6: Most active entities

contracts (*montant*). Clicking on a link this time brings up a company profile page, as shown in figure 5. This information is entirely built from the CFT and awards documents. The first box shows the known addresses for this organisation (here, its different branches), the categories of awarded contracts (*activités des contrats octroyés*), and business relationships (*relations d'affaires*), which in this case are the awardees to its tenders. Finally the CFTs for this organisation are listed.

Department of Agriculture

Adresses
 Department of Agriculture, Agricultural Marketing Service, Cotton Program, 3275 Appling Road, Room 1, Memphis, TN, 38193
 Department of Agriculture, Agricultural Research Service, Beltsville Area, FMDD, Bldg. 003, Rm. 320, BARC-West 10300 Baltimore Avenue, Beltsville, MD, 20705-2350
 Department of Agriculture, Animal and Plant Health Inspection Service, Administrative Services Division/Purchasing, 100 North 6TH Street Butler Square, 5TH Floor, Minneapolis, MN, 55403

Activités des contrats octroyés
 111 - Crop Production
 115 - Support Activities for Agriculture and Forestry
 212 - Mining (except Oil and Gas)
 333 - Machinery Manufacturing
 339 - Miscellaneous Manufacturing
 484 - Truck Transportation

Relations d'affaires
 Affymetrix Inc.
 CHARLES ROVER/SPAFAS
 CHESTNUT RIDGE FORESTRY Joel & Esther Fyock PO Box
 CONTRACT MANAGEMENT INC
 ENPRO INC.
 J.D. Parrella Electric Inc.

Appels d'offre en tant qu'organisme adjudicateur
 [281 appels d'offres] Pages: 1 · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · >>

B -- Design and Synthesis of 70-mer Oligonucleotide Probes
 Design and Synthesis of 70-mer Oligonucleotide Probes The USDA, Agricultural Research Service in Albany, California has a requirement for the design and synthesis of 70-mer Oligonucleotide probes for ...
 Catégorie(s) NAICS: 334 (Computer and Electronic Product Manufacturing) 541 (Professional, Scientific, and Technical Services)

Figure 5: Company profile

5 Conclusion

Our tool has been in use by our commercial partners, and deployed in several applications: as an aid for business opportunities watch for the St-Hyacinthe (Québec) region, as a CFT search facility for the Canada's metal industry portal (Net-Metal²), and as an “issue” or “thematic” watch for the Quebec travel industry.

Our study of information extraction and classification techniques show results comparable to those reported in the literature. We have shown that sentence filtering brings a strong increase to classification. However the combination of filtering and named entities seemed to bring only a small increase in the macro-F1 measure. In another work we tried using controlled vocabulary derived from the classification schema, but could not demonstrate significant gain [10] either. We believe the advantages on the interface to query refinement and collection browsing are obvious; however we would like to be able to evaluate the querying the same way we evaluated classification. To that end, we are currently working with SDTI to build a query collection.

Another way to improve classification is to combine it with information extraction. Our first experiments with conditional classification, i.e. using the conditional probabilities observed on the training set between two classification standards, have not been up to our expectations. A possible way to improve these results is to combine it with fixed conver-

sion rules that are available between some standards. We are also trying to use *concepts*, as extracted by Nstein *Nconcept* to improve classification. Preliminary results are promising. This is supported by other works [3], which have shown that term indexing could be improved with latent semantic indexing and boosting strategies.

Other directions for future work include Intelligent Web Robots [4], Multilingual Information Retrieval [11], and Filtering. Multilingual Information Retrieval can help international tenders be aware and submit for calls published in foreign languages. This is especially important considering that contracting authorities tend to favor local tenders, and therefore publish only in the local language.

Acknowledgments

This project was financed jointly by Nstein Technologies and NSERC. We would also like to thank the following people who have made this project possible: Claude Martineau, from SDTI and Robert-Gérin Lajoie from Cirano.

References

- [1] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu. Intelligent crawling on the world wide web with arbitrary predicates. In *Proceedings International WWW Conference*, 2001.
- [2] M. Betts. The future of business intelligence. *Computer World*, 14 April 2003.

²<http://www.netmetal.net/>

- [3] L. Cai and T. Hofmann. Text categorization by boosting automatically extracted concepts. In *Proceedings of the 26th annual international ACM SIGIR conference*, pages 182–189, 2003.
- [4] M. Chau, D. Zeng, H. Chen, M. Huang, and D. Hendriawan. Design and evaluation of a multi-agent collaborative web mining system. *Decision Support Systems*, 35(1):167–183, 2003.
- [5] E. Dumbill. High hopes for the universal business language. *XML.com*, O'Reilly, November 7 2001.
- [6] J. T. Jason D. M. Rennie, Lawrence Shih and D. R. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
- [7] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. Named entity recognition from diverse text types. In *Recent Advances in Natural Language Processing*, pages 257–274, 2001.
- [8] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. <http://www.cs.cmu.edu/~mccallum/bow>, 1996.
- [9] F. Paradis, Q. Ma, J.-Y. Nie, S. Vaucher, J.-F. Garneau, R. Gérin-Lajoie, and A. Tajarobi. Mboi: Un outil pour la veille d'opportunités sur l'internet. In *Colloque sur la Veille Stratégique Scientifique et Technologique*, 25–29 October 2004.
- [10] F. Paradis and J.-Y. Nie. Étude sur l'impact du sous-langage dans la classification automatique d'appels d'offres. In *CORIA*, 9–11 mars 2005.
- [11] C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors. *Advances in Cross-Language Information Retrieval Systems*. Springer, 2003.
- [12] E. F. T. K. Sang and F. D. Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147, 2003.
- [13] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 44(1), 1999.
- [14] Y. Yang. An evaluation of statistical approaches to text categorization. *Journal of Information Retrieval*, 1(1/2):67–88, 1999. An excellent reference paper for comparisons of classification algorithms on the Reuters collection.
- [15] Y. Yang. A study on thresholding strategies for text categorization. In *Proceedings of SIGIR-01, 24th ACM International Conference on Research and Development in Information Retrieval*, 2001.

Reducing Critiquing Repetition in Conversational Recommendation *

Maria Salamó, Barry Smyth, Kevin McCarthy, James Reilly, Lorraine McGinty

Adaptive Information Cluster, Smart Media Institute,

Department of Computer Science, University College Dublin (UCD), Ireland

{maria, barry.smyth, kevin.mccarthy, james.d.reilly, lorraine.mcginthy}@ucd.ie

Keywords: Case-Based Reasoning, recommender systems, critiquing, user-feedback

Abstract

Critiquing is a widely applicable and powerful form of user feedback for case-based recommender systems. Instead of providing specific value preferences, a user expresses their feature preferences by applying a series of directional critiques over recommendations. For example, a user might ask for a ‘less expensive’ vacation in a travel recommender; ‘less expensive’ is a critique over the *price* feature. Critiquing approaches constrain the feature’s value space by only considering those remaining cases that satisfy the current critique as the basis for the next recommendation cycle. In this paper we highlight a common source of inefficiency with the standard form of critiquing, which often causes users to iteratively re-select the same critique in order to refine a specific feature. This leads to longer recommendation sessions than necessary. We propose a solution for reducing these critique repetitions, resulting in improved recommendation efficiency.

1 Introduction

Recommender systems help users navigate to products of preference by combining ideas and technologies from information retrieval, artificial intelligence and user modelling, among others, to provide users with more proactive and personalized information services. Conversational, case-based recommender systems [Aha *et al.*, 2000], in particular, play an important role in e-commerce applications. They help users navigate through complex product spaces in pursuit of suitable products using a cyclical recommendation process. In each recommendation cycle, the recommendation agent will suggest new products to the user and solicits feedback in order to refine its search for an appropriate suggestion for the user. The expectation is that, on each cycle the recommender learns more about the users *soft* product preferences and subsequent cycles will prioritise products among the available options that best satisfy these. Ideally, each recommendation cycle relocates the user in a different part of the product space, closer to their target product than they

were previously. Advantages of the approach include, (1) users have more control over the navigation process [Burke *et al.*, 1997], and (2) users are guided to target products faster than standard browsing and/or alternative recommendation approaches [Burke *et al.*, 1997; McGinty and Smyth, 2002; Shimazu, 2002].

Different types of recommendation agents can be distinguished by the type of feedback that they support; examples include value elicitation, ratings-based feedback, preference-based feedback [Smyth and McGinty, 2003a]. In this paper we are especially interested in a form of user feedback called *critiquing* [McGinty and Smyth, 2003b], where a user indicates a directional feature preference in relation to a presented recommendation. For example, in a travel vacation recommender, a user might indicate that they are interested in a vacation that is *longer* than the currently recommended option; in this instance, *longer* is a critique over the *duration* feature.

Within the recommender systems literature the basic idea of critiquing can be traced back to the seminal work of Burke *et al.* [Burke *et al.*, 1995; 1997]. For example, *Entrée* is the quintessential recommender system that employs critiquing (also sometimes referred to as *tweaking*) in the restaurant domain, allowing users to critique restaurant features such as *price*, *style*, *atmosphere* etc. The advantage of critiquing is that it is a fairly lightweight form of feedback, in the sense that the user does not need to provide specific feature value, while at the same time helping the recommender to narrow its search focus quite significantly [McGinty and Smyth, 2003a]. Recently, there has been renewed interest in critiquing, as recommender systems become more commonplace, and a number of enhancements have been proposed to the basic critiquing approach. For instance, an improved approach [Reilly *et al.*, 2004] is to consider a user’s critiquing history, as well as the current critique, when making new recommendations. This approach has shown significant improvements on recommendation efficiency.

The traditional implementation of critiquing can lead to protracted recommendation sessions as there is a tendency towards relatively minor changes in the values of critiqued features from cycle to cycle, as also noticed in [Burke, 2002]. For example, our holiday-maker might have received a recommendation for a luxury 2-week package in Spain for €2000. They might be interested in something around the €1000 mark and so may indicate that they want something

*This material is based on works supported by Science Foundation Ireland under Grant No. 03/IN.31361.

cheaper. However, the next recommendation might be for a €1750 holiday and so they might again select the *cheaper* critique again and continue like this until a €1000 holiday is presented. There is little to be gained from this type of interaction and in this paper we propose a simple modification to the traditional form of critiquing that facilitates larger jumps through the product space in order to focus in more rapidly on satisfactory product cases. We further show that our approach has the potential to deliver improvements in recommendation efficiency.

2 Observations and Implications

The motivation for the work reported in this paper arose following an in-depth analysis of live-user behavior during a large-scale evaluation trial of a critiquing-based recommender during December 2004. Users included undergraduate and postgraduate students from the department of Computer Science at University College Dublin. Trial participants were invited to use our purpose built digital camera recommender system. During the trial, participants were asked to use the recommender to *shop* for one of 25 target cameras. In total 76 unique users participated (47% postgraduate, 53% undergraduate), generating a total of 1092 recommendation sessions.

Our analysis of their behavior revealed a number of interesting characteristics. First of all, we found that in a very large proportion of recommendation sessions users demonstrated *critique repetition*, whereby they continue to critique a given feature in the same way for a number of successive cycles. For example, Figure 1 shows the percentage of sessions exhibiting this type of behaviour. We see that more than 40% of sessions included repeating sequences for a single critique. Indeed only 11% of sessions contained no critique repeats at all so we can conclude that this type of repetition is commonplace.

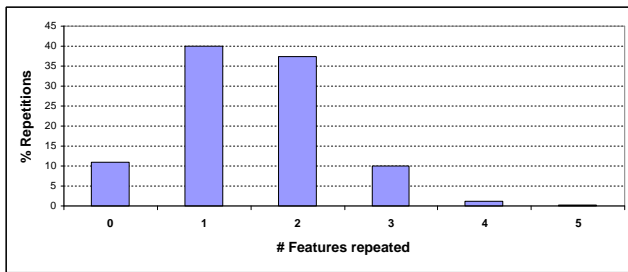


Figure 1: Recommendation sessions with sequences of repeating critiques.

Our analysis also highlights that certain critique repetitions are more frequent than others. For example, in Figure 2 we see that sequences of critiques on the *weight* feature occur in 11.82% of recommendation sessions. The *price* feature is susceptible to critique repetition in more than 18% of sessions, but other features such as *manufacturer*, *model* or *storage type* are less likely to be involved in repeating sequences.

Another important issue is the average length of critique repetitions. Figure 3 graphs the average length of the repeating sequences for each critique type. We see, for example, that the minimum repetition length is 1.59 cycles for the *storage* feature, but repeat sequences for the *manufacturer* can be more than 5 cycles long on average.

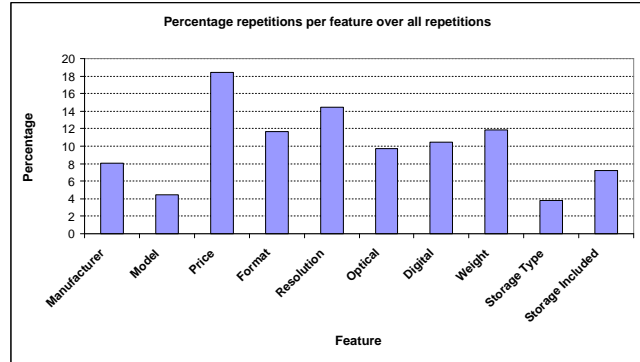


Figure 2: Percentage critiques repeated in camera dataset.

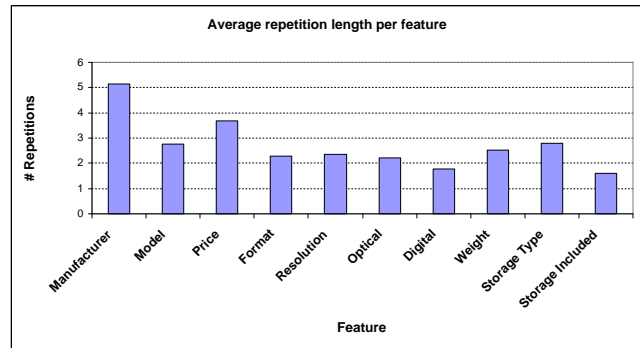


Figure 3: Repetitions per feature in camera dataset.

These observations tell us that, as predicted, users do frequently repeat the critiques they select in order to refine specific feature preferences. We believe that this is a result of the way in which critiquing is normally implemented as we shall discuss in the next section. These repeats are wasteful, from a user and system perspective, and in what follows we describe an approach to reducing their frequency by allowing the recommendation agent to make larger jumps through the product space as a result of a critique.

3 Eliminating Feedback Repetition

In this section we will describe the *standard* implementation of critiquing, as used by [Burke *et al.*, 1996]. We will argue that by its very nature there is a tendency for this form of critiquing to lead to relatively minor changes in the value of the critiqued feature from cycle to cycle and that this can lead to

the type of repetitive feedback observed in live user trials and discussed in the previous section. We will further describe an approach that attempts to eliminate, or at least minimise this feedback repetition.

3.1 Standard Critiquing

In this work we will assume a conversational recommender system in the likeness of Entrée [Burke *et al.*, 1996]. Each recommendation session starts with an initial user query and this will result in the retrieval of the most similar case available in the first recommendation cycle. The user will have the opportunity to accept this case, thereby ending the recommendation session, or to critique this case as a means to influence the next cycle. A simplified version of the basic algorithm is given in Figure 4 and consists of 3 key steps: (1) a new case c_r is recommended to the user based on the current query; (2) the user reviews the recommendation and applies a directional feature critique, cq ; (3) the query, q is revised the query for the next cycle. The recommendation process terminates either when the user is presented with a suitable case, or when they give up.

```

q: query, CB: CaseBase, cq: critique, cr : current recommendation

1.  define Standard_Critiquing(q, CB)
2.  cq:= null
3.  begin
4.  do
5.    cr ← ItemRecommend(q, CB, cq)
6.    cq ← UserReview(cr, CB)
7.    q ← QueryRevise(q, cq, cr)
8.  until UserAccepts(cr)
9.  end

10. define ItemRecommend(q, CB, cq)
11. begin
12.  CB' ← {c ∈ CB | Satisfies(c, cq)}
13.  CB' ← sort cases in CB' in decreasing order of their sim to q
14.  cr ← most similar case in CB'
15.  return cr
16. end

17. define UserReview(cr, CB)
18. begin
19.  cq ← user critique for some f ∈ cr
20.  CB ← CB - cr
21.  return cq
22. end

23. define QueryRevise(q, cr)
24. begin
25.  q ← cr
26.  return q
27. end

```

Figure 4: The standard critiquing algorithm.

Importantly, in the **ItemRecommend** step, when the user critiques a case the critique itself acts as a filter over the remaining cases (see line 12 in Figure 4), and the case chosen for the next cycle is that case which is compatible with the critique and which is maximally similar to the query, q (lines 13 and 14). In other words, once a critique has been selected, the recommender first eliminates any items that are incompatible with this critique, and then selects the next recommendation from those that remain.

The feedback repetition that we have observed tends to occur in well populated case-bases or regions of the product space, where it is likely that there will be a case that is a close match to the critiqued case within the remaining cases that are compatible with the critique. Moreover, this newly chosen case is likely to be similar to the critiqued case in terms of the critiqued feature [Burke, 2002]. And if it is selected for the next cycle then the user will be presented with a new case that offers little in the way of change compared to the previous recommendation. For example, if the user has asked for a *less expensive* camera than a €2000 recommendation then they may receive a recommendation for a €1900 camera in the next cycle; probably the user is looking for something quite a bit cheaper.

3.2 Using Value Ranges to Reduce Repetition

The standard critiquing approach is highly susceptible to feature-critique repetitions that offer only a minor change in the relevant feature value from each cycle to the next. We propose that this is largely due to the linear search policy it uses to navigate through the value space for the critiqued feature. The result is that the recommendation agent takes very *short steps* through the space of possible alternatives. In this section we describe how the standard critiquing algorithm can be easily altered to facilitate *larger jumps* through the value space for a given feature by taking a more efficient binary search approach.

```

q: query, CB: CaseBase, cq: critique, cr : current recommendation

1.  define QueryRevise(q, cq, CB, cr)
2.  begin
3.    q ← cr
4.    CB' ← {c ∈ CB | Satisfies(c, cq)}
5.    fcq ← set value in q for critiqued feature f ∈ cr by Eq. 1 or 2
6.    return q
7.  end

```

Figure 5: Adapting the standard critiquing algorithm *QueryUpdate* procedure to reduce critique repetition.

Figure 5 demonstrates how the standard algorithm can be easily extended to support our proposed approach. The only procedure which is affected is the **QueryRevise** step. Just as before the new query is updated with all of the features from the current recommendation, c_r . In addition two new steps are added. First, the recommender agent gathers all of the available cases to the current feature critique (see line 4 of Figure 5). The second step involves determining the value-change the critiqued feature will take on in the revised query, q , used for retrieval of the next recommendation. Importantly, in our approach all the remaining cases, CB' , influence the final value. There are many approaches that could be used to compute this value. In this paper we examine two possibilities - by computing the mean (see equation 1) or the median (see equation 2) for all cases in CB' .

$$f_{cq} = \frac{\sum_{\forall c} CB'_c(\text{feature } f \text{ critiqued by } cq)}{n} \quad (1)$$

For both approaches the recommender agent collects all of the alternative value possibilities for the critiqued feature from the cases covered by CB' . For instance, if the critiqued feature were [price, <, €2000] the recommender would gather all value options that were less than €2000 from the set of remaining cases (e.g., €1800, €1650, €1600, €1570, €1460, €1350, etc.). Equation 1 assigns a value for the critiqued feature $f_{cq} \in q$ by calculating the average feature value over all the relevant cases.

$$f_{cq} = \begin{cases} CB'_{n+1/2}(f \text{ of } cq) & \text{if odd \#cases} \\ \frac{CB'_{n+1/2}(f \text{ of } cq) + CB'_{(n+1/2)+1}(f \text{ of } cq)}{2} & \text{if even \#cases} \end{cases} \quad (2)$$

For Equation 2 it is assumed that the remaining case options, CB' are first sorted in ascending order. Here $CB'_i(f \text{ in } cq)$ is the feature value critiqued by cq in the i^{th} case. The median value corresponds to a cumulative percentage of 50% (i.e., 50% of the values are below the median and 50% of the values are above the median). We place the critiqued features in ascending value order and find the middle value if the number of cases is odd or find the middle pair and compute the mean value between them if we have an even number of cases.

One important point, that also needs to be considered, is previous critiques on the same feature. Lets illustrate it with an example, imagine that the user has asked in a previous cycle for a *less expensive* vacation than a €2500 recommendation and, in the current cycle, the user says that he/she prefers a *more expensive* than a €1000 vacation. In such situation in the current cycle, all the cases including those that exceed a €2500 vacation will satisfy the current critique *more expensive* than €1000. If we compute the mean or the median value to jump larger in the search space, we also include those cases rejected previously by the user. To avoid these situations, we maintain a history with all the critiques applied by the user in order to cut correctly off the search space. The history constitutes a type of *user model* [Reilly *et al.*, 2004] that reflects all the user preferences in the current session. The previous critiques stored in the user model are treated as a set of *soft constraints* [Stolze, 2000] that allow us to control the number of remaining cases that will be used to compute the mean or median values.

```

q: query, CB: CaseBase, cq: critique, cr : current recommendation
1. define QueryRevise(q, cq, CB, cr)
2. begin
3.   q ← cr
4.   CB' ← {c ∈ CB | Satisfies(c, cq)}
5.   CB' ← eliminate cases that conflict with prior critiques
6.   fcq ← set value in q for critiqued feature f ∈ cr by Eq. 1 or 2
7.   return q
8. end

```

Figure 6: Illustrating a further necessary revision to the algorithm that ensures that previous critique applications are also considered.

So, following the earlier example, we only consider to compute the mean or median of those cases that are *more expensive* than €1000 and *less expensive* than €2500. As detailed in line 5 of Figure 6, before computing the mean or the median, we check for the existence of previously applied critiques that contest the inclusion of cases in CB' , and eliminate these cases from further consideration. Put differently, we use prior critiques to decide what cases should be covered by CB' , and to ultimately set the value selection bounds for f_{cq} .

As mentioned previously, the key motivation behind our *binary search* extension to standard critiquing was to reduce critique repetition sequences, and improve recommendation efficiency by focusing more rapidly on satisfactory products for users. Figure 7 illustrates graphically why users may need to apply repeated critiques over the same feature over a period of consecutive recommendation cycles, and demonstrates how our approach can help to combat this. Let us assume that a user interfaces with a travel recommender which only has two possible features to critique *duration* and *price*, and each bullet is a case from the list of relevant cases.

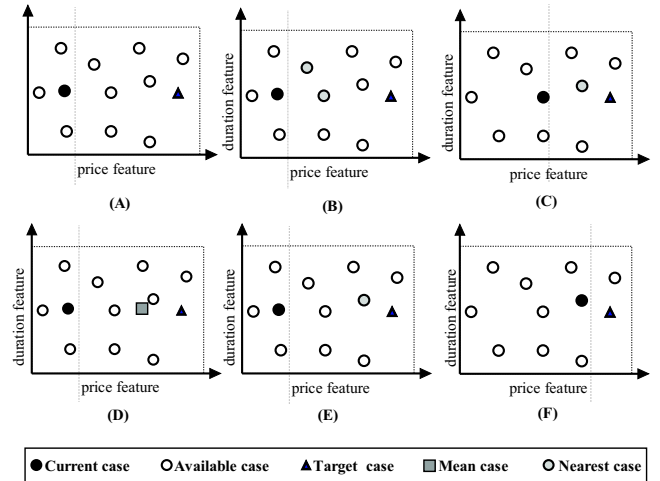


Figure 7: Graphical illustration of the operation of the standard critiquing approach (A-C), and the more efficient proposed approach (D-F).

Figure 7(A) shows the initial scenario. The user is looking for a vacation that is *more expensive* than the current recommendation. Figure 7(B) shows the behavior of a standard critiquing recommender system. The system will select one of the cases (grey bullets) that are nearest the current recommendation. However, clearly both of these are still far from the target case. This situation produces a new cycle (described in Figure 7(C)) where the previous recommended case is removed from the case base because it has not been selected, and one of the previous nearest cases is the current recommended case and a new critique *more expensive* has been generated. Again, the nearest case is not the target case and a further cycle is necessary, see Figure 7(F)).

Figure 7(D) shows the behavior of our proposed approach

given the same initial scenario as shown in Figure 7(A). We see that this approach is capable of selecting an alternative case from a different part of the options space in the next cycle. In contrast to the duration feature, in this instance the value for the price feature in the newly recommended case will not be the same as (or necessarily similar to) the previous recommendation. For this reason, the nearest case is a long way from the current recommendation, and closer to the target, saving one intermediate cycle in the process (see Figure 7(E)). The next and final cycle is shown in Figure 7(F).

In short, this binary search style approach enables the recommender to focus its search on those candidate cases that: (1) satisfy the current critique; (2) fulfill previously applied critiques; and (3) that are similar to the current case but further away from it, and thus has the capability of navigating the search space of options quickly.

4 Evaluation

In this paper so far we have argued that the traditional form of critiquing is limited by its tendency to attract sequences of the same critiques from users as they respond to incremental changes in case features during subsequent recommendation cycles. We have proposed a simple modification to this standard form of critiquing by encouraging larger changes in the values of critiqued features during recommendation; we have described one particular implementation that is inspired by the efficiency of binary search techniques compared to linear search techniques. In this section we describe the results of a preliminary evaluation that demonstrates that this new approach has the ability to lead to a reduction in average recommendation session length.

4.1 Setup

We evaluate the *mean* and *median* variations of our binary-search inspired technique, described previously, in comparison to the standard version of critiquing. The evaluation was performed using the standard Travel dataset (available from <http://www.ai-cbr.org>) which consists of 1024 vacation cases. Each case is described in terms of 9 features including *price*, *duration*, etc. The dataset was chosen because it contains numerical and nominal features and it also provides a wide search space.

4.2 Methodology

We would like to have carried out an online evaluation with live-users, but unfortunately this was not possible. As an alternative we opt for an offline evaluation similar to the one described by [Smyth and McGinty, 2003b]. Accordingly, each case (base) in the case-base is temporarily removed and used in two ways. First, it serves as a basis for a set of queries by taking random subsets of its features. We focus on subsets of 1, 3 and 5 features to allow us to distinguish between hard, moderate and easy queries respectively. Second, we select the case that is most similar to the original base. These cases are the recommendation targets for the experiments. Thus, the base represents the ideal query for a user, the generated query is the initial query provided by the ‘user’, and the target is the best available case for the user. Each generated query is a

test problem for the recommender, and in each recommendation cycle the ‘user’ picks a critique that is compatible with the known target case; that is, a critique that when applied to the remaining cases, results in the target case being left in the filtered set of cases. Each leave-one-out pass through the case-base is repeated 10 times and the recommendation sessions terminate when the target case is returned.

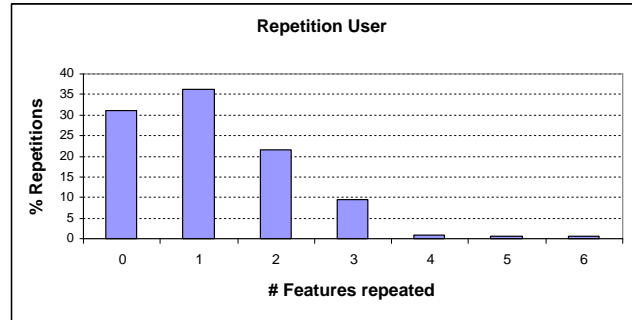


Figure 8: New repetition artificial user for our offline evaluation recommender.

One important modification to the original [Smyth and McGinty, 2003b] evaluation methodology that we implement here in response to our real-user observations relates to the frequency of repetition among selected critiques. We use a modified artificial user model that is informed by our real-user studies. The new model is designed to respond to recommendations in a manner that is more consistent to the responses observed from the real-users [McCarthy *et al.*, 2005]. In particular, our artificial user model repeats critique selections during recommendation sessions until its target feature values are met. For example, suppose our artificial user is looking for a 3-week vacation and they are presented a 3-day city-break. They are likely to ask for a *longer* vacation by critiquing the *duration* feature. Suppose that the next case recommended is for a 1-week break.

The artificial user model originally implemented by [Smyth and McGinty, 2003b] would be unlikely to re-critique the *duration* feature in this instance, even though our live-user trials indicate that this type of repetition is commonplace among real users. Our new artificial user model is designed to bias critique selection in this way, by forcing the reselection of the critique until the target feature value is satisfied (or exceeded) by the newly recommended case; we argue that this better reflects the behaviour of live-users. For example, the summary behaviour of this new type of user is presented in Figure 8 and contrasts with the corresponding behaviour of the artificial users implemented by [Smyth and McGinty, 2003b], which is presented in Figure 9. The former exhibits a critique selection pattern that is more consistent with the live-user results presented previously in Figure 1, with many sessions contain sequences of repeated critiques containing 1, 2 and 3 features. In contrast Figure 9’s results show that there is far less repetition.

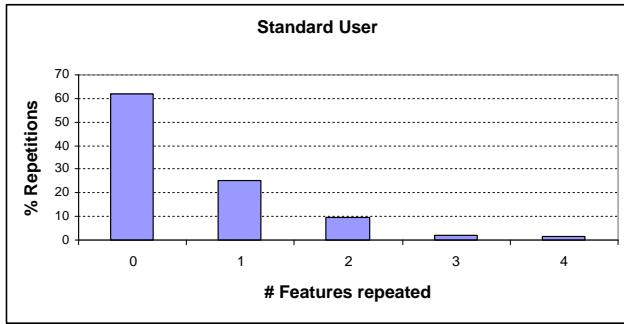


Figure 9: Standard artificial user for our offline evaluation recommender.

4.3 Recommendation Efficiency

Our main objective at this stage is to understand the degree to which our modified critiquing approach can improve basic recommendation efficiency — by which we mean average recommendation session length — when compared to the standard form of critiquing. Figure 10 presents a graph of average session length for the travel dataset, separated in recommendation sessions of different difficulty levels (as dictated by the initial query length). The results are consistent over all the queries (easy, moderate and hard) and show that our new approach to critiquing has the potential to improve basic recommendation efficiency by reducing average session length. For example, for both of our variations (*mean* and *median*) we find a relative session length reduction of between 15% and 18%, with some variation in the relative benefit due to the *mean* and *median* versions as shown in Figure 11. For instance, the *mean* version offers lower benefit in difficult and moderate queries and higher benefit in easy queries. This is to be expected perhaps since the easy queries naturally result in shorter sessions and thus there are fewer opportunities for finding good lower and upper critique bounds to focus the search space properly, and hence fewer opportunities to their benefit to be felt.

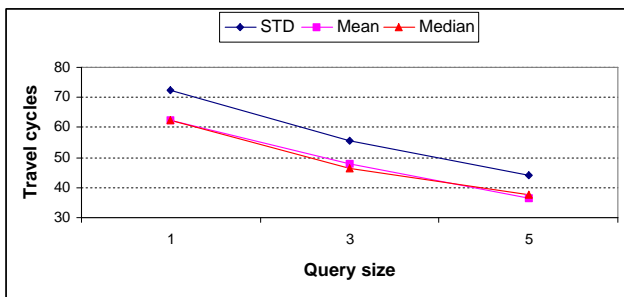


Figure 10: Efficiency results for standard critiquing, mean binary search and median binary search.

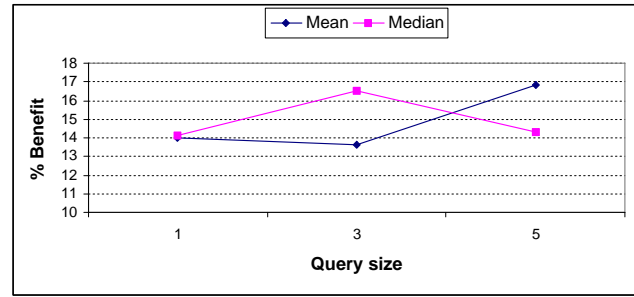


Figure 11: Benefit for mean and median binary search over standard critiquing.

5 Conclusions

Critiquing is a form of user-feedback that is ideally suited to many conversational case-based recommender scenarios. In this paper we have made some observations about how real users interact with a critiquing-based recommender. In particular, we have highlighted a tendency for repetitious critiques and traced this back to the way in which the standard form of critiquing tends to be implemented. In response we have described how this standard form of critiquing can be modified to influence the recommendation agent to produce suggestions that are less likely to attract sequences of repeat critiques from the user.

Our experiments, while preliminary and limited by the use of an artificial user model — albeit an artificial user model that has been informed by the behaviour of real users — indicate that this modified approach to critique can deliver efficiency benefits. Reductions in the average length of recommendation sessions were noted when compared to the standard form of critiquing. Our new technique is general enough to be applicable across a wide range of recommendation scenarios, especially those that assume a complex product-space where recommendation sessions are likely to be protracted.

References

- [Aha *et al.*, 2000] D.W. Aha, L.A. Breslow, and H. Muñoz-Avila. Conversational case-based reasoning. *Applied Intelligence*, 14:9–32, 2000.
- [Burke *et al.*, 1995] R. Burke, K. Hammond, and J. Kozlovsky. Knowledge-based Information Retrieval for Semi-Structured Text. In *Proceedings of the AAAI Fall Symposium on AI Applications in Knowledge Navigation and Retrieval*, pages 19–24. AAAI Press, 1995.
- [Burke *et al.*, 1996] R. Burke, K. Hammond, and B. Young. Knowledge-based navigation of complex information spaces. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 462–468. AAAI Press/MIT Press, 1996. Portland, OR.
- [Burke *et al.*, 1997] R. Burke, K. Hammond, and B.C. Young. The FindMe Approach to Assisted Browsing. *Journal of IEEE Expert*, 12(4):32–40, 1997.

- [Burke, 2002] R. Burke. Interactive Critiquing for Catalog Navigation in E-Commerce. *Artificial Intelligence Review*, 18(3-4):245–267, 2002.
- [McCarthy *et al.*, 2005] K. McCarthy, L. McGinty, B. Smyth, and J. Reilly. On the Evaluation of Dynamic Critiquing: A Large-Scale User Study. In *Proceedings Twentieth National Conference on Artificial Intelligence*, page To Appear. AAAI Press / The MIT Press, 2005.
- [McGinty and Smyth, 2002] L. McGinty and B. Smyth. Comparison-Based Recommendation. In Susan Crow, editor, *Proceedings of the 6th European Conference on Case-Based Reasoning*, pages 575–589. Springer, 2002. Aberdeen, Scotland.
- [McGinty and Smyth, 2003a] L. McGinty and B. Smyth. On the Role of Diversity in Conversational Recommender Systems. In K.D. Ashley and D.G. Bridge, editors, *Proc. 5th International Conference on Case-Based Reasoning*, pages 276–291. Springer, 2003.
- [McGinty and Smyth, 2003b] L. McGinty and B. Smyth. Tweaking Critiquing. In *Proceedings of the Workshop on Personalization and Web Techniques at the International Joint Conference on Artificial Intelligence*. Morgan-Kaufmann, 2003.
- [Reilly *et al.*, 2004] J. Reilly, K. McCarthy, L. McGinty, and B. Smyth. Incremental Critiquing. In M. Bramer, F. Coenen, and T. Allen, editors, *Research and Development in Intelligent Systems XXI. Proceedings of AI-2004*, pages 101–114. Springer, 2004. Cambridge, UK.
- [Shimazu, 2002] H. Shimazu. ExpertClerk: A conversational Case-Based Reasoning Tool for Developing Sales-clerk Agents in E-Commerce Webshops. *Artificial Intelligence*, 18(3-4):223–244, 2002.
- [Smyth and McGinty, 2003a] B. Smyth and L. McGinty. An Analysis of Feedback Strategies in Conversational Recommender Systems. In P. Cunningham, editor, *Proceedings of the 14th National Conference on Artificial Intelligence and Cognitive Science*, 2003. Dublin, Ireland.
- [Smyth and McGinty, 2003b] B. Smyth and L. McGinty. The Power of Suggestion. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Morgan-Kaufmann, 2003.
- [Stolze, 2000] M. Stolze. Soft Navigation in electronic product catalogs. *International Journal on Digital Libraries*, 3(1):60–66, 2000.

A Probabilistic Temporal Interval Algebra Based Multi-agent Scheduling System*

Elhadi Shakshuki, André Trudel, Yiqing Xu, Boya Li
Jodrey School of Computer Science
Acadia University, Wolfville, NS, Canada B4P 2R6.
{Elhadi.Shakshuki; Andre.Trudel; 072748x; 0700171}@acadiau.ca

Abstract

Recommender systems are useful for many real-world problems. In this paper, we focus on scheduling problems. An IA (Interval Algebra) network is a graph where each node represents an interval. Directed edges in the network are labelled with temporal interval relations. A probabilistic IA network has probabilities associated with the relations on the edges which can be used to capture preferences. A probabilistic IA agent (PIA-Agent) is assigned a probabilistic IA network. PIA-Agent's networks are connected via edges. We present an algorithm which allows the PIA-Agents to collaboratively solve and recommend a temporal schedule which is optimal at the agent level under the given local constraints. We conclude with an example from the university domain which involves a professor, student and secretary. The goal of the example is to demonstrate how the PIA-Agents recommend a globally consistent solution which attempts to maximize the desires of the individuals.

1 Introduction

Recommender systems alleviate the problem of users being the controllers of the system and recommend the most probable solutions from different sources based on the user's interests and preferences.

Recommender systems have been widely advocated to help make choices among recommendations from various sources [Resnick and Varian, 1997]. Although, there are several strategies and systems proposed by many researchers, it has been argued in [Herlocker et al., 2004] that there is no universally best method for all users in all situations to get the desired recommendations.

Recently, many systems have been built to help users deal with large quantities of information coming from various sources. For example in [Lieberman et al., 2001], they propose agents that assist a user browsing the World Wide Web by tracking his or her behaviour. González et al. [2005] describe an approach for the next generation multi-agent based recommender systems using smart user modeling. The main objective of their approach is to have a recommender system with a portable user model, which will interact with services in several open, distributed and heterogeneous environments to communicate user preferences in several domains. A multi-agent system for collaborative management of distributed bibliographical databases is proposed in [Karoui, et al., 2004]. In this system, each agent is responsible to help a user filling in bibliographical records, verifying the correctness of information entered and recommending relevant bibliographical references from other agents of the system. Each agent applies a case-based reasoning approach in order to provide other agents with the requested recommendations. Other researchers focused on developing systems to help teachers. For example, Yang et al. [2002] have developed a multi-agent recommender system which consists of two types of agents namely user interface and lesson plan agents. The objective of these agents is to work together to reduce teachers' overload in terms of lesson plan editing from the Web. In this approach the course objects are represented as graph and a sharable content object reference model is adopted. Alternatively, researchers in [Wei et al., 2004] proposed agents for a market-based recommender system. In this system, agents are able to bid and recommend items in order and allow the user to give ratings based on his/her satisfaction.

This paper proposes a multi-agent scheduling recommender system that combines agents and constraint satisfaction problems (CSPs). In this work, we are only concerned with a special type of CSP called an Interval Algebra network (IA network).

1.1 Temporal Reasoning

Allen [Allen, 1983] defines a temporal reasoning approach based on intervals and the 13 possible binary relations

* This project is supported in part by grants from NSERC (the Natural Sciences and Engineering Research Council of Canada) and a grant from the Research and Graduate Office at Acadia University.

between them (see Table 1). The relations are before (b), meets (m), overlaps (o), during (d), starts (s), finishes (f), and equals (=). Each relation has an inverse. The inverse symbol for b is bi and similarly for the others: mi, oi, di, si, and fi. The inverse of equals is equals. A relation between two intervals is restricted to a disjunction of the basic relations, which is represented as a set. For example, $(A \text{ m } B) \vee (A \text{ o } B)$ is written as $A \{m,o\} B$. The relation between two intervals is allowed to be any subset of $I = \{b,bi,m,mi,o,oi,d,di,s,si,f,fi,=\}$ including I itself.

An IA (Interval Algebra) network is a graph where each node represents an interval. Directed edges in the network are labelled with subsets of I. By convention, edges labeled with I are not shown. An IA network is consistent (or satisfiable) if each interval in the network can be mapped to a real interval such that all the constraints on the edges hold (i.e., one disjunct on each edge is true).

An IA network is a binary constraint satisfaction problem (CSP) with infinite domains. The intervals are the variables. The domain of each variable is the set of pairs of reals of the form (x,y) where $x < y$. The constraint between two variables i and j is the label on the edge (i,j) in the IA network.

A probabilistic IA network is an IA network with probabilities associated with each interval relation. For example, if we prefer to read the newspaper during breakfast instead of before, we have $\{d(0.9), b(0.1)\}$ “breakfast”. Directed edges in the network are labeled with subsets of I, and each relation in the subset is assigned a probability. The probabilities on an edge sum to 1. By convention, we list the labels in a set by decreasing order of probability. A probabilistic IA network is consistent (or satisfiable) if one disjunct on each edge is true, and the product of the single disjunct on each edge which is true is maximized.

A *PIA-Agent* is an agent which has ownership and control over a probabilistic IA network. A node in one *PIA-Agent*’s probabilistic IA network can be connected by an edge to a node in another *PIA-Agent*’s network. The individual *PIA-Agent* networks, along with their interconnecting edges, is called a *PIA-Agent network*. For example, Figure 1 is a 3 *PIA-Agent* network: Professor, Student, and Secretary. Each node in the network represents a real world event in the *PIA-Agent*’s daily life.

The probabilistic IA network completely contained within a *PIA-Agent* is called its *internal network*. A *PIA-Agent*’s internal network along with the edges that connect the internal network to other networks is called its *external network*. Note that the external network contains the nodes at both ends of the edges that connect the internal network to other networks. For example, the Professor’s internal network in Figure 1 is shown in Figure 2, and its external network is shown in Figure 3. Note that some inter-agent edges have two label sets associated with them. For example, the edge from node B5 to A5 in Figure 1 has the sets $\{bi(0.8), d(0.2)\}$ and $\{d(0.7), bi(0.3)\}$ (the sets do not necessarily have to contain the same labels). The reason for the dual sets is that the former is associated with the

Professor and the latter with the Student (i.e., the Professor prefers that questions be asked after class, while the Student prefers to ask questions during class). When both agents have the same set of labels, we only write it once (e.g., edge B2, A4). Internal network edges will always have only one set of labels.

Relation	Symbol	Example
X before Y	b	XXX YYY
X meets Y	m	XXX YYY
X overlaps Y	o	XXXX YYYY
X during Y	d	XXX YYYYYYY
X starts Y	s	XXX YYYYYYY
X finishes Y	f	XXX YYYYYYY
X equals Y	=	XXX YYY

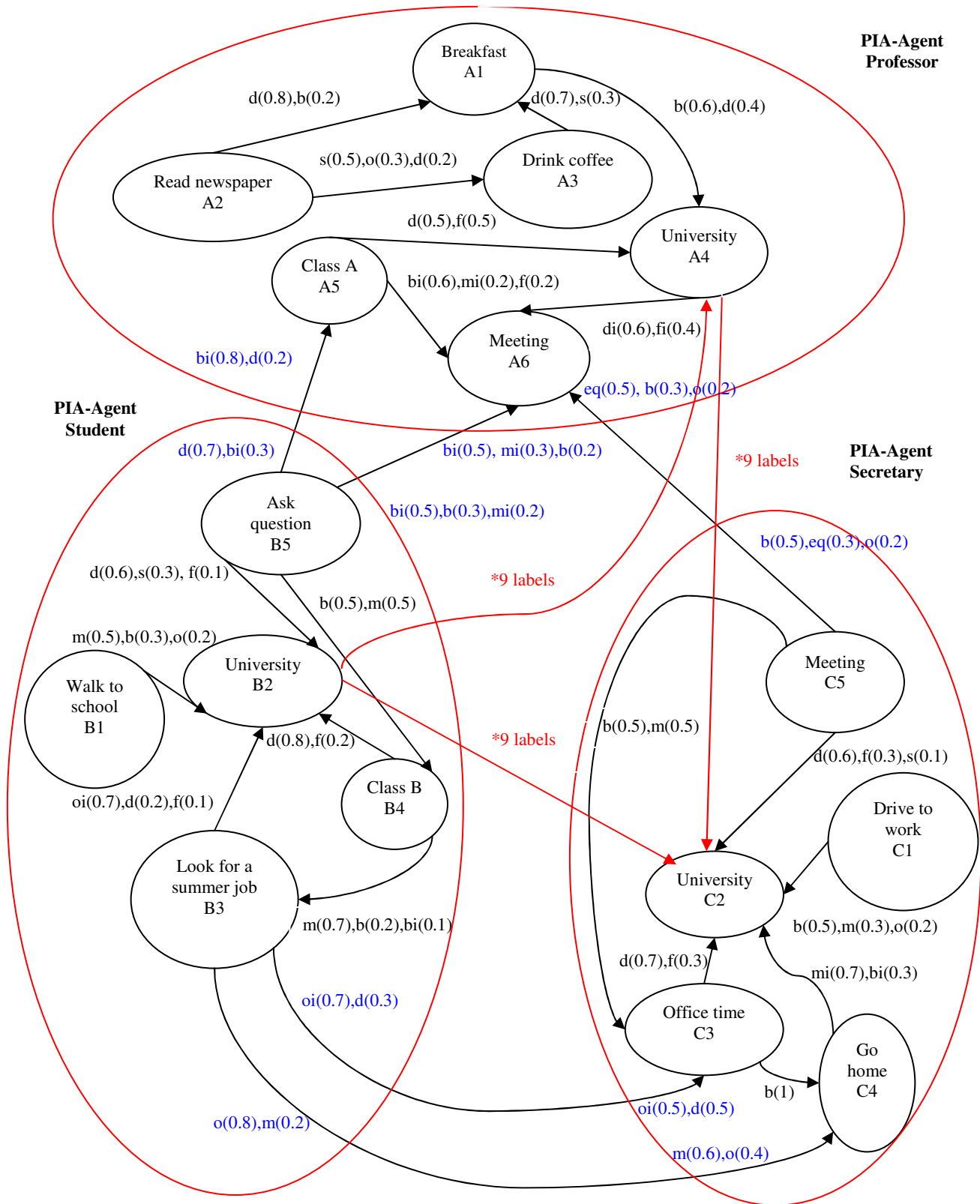
Table 1: Allen’s interval relations.

A *PIA-Agent* has complete control and knowledge of its internal network. For example in Figure 1, only the Professor can make its internal network consistent. The student and secretary do not know the structure of the Professor’s internal network and cannot change any of its labels. An edge between two *PIA-Agents* is shared by the agents. For example, the edge from node B5 to A5 in Figure 1 is shared by the Student and Professor. They both locally store a personal copy of what they consider to be the label set on the edge. The Professor stores $\{bi(0.8),d(0.2)\}$ and the Student $\{d(0.7),bi(0.3)\}$. The Professor cannot view or modify the label set stored by the Student, and vice versa.

Each *PIA-Agent* can communicate with every other *PIA-Agent*.

The problem we consider in this paper has a *PIA-Agent* network as input. The output is a consistent network where the product of the probabilities on the unique label assigned to each edge is near optimal. Our proposed algorithm makes each *PIA-Agent* external network locally optimal subject to the constraints imposed by its neighbouring *PIA-Agents*. We cannot guarantee global optimality. For example, a solution to Figure 1 is shown in Figure 4.

Note that we are not solving a typical IA network as in [van Beek & Manchak, 1996]. We are dealing with probabilities, and we cannot use an algorithm which has global access to the network. Instead, *PIA-Agents* make local changes to the network, and collaborate to guarantee global consistency.



*9 labels: eq(0.2),o(0.1), oi(0.1), s(0.1), si(0.1), d(0.1), di(0.1), f(0.1), fi(0.1)

Figure 1: Professor-Student-Secretary PIA-Agent Network.

Our PIA-Agent system architecture has PIA-Agents that are adaptive, autonomous, goal driven, dynamic, collaborative, and act as mediators between the user(s) and problem domain [Shakshuki et al., 2003]. Upon our proposed algorithm's completion, the PIA-Agents are in a position to recommend a temporal schedule that based on the negotiation and constraints imposed by the other PIA-Agents, is locally optimal.

In the remainder of the paper, we present our algorithm and then a specific example from a simplified university domain.

2 Problem solving component

Our system's overall architecture is described in [Shakshuki et al., 2005]. In this paper, we modify the problem solving component to deal with probabilities. This component consists primarily of an algorithm used by the PIA-Agent to solve its external network. Before presenting the probabilistic version of this algorithm, we describe in the following sub-section the pre-processing required by the algorithm.

2.1 Pre-processing

In addition to its internal network, each PIA-Agent must store information to be used by the algorithm. Each PIA-Agent is assigned a unique integer called its *ID*. *ID* is used by the algorithm to control the execution order and backtracking. If there are n agents, then $1 \leq ID \leq n$. The *IDs* are assigned as follows. Choose one agent randomly to be assigned $ID=1$; this will be the first PIA-Agent to execute. For example, choose the Professor in Figure 1. Instead of randomly choosing the first PIA-Agent, we could pick the PIA-Agent with the largest internal network, largest number of edges going to other PIA-Agents, or the largest total number of labels. Future work will involve experimenting with these heuristics. We next construct a connected graph G from the original PIA-Agent network. Each internal network collapses to a single node, and multiple directed edges between two internal networks collapse to a single undirected edge. For example, the connected graph G for Figure 1 is shown in Figure 5 (A = Professor, B = Student, C = Secretary).

In Figure 5, the node in the center is assigned $ID=1$. Perform a breadth first search starting at this node, and assign an increasing *ID* to each IA-Agent as it is visited during the traversal. For example, B (student) has $ID=2$, and C (Secretary) has $ID=3$. Note that the breadth first traversal guarantees that if a PIA-Agent's *ID* is not 1, then it is connected to at least one other IA-Agent with a smaller *ID*.

After the *IDs* are assigned, pairs of agents that share one or more edges need to negotiate to agree on a unique label set for each shared edge. For example, the Professor and Student in Figure 1 each store a different label set for edge B5, A5.

There are many approaches that can be taken to reconcile the label sets:

- Randomly chose one over the other.

- Assign priorities to the PIA-Agents and chose the label set associated with the highest ranked agent. A tie breaking rule may be needed.
- Apply the principle of indifference and take the average of the individual labels.

For our example in Figure 1, we give the Professor priority over the other two. The Student and Secretary have equal priority and use the principle of indifference. If a shared edge has a unique label set, it is not changed. After negotiating, the shared label sets are:

- Edge (B5,A5): $bi(0.8),d(0.2)$
- Edge (B5,A6): $bi(0.5),mi(0.3),b(0.2)$
- Edge (B2,A4): 9 labels
- Edge (A4,C2): 9 labels
- Edge (C5,A6): $eq(0.5),b(0.3),o(0.2)$
- Edge (B2,C2): 9 labels
- Edge (B3,C3): $oi(0.6),d(0.4)$
- Edge (B3,C4): $o(0.6),m(0.4)$

Each PIA-Agent stores a local copy of the label set assigned to a shared edge.

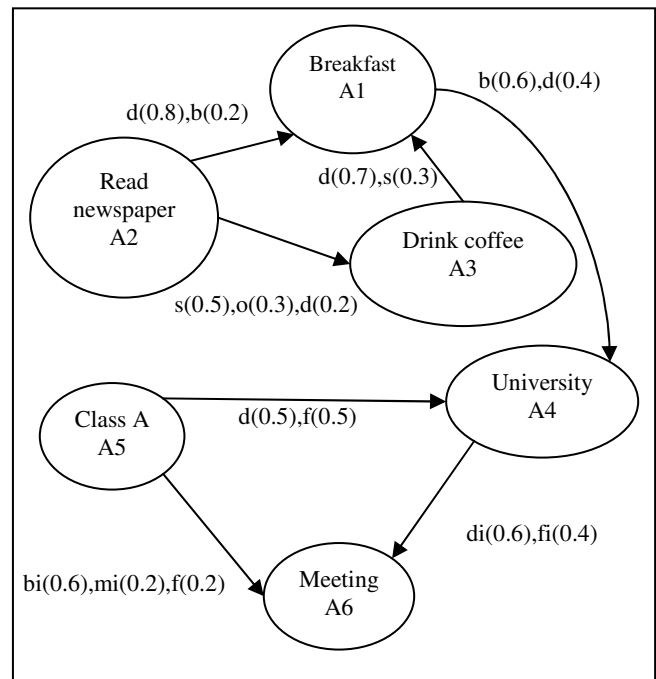


Figure 2: Professor's internal network.

Edges may need to be added to the PIA-Agent network. For every pair of PIA-Agents K and J : let N_J be the set of nodes in J that are connected to a node in K . For every node y in K that is connected to a node in J , we need to add edges so that y is connected to every node in N_J . The added edges have the label "I". Each label in the set "I" is assigned a probability of $1/13$. Note that the added edges do not affect the final solution. For example, between the Professor and

Student in Figure 1, we must add the following edges: (B2,A5), (B2,A6), and (A4,B5). It does not matter which direction the arrow points.

If there is an edge between PIA-Agents i and j , then both agents store a copy of the set of labels assigned to the edge between them. Also, we must keep track of the current singleton label from the set that is assigned to the edge. We call this singleton label: *label*. Initially, *label* is set to the first label. For example, if after negotiation the labels on an edge between agents i and j is $\{bi(0.8),mi(0.2)\}$, then both agents store a local copy of *label* = $bi(0.8)$. If i changes its value for *label*, it does not affect j 's copy. The agents must communicate and negotiate to synchronize their local copies of *label*.

During algorithm execution, if a PIA-Agent discovers a labelling on its edges that are connected to other PIA-Agents that cannot lead to a consistent external network, this labelling is stored in a *bad* set of the agent self model. Initially *bad* is empty. Note that no edges from a PIA-Agent's internal network are stored in its *bad* set.

Also during algorithm execution, whenever a PIA-Agent finds a consistent labelling for its external network, the labelling is stored in the set *solutions*. This set is initially empty and can potentially allow old solutions to be re-used during backtracking. The first element of *solutions* always contains the PIA-Agent's most recent solution.

The initial information stored by each PIA-Agent in Figure 1 is shown in Figure 6. After negotiation, the

changes to the information in Figure 6 are shown in Figure 7. Note that due to space limitations, we only show the changed information and omit each PIA-Agent's internal network representation.

2.2 Algorithm

The PIA-Agent problem solver is based on our proposed algorithm which is described in Figure 8. The algorithm is explained through examples in the next section. Notes on line 2 of the proposed algorithm:

- If the PIA-Agent discovers combinations of labels on the edges that connect it to other agents that cannot lead to a solution, these combinations recorded in its local *bad* set.
- The PIA-Agent does not change the labels on edges connected to another PIA-Agent without first receiving an approval message from the other agent. When the other agent receives the approval request, it first checks its *bad* set. If the change does not violate its *bad* set, it updates its edge labels and returns an approval message. Otherwise, a message rejecting the change is returned.
- To minimize the impact on other PIA-Agents, the currently executing PIA-Agent should first try to change labels on its internal network before attempting to modify the labels on edges connected to other PIA-Agents.
- Any algorithm for solving an IA network can be used.

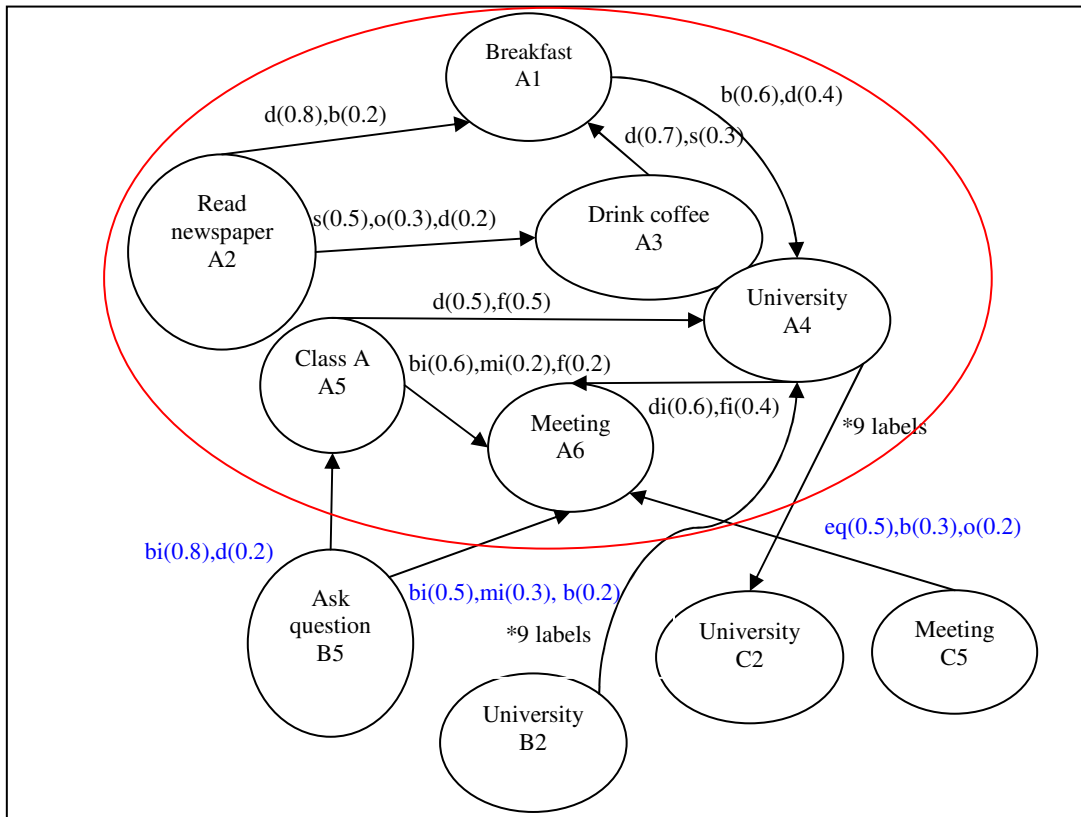


Figure 3: Professor's external network.

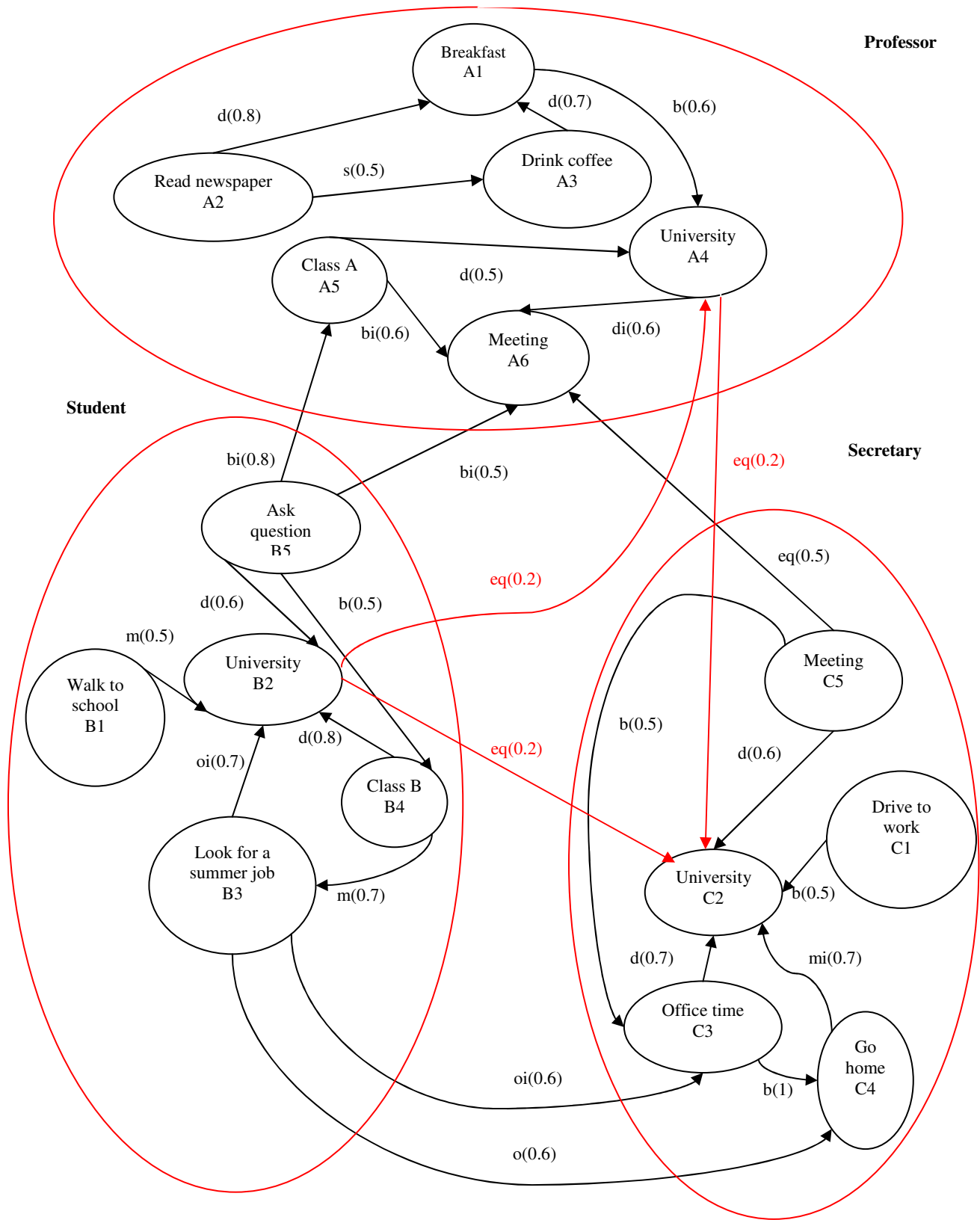


Figure 4: Global solution.

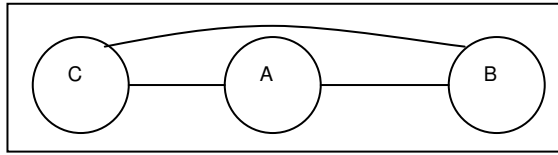


Figure 5: Connected graph G.

```

ID=1 // Professor
Edge (B5,A5): {bi(0.8),d(0.2)}, label=bi(0.8)
Edge(B5,A6):{bi(0.5), mi(0.3),b(0.2)},
label=bi(0.5)
Edge(B2,A4):{eq(0.2),o(0.1),oi(0.1),s(0.1),
si(0.1),d(0.1),di(0.1),f(0.1),fi(0.1)},
label=eq(0.2)
Edge(A4,C2):{eq(0.2),o(0.1),oi(0.1),s(0.1),
si(0.1),d(0.1),di(0.1),f(0.1),fi(0.1)},
label=eq(0.2)
Edge(C5,A6):{eq(0.5), b(0.3),o(0.2)},
label=eq(0.5)
Edge(B2,A5):I, label=di(1/13) //added edge
Edge(A4,B5):I, label=di(1/13) //added edge
Edge(B2,A6):I, label=di(1/13) //added edge
Edge(A6,C2):I, label=d(1/13) //added edge
Edge(A4,C5):I, label=di(1/13) //added edge
bad = ∅
solutions = ∅

ID=2 // Student
Edge (B5,A5): {d(0.7),bi(0.3)}, label=d(0.7)
Edge(B5,A6):{bi(0.5),b(0.3),mi(0.2)}, label=bi(0.5)
Edge(B2,A4):{eq(0.2),o(0.1),oi(0.1),s(0.1),
si(0.1),d(0.1),di(0.1),f(0.1),fi(0.1)}, label=eq(0.2)
Edge(B2,C2):{eq(0.2),o(0.1),oi(0.1),s(0.1),
si(0.1),d(0.1),di(0.1),f(0.1),fi(0.1)},
label=eq(0.2)
Edge (B3,C3): {oi(0.7),d(0.3)}, label=oi(0.7)
Edge (B3,C4): {o(0.8),m(0.2)}, label=o(0.8)
Edge (B2,A5):I, label=di(1/13) //added edge
Edge (B2,A6):I, label=di(1/13) //added edge
Edge (A4,B5):I, label=di(1/13) //added edge
Edge (B2,C3): I, label=eq(1/13) //added edge
Edge (B2,C4): I, label=m(1/13) //added edge
Edge (B3,C2): I, label=oi(1/13) //added edge
bad = ∅
solutions = ∅

ID=3 // Secretary
Edge(A4,C2):{eq(0.2),o(0.1),oi(0.1),s(0.1),
si(0.1),d(0.1),di(0.1),f(0.1),fi(0.1)},
label=eq(0.2)
Edge(C5,A6):{b(0.5),eq(0.3),o(0.2)}, label=b(0.5)
Edge(B2,C2):{eq(0.2),o(0.1),oi(0.1),s(0.1),
si(0.1),d(0.1),di(0.1),f(0.1),fi(0.1)},
label=eq(0.2)
Edge (B3,C3): {oi(0.5),d(0.5)}, label=oi(0.5)
Edge (B3,C4): {m(0.6), o(0.4)}, label=m(0.6)
Edge (A4,C5):I, label=di(1/13) //added edge
Edge (A6,C2):I, label=d(1/13) //added edge
Edge (B2,C3):I, label=eq(1/13) //added edge
Edge (B2,C4):I, label=m(1/13) //added edge
Edge (B3,C2):I, label=oi(1/13) //added edge
bad = ∅
solutions = ∅

```

Figure 6: Initial information

```

ID=1 // Professor
No change.

ID=2 // Student
Edge (B5,A5): {bi(0.8),d(0.2)}, label=bi(0.8)
Edge(B5,A6):{bi(0.5),mi(0.3),b(0.2)},
label=bi(0.5)
Edge (B3,C3): {oi(0.6),d(0.4)}, label=oi(0.6)
Edge (B3,C4): {o(0.6),m(0.4)}, label=o(0.6)

ID=3 // Secretary
Edge(C5,A6):{eq(0.5),b(0.3),o(0.2)},label=eq(0.5)
Edge (B3,C3): {oi(0.6),d(0.4)}, label=oi(0.6)
Edge (B3,C4): {o(0.6),m(0.4)}, label=o(0.6)

```

Figure 7: Status of the information after negotiation

```

1. current = 1
2. PIA-Agent ID=current solves its external network.
3. If a solution is found:
4.   | If labels on edges between current and other PIA-Agents
5.   | were modified:
6.   |   | Set current to be the minimum of (current + 1) and
7.   |   | the ID's of the PIA-Agents whose edge labels have
8.   |   | been changed.
9.   |   | Go to 2.
10.  | Else
11.  |   | If current=n then terminate in success.
12.  |   | current = current + 1
13.  |   | Go to 2.
14. Else terminate in failure.

```

Figure 8: Algorithm.

3 Example

We use the algorithm in Figure 8 to solve the network in Figure 1. Due to space limitations, we do not deal with the situation when there is no solution. Assume the PIA-Agents have finished the pre-processing and negotiation phases of the algorithm and their stored information is as shown in Figure 7. The Professor executes line 2 of the algorithm in Figure 8 and finds a solution to its external network:

```

solutions={{(B5,A5)=bi(0.8), (B5,A6)=bi(0.5), (B2,A4)=eq(0.2),
(A4,C2)=eq(0.2), (C5,A6)=eq(0.5), (B2,A5)=di(1/13),
(A4,B5)=di(1/13), (B2,A6)=di(1/13), (A6,C2)=d(1/13),
(A4,C5)=di(1/13), (A2,A1)=d(0.8), (A2,A3)=s(0.5), (A3,A1)=d(0.7),
(A1,A4)=b(0.6), (A5,A4)=d(0.5), (A5,A6)=bi(0.6), (A4,A6)=di(0.6)}}

```

Since the Professor did not change labels on edges to other PIA-Agents, lines 11-13 in Figure 8 sets $current=2$ (i.e., the Student). The Student then solves its external network:

```

solutions={{(B5,A5)=bi(0.8), (B5,A6)=bi(0.5), (B2,A4)=eq(0.2),
(B2,C2)=eq(0.2), (B3,C3)=oi(0.6), (B3,C4)=o(0.6), (B2,A5)=di(1/13),
(B2,A6)=di(1/13), (A4,B5)=di(1/13), (B2,C3)=eq(1/13),
(B2,C4)=m(1/13), (B3,C2)=oi(1/13), (B5,B2)=d(0.6), (B5,B4)=b(0.5),
(B4,B2)=d(0.8), (B1,B2)=m(0.5), (B3,B2)=oi(0.7),
(B4,B3)=m(0.7)}}

```

Fortunately, since the Student also does not change labels on the edges to the Professor and Secretary, lines 11-13 in Figure 8 sets $current=3$ (i.e., the Secretary). It is then the Secretary's turn to find a solution (i.e., line 2). The current labelling on the edges between the Student and Secretary cannot lead to a solution and the Secretary records this combination in its bad set. The Secretary then solves its

external network and the solution involves changing labels on the edges between the Student and itself. Since the Student's *bad* set is currently empty, the Student approves the changes. The updated information stored in each agent is shown in Figure 9.

Since the Secretary changed labels on edges between itself and the Student, lines 6-8 of Figure 8 chooses the minimum of 2 and 4 which is 2 (i.e., the Student). We then go to line 2 and the Student attempts to find a solution. The Student easily finds a solution which does not involve changing labels on edges to other PIA-Agents. The Student's updated solution set becomes:

```
solutions={ { (B5,A5)=bi(0.8), (B5,A6)=bi(0.5), (B2,A4)=eq(0.2),
(B2,C2)=eq(0.2), (B3,C3)=oi(0.6), (B3,C4)=o(0.6), (B2,A5)=di(1/13),
(B2,A6)=di(1/13), (A4,B5)=di(1/13), (B2,C3)=di(1/13),
(B2,C4)=m(1/13), (B3,C2)=oi(1/13), (B5,B2)=d(0.6), (B5,B4)=b(0.5),
(B4,B2)=d(0.8), (B1,B2)=m(0.5), (B3,B2)=oi(0.7), (B4,B3)=m(0.7) },
{ (B5,A5)=bi(0.8), (B5,A6)=oi(0.5), (B2,A4)=eq(0.2), (B2,C2)=eq(0.2),
(B3,C3)=oi(0.6), (B3,C4)=o(0.6), (B2,A5)=di(1/13), (B2,A6)=di(1/13),
(A4,B5)=di(1/13), (B2,C3)=eq(1/13), (B2,C4)=m(1/13),
(B3,C2)=oi(1/13), (B5,B2)=d(0.6), (B5,B4)=b(0.5), (B4,B2)=d(0.8),
(B1,B2)=m(0.5), (B3,B2)=oi(0.7), (B4,B3)=m(0.7) } }.
```

It is then the Secretary's turn to find a solution. Since the Secretary's current solution is still valid and no labels have been modified, the algorithm terminates in success. The final solution is shown in Figure 4.

```
ID=2 // Student
Edge (B2,C3): I, label=di(1/13) //added edge
ID=3 // Secretary
Edge (B2,C3):I, label=di(1/13) //added edge
bad={{(A4,C2)=eq(0.2), (C5,A6)=eq(0.5),(B2,C2)=eq(0.2),
(B3,C3)=oi(0.6), (B3,C4)=o(0.6), (A4,C5)=di(1/13), (A6,C2)=
d(1/13), (B2,C3)=eq(1/13), (B2,C4)=m(1/13), (B3,C2)=oi(1/13)}}
solutions={{(A4,C2)=eq(0.2), (C5,A6)=eq(0.5), (B2,C2)=eq(0.2),
(B3,C3)=oi(0.6), (B3,C4)=o(0.6), (A4,C5)=di(1/13), (A6,C2)=
d(1/13), (B2,C3)=di(1/13), (B2,C4)=m(1/13), (B3,C2)=oi(1/13),
(C1,C2)=b(0.5), (C3,C2)=d(0.7), (C3,C4)=b(1), (C4,C2)=mi(0.7),
(C5,C2)=d(0.6), (C5,C3)=b(0.5)}}
```

Figure 9: Status after the Secretary finds a solution.

4 Conclusion and Future Work

This paper reports on ongoing research to develop a recommender system that combines agents and probabilistic Interval Algebra networks (IA networks), which are a special type of CSP. We presented an algorithm for solving a PIA-Agent network. Our proposed algorithm in turn applies a probabilistic IA network algorithm on sub-graphs of the network. Our approach is independent of the particular IA network algorithm chosen. Agents are used to resolve conflicts between sub-graphs to find a global solution to the network. A prototype of the proposed PIA-Agent architecture is under development.

Our future work involves the following: (1) completing the implementation of the prototype so that we can carry out more extensive experiments with real users. A description of the preliminary implementation appears in [Shakshuki et al., 2005]. (2) Proving the correctness of our proposed algorithm. (3) Incorporating a learning component in the proposed system using machine learning techniques.

(4) Exploring and developing different negotiation strategies, and finally (5) investigating the possibility of introducing concurrency in our algorithm. For example, while idle an agent can search for and store alternative solutions which may be useful in the future.

References

- [Allen, 1983] Allen, J.F., Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, vol. 26, pages 832-843, 1983.
- [González, et al., 2005] González, G., López, B., Rosa, J. A Multi-agent Smart User Model for Cross-domain Recommender Systems. Workshop on the Next Stage of Recommender Systems Research, San Diego, USA, January 2005.
- [Herlocker et al., 2004] Herlocker, J., Konstan, J., Terveen, L., Riedl, J. Evaluating Collaborative Filtering Recommender Systems. *ACM Transaction Information Systems*, pages 5–53, 2004.
- [Karoui, et al., 2004] Karoui, H., Kanawati, R., Petrucci, L. An Intelligent Peer-to-Peer Multi-Agent System for Collaborative Management of Bibliographic Databases. UK CBR workshop, December 2004.
- [Lieberman et al., 2001] Lieberman, H., Christopher, F., Louis, W. Exploring the Web with Reconnaissance Agents, *Communications of the ACM*, pages 69-75, 2001.
- [Resnick and Varian, 1997] Resnick, P. and Varian, H.R. Recommender Systems. *Communications of the ACM*, 40, pages 56–58, 1997.
- [Shakshuki et al., 2003] Shakshuki, E., Ghenniwa, H. and Kamel, M. Agent-based System Architecture for Dynamic and Open Environments. *International Journal of Information Technology and Decision Making*, vol. 2, no. 1, pages 105-133, 2003.
- [Shakshuki et al., 2005] E. Shakshuki, A. Trudel, Y. Xu, and B. Li. A distributed time management system based on agents, Allen's interval algebra, and preferences. Submitted to the *International Journal of IT and Web Engineering (IJITWE)*, 2005.
- [van Beek and Manchak, 1996] van Beek, P. and Manchak, D., The Design and Experimental Analysis of Algorithms for Temporal Reasoning. *Journal of Artificial Intelligence Research*, pages 1-18, 1996.
- [Wei et al., 2004] Wei, Y., Moreau, L. and Jennings, N. Market-Based Recommender Systems: Learning Users' Interests by Quality Classification". 6th International Bi-Conference Workshop on Agent-Oriented Information Systems, pages 119-133, New York, US, July 2004.
- [Yang, et al., 2002] Yang, J. Lin, T., Wu, K. An Agent-Based Recommender System for Lesson Plan Sequencing. IEEE International Conference on Advanced Learning Technologies (ICALT 2002), Kazan, Russia, September 2002.

Feature combination in a recommender system using distributed items: The case of JukeBlog

Hugues Tremblay-Beaumont Esma Aïmeur

University of Montréal

Department of Computer Science and Operations Research

{tremlhu, aimeur}@iro.umontreal.ca

Abstract

Weblogs are very popular Internet tools. There exist weblogs on a variety of subjects, such as politics, technology and culture. Music weblogs made their apparition in 2003 and have, since then, rapidly gained in popularity. Music weblogs are like traditional weblogs, except that a song is attached to each post. There exist so many music weblogs that it is difficult for their readers to discover weblogs and posts that match their tastes. In order to help weblog readers find new interesting weblogs and posts, we introduce JukeBlog, a music weblog recommender system that uses a Feature Combination hybrid recommendation technique, combining collaborative and content based filtering.

1 Introduction

The popularity of weblogs has increased dramatically over the last years. Technorati [URL 1] references over 9 million weblogs. Lately, we have witnessed the emergence of variants on the weblog theme. Although there is marked interest in the community for traditional weblogs (political, cultural, technological, etc.), there is increasing interest for new themes, such as photography, music and video weblogs. A music weblog, also known as MP3 blog, follows all the characteristics of a normal weblog, and adds an extra feature: posts must contain a musical file (most of the time, in the MPEG Layer-3 format). Some of the most popular music weblogs include *Teaching The Indie Kids To Dance Again* [URL 2], *Fluxblog* [URL 3] and *The Tofu Hut* [URL 4]. The purpose of music weblogs is *not* to release copyrighted songs on the Internet, but to promote new emerging artists and unknown or forgotten songs. A music weblog has a characteristic that distinguishes it from the majority of other weblog types: it is linked to a certain art form. Therefore, it is particularly sensitive to the *taste* (ballads, rock songs, etc.) of the reader and not just to his *interest* (cars, baseball, etc.). Today, there exist a large number of music weblogs offering a multitude of posts and information to their community, hence the need for a

recommender system. Our system, JukeBlog, aims at helping users find new weblog posts by guiding their reading and listening choices. It is a feature combination hybrid recommender system using collaborative filtering and content filtering techniques in order to recommend posts.

This paper is organized as follows: Section 2 introduces the concepts of *weblogs* and *recommender systems*; Section 3 describes our system, JukeBlog: the collaborative filtering technique we used, how we implemented content filtering and the feature combination technique we developed for hybridization; Section 4 highlights the testing procedure and the results observed; and Section 5 concludes the paper and presents directions for future work.

2 Concepts

In this section, we introduce two concepts that are related to our system: weblogs and recommender systems.

2.1 Weblogs

The term *weblog* (or more simply *blog*), coined by John Barger in December 1997, is used to designate a website that contains time-stamped posts. *Lessig Blog* [URL 5], *Paolo Massa Blog* [URL 6] and *Dooce* [URL 7] are examples of popular weblogs. Although the definition of a weblog is in the course of being widened, it is traditionally a web site *edited by a single person*. It contains *posts*, composed of text or any other type of content and is updated on a regular basis. The content of a weblog is *free to public access* and is organized using a *post archiving system*. Although that is not always the case, posts are usually presented to the reader in reversed chronological order. Indeed, weblogs offer great flexibility in their representation of information. A detailed definition of what constitutes a weblog can be found in [Aïmeur *et al.*, 2005].

A great number of blog-watchers (the term used to designate people who read many blogs) visit weblogs as they would visit traditional websites. Others prefer to register the addresses of regularly read weblogs into an *aggregator*. An aggregator is a piece of software or a website that regroups a number of weblogs (chosen by the aggregator user) and puts

all of their posts in one place for easy follow-ups. These tools are possible due to the existence of *syndicated* content. Syndicated content are automatically generated XML (eXtensible Markup Language) files containing post data and meta-data, updated each time weblogs are updated. These files can be of various formats, such as: **RSS** (Really Simple Syndication), **RSS2** (Really Simple Syndication 2), **RDF** (Resource Description Framework) and **Atom**.

2.2 Recommender systems

Recommender systems are a revolution in the world of social interaction. They permit the automation of concepts such as *word of mouth* and *groups of interest*. Their goal is to target the information presented to the user so that he can make a wiser choice of items in a wide range of subjects. Recommendation algorithms can be categorized based on the information they use to make recommendations. Two commonly used types of algorithms are *collaborative filtering*, when the system bases the recommendation on the *ratings* given by other users of the system, and *content filtering*, when it bases the recommendation on the *content* of the item. A recommender system using more than one category of algorithms is called a *hybrid* system. There exist seven types of hybrid systems [Burke, 2002], including *Weighted* (two or more algorithms are applied independently and the results are then mixed by applying a weight to each of them), *Switching* (the recommender system chooses between a number of algorithms depending on the situation) and *Feature Combination* (data from different type of algorithms are combined into a single calculation). One of the most important papers that relate to feature combination is [Basu *et al.*, 1998]. The authors developed a recommender system (*Ripper*) that uses content-based techniques over a data set that contains both content and collaborative data. Our system, JukeBlog, uses a feature combination hybrid approach combining collaborative filtering and content filtering by considering content data as collaborative data in the collaborative filtering equation. [Miller, 2005] introduces the idea of combining blogs and recommender systems. The article suggests that the current weblog's infrastructure could be used in order to build a platform on which a personal recommender system could run. That recommender system would not, as opposed to JukeBlog, recommend blog posts to its users. Instead, the XML infrastructure of blogs would constitute a way to ease the publication of collaborative data used in a recommender system.

3 JukeBlog

JukeBlog is a music weblog recommender system. It retrieves syndicated weblog information on a daily basis from the Internet and stores it into its own database. Later on, JukeBlog combines collaborative filtering and content filtering techniques into a feature combination hybrid recommender system to suggest and recommend posts to users. Figure 1 illustrates JukeBlog's user interface.

3.1 Incomplete information

One of the issues encountered during the development of JukeBlog relates to the choice of the information included in the XML documents. That choice is up to the discretion of the weblog author, and is totally out of our control. But the majority of information retrieval algorithms are based on the analysis of complete texts. Therefore, in the eventuality that a weblog author includes only a sample of the post's content in the syndication file, its analysis becomes impossible. In order to solve this problem, we base our content filtering analysis on a user-given keyword system instead of the traditional text-analysis. Our database contains only meta-data about the posts, and includes a link to the web page of the post so that the user can view it in its original context.

3.2 JukeBlog's approach

In JukeBlog, users are constantly presented random music weblog posts. Even when a user registers and enters the system for the first time, he is presented a random post. After having read the post (and listened to the song), the user has to rate it on a scale ranging from 1 to 5. The user is also presented the opportunity to assign up to 3 different keywords to the post (used for content filtering). After the user has given a number of ratings, JukeBlog is able to make score predictions for posts in the database. These posts are then recommended to the user in decreasing order, from the most to the least relevant.



Figure 1: JukeBlog's user interface

Collaborative filtering

The first method used to predict ratings is collaborative filtering. We used a common technique for that (used by GroupLens [Resnick *et al.*, 1994] amongst others). It is a memory-based algorithm, so it uses the whole set of ratings given by the users. Users must give a rating ranging for 1 to 5 to the posts presented to them. The system then computes

the similarity between each user, using the Pearson correlation formula (Equation 1), in order to create a neighbourhood for each user:

$$corr(A,B) = \frac{\sum_j (v_{A,j} - \bar{v}_A)(v_{B,j} - \bar{v}_B)}{\sqrt{\sum_j (v_{A,j} - \bar{v}_A)^2 \sum_j (v_{B,j} - \bar{v}_B)^2}}$$

Equation 1: Pearson correlation formula

In this equation, j ranges over the posts, $v_{A,j}$ refers to the rating given by user A to post j , and \bar{v}_A refers to the average rating given by user A to posts rated in common with user B . Once the neighbourhood of a user is computed, the prediction computation step is performed. It is based on the ratings given to an item and on the similarity indexes computed using Equation 1. The equation used is quite simple:

$$pr_{A,j} = \bar{v}_A + \frac{\sum_i (v_{i,j} - \bar{v}_i) \times corr(A,i)}{\sum_i |corr(A,i)|}$$

Equation 2: Rating prediction formula

In this equation, i ranges over the set of users, $corr(A,i)$ refers to the similarity between user A and user i (calculated using Equation 1), \bar{v}_A refers to the average of ratings given by user A to posts rated in common with user i and $v_{i,j}$ refers to the rating given to item j by user i .

Content filtering and information retrieval

We deemed important not to base the predictions solely on collaborative filtering. Taking into account the information on the content of a post adds a crucial new dimension to the recommender system and refines its output. However, as stated in Section 3.1, some authors choose to syndicate only a sample of their posts, and text analysis based on incomplete data often results in incorrect categorization. It would have been possible to circumvent the problem by retrieving the information directly from the post’s web page (*à la* web search engine such as Google) instead of retrieving the syndicated document. However, that method is not useful in our context. Web pages of posts do not only contain post-specific information, but may also contain other data such as blog-specific information, advertisers or “similar blogs” lists. Such information is irrelevant to the text analysis. The most common weblog categorization method is the association of keywords to posts. Categorization engines like *Flickr* [URL 8] and *del.icio.us* [URL 9] have developed tools that permit such a categorization. We decided to reuse the idea and let users assign keywords to posts in JukeBlog’s database. An advantage of the attribution of keywords over the automated content analysis is that information collected about posts is related directly to the songs rather than the text. Although important progress has been observed in the multimedia information retrieval

domain, we believe that the categorization would be more successful, especially if we let the users do it themselves. Moreover, since the source data is distributed, the system has no direct control over the music files. Therefore, it is harder (even impossible in many cases) to locate these files in order to analyse them. For example, the address may not point directly towards a file with the “.mp3” extension, but towards an address that would redirect the user towards the file. What follows is an explanation on how the socially-controlled keyword system works. When a user rates a post, the system offers him the option to choose up to 3 different keywords associated to the post. Keywords are chosen from a drop-down list of 25 words. Keywords have a double function: they help the user’s taste profiling (user A likes humorous, happy and cheerful songs, but hates sad and romantic songs) and they help determine the posts’ content. In addition, JukeBlog will not allow a user to assign opposite terms, such as “calm” and “aggressive”, to the same post.

Hybridization

JukeBlog uses a Feature Combination hybrid approach, combining both content and collaborative filtering. However, instead of using content filtering techniques over a collaborative filtering-aware system, as in [Basu *et al.*, 1998], we use a collaborative filtering technique on data of mixed types. Feature combination techniques let us incorporate inherent similarity considerations into the computation of predictions. Let’s use an example in order to illustrate our method. Table 1 highlights all the ratings of user A :

Table 1: Ratings of user A

Post	Rating	Keyword
P1	5	Smooth
P2	1	Aggressive
P3	1	Aggressive
P4	4	Cheerful
P5	2	Cheerful

These entries are then grouped by keywords. The average of the similar keywords’ ratings is assigned to that keyword. Table 2 illustrates the result of aggregating the data in Table 1.

Table 2: Aggregated ratings of user A

Keyword	Rating	Number of posts (n)
Smooth	5	1
Aggressive	1	2
Cheerful	3	2

Table 2 represents the “query” part of information retrieval. It is a weighted keyword query in which we ask the recommender system to retrieve posts that relate to “Smooth” music (weight of 5), and to exclude “Aggressive” (weight of 1) music. Next, during the computation of the prediction for a certain post j , we compare the whole set of keywords

given to post j by all the users to the data from Table 2 (i.e. to all the keywords given by user A to any post). Table 3 illustrates the keywords given to post j :

Table 3: Keywords for post j

User	Keyword
B	Cheerful
C	Smooth
D	Smooth
E	Romantic

Each keyword for post j that corresponds to a keyword given by user A is considered in our calculation, as if it were another virtual user V that gave that rating to post j . In our example, we would simulate 3 users: one that gave post j a rating of “3” (because of the keyword *Cheerful*), and two that gave it a rating of “5” (because of the keyword *Smooth*, present twice). The advantage of our hybrid method is that it is easy to combine both types of information together: content filtering information and collaborative filtering information. We adapted Equation 2 in order to incorporate content and collaborative filtering in our prediction. Equation 3 illustrates the adapted rating prediction equation:

$$pr_{A,j} = v_A + \frac{\sum_i (v_{i,j} - \bar{v}_i) \times corr(A,i) + \sum_k n_k w_k}{\sum_i |corr(A,i)| + \sum_k n_k}$$

Equation 3: Our adapted rating prediction formula

Here, i ranges over the users, k ranges over all the keywords assigned by *any* user to post j , w_k is the rating assigned to keyword k by user A , and n_k is the number of posts for which user A has assigned keyword k . Equation 4 illustrates an example of computation:

$$pr_{A,j} = v_A + \frac{\sum_i (v_{i,j} - \bar{v}_i) \times corr(A,i) + 2 \times 3 + 1 \times 5 + 1 \times 5}{\sum_i |corr(A,i)| + 2 + 1 + 1}$$

Equation 4: Computation example

The retrieval model is a “Best Match Retrieval” (as opposed to “Exact Match Retrieval”) model, because the posts returned to the users are in order of relevance to the query, and can only partially match it. We chose to recommend all the posts for which the algorithm predicts a score that is higher than the user’s average rating. Algorithm 1 shows the procedure:

```

foreach (user i having rated post j)
    upper = upper + ( rating[i][j] - average( rating[i] ) )
                * similarity( a, i )
    lower = lower + average( rating[i] )

foreach (keyword k assigned to any post p by user a)
    w[a][k] = w[a][k] + rating[a][p]
    n[a][k] = n[a][k] + 1

foreach (keyword k assigned to post j by any user)
    foreach (keyword l assigned by user a)
        if (k == l)
            upper = upper + w[a][k]
            lower = lower + n[a][k]

prediction[a][j] = average( rating[a] ) + upper / lower

```

Algorithm 1: Algorithm of our hybrid system

4 Testing and results

The aim of JukeBlog is to help music weblog community members in their browsing activity. JukeBlog can be considered a successful system if its rating predictions are close to its users’ actual ratings. To test our recommender system, we limited our database to posts published on a particular day, giving us 82 posts to rate. A total of 64 users were randomly assigned to one of two cases: they were either given recommendations based solely on the collaborative filtering algorithm, or they were given recommendations based on our hybrid method. We then calculated the Mean Absolute Error (MAE) [Sarwar *et al.*, 2001] on both of these sets. The MAE calculates the average error made by the system between its predictions and the ratings given by the user. The more accurate the system is, the lower the MAE should be. The results are shown in Figure 3.

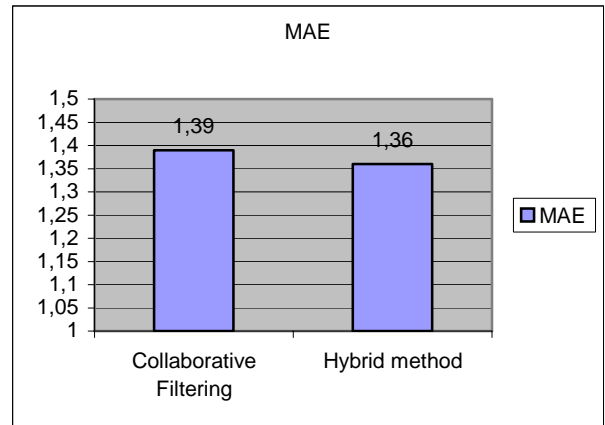


Figure 3: Mean Absolute Error

Collaborative filtering gives us a MAE of 1.39, while our hybrid method gives us 1.36. Based on these results, the Hybrid method appears to offer a small advantage over collaborative filtering. Since the content filtering part of our

hybrid method is based on keywords given by the users, a rich input is needed in order to see a difference between the two algorithms. However, testing a music weblog post is a time consuming task, and most of our testers only assigned a small number of ratings and keywords. Half of them evaluated less than 10 posts, which is a low number to profile the users and posts. Therefore, these results are preliminary. More testing is required to determine the benefits of JukeBlog. On a qualitative note, JukeBlog has received very favourable reviews from some highly visited weblogs.

5 Conclusion

Today, the number of weblogs is very high, resulting in sub-optimal content discovery. That is why we believe that a recommender system can play an important role in the weblog community. However, traditional content filtering in the field of uncontrolled distributed items involves a number of issues. These issues range from the heterogeneous nature of the items to recommend (both in the form and in the content) to the various amount of information we get concerning each of them. That is the reason why traditional text/multimedia-analysis information retrieval techniques can hardly be put into application. Therefore, we developed a technique that makes use of the ever important “community” concept in weblogs to attach content information to posts. We then combined collaborative filtering data with content filtering data in JukeBlog, our Feature Combination hybrid recommender system. At this stage, it is still early to determine both the applicability of a recommender system to blog posts and the value of our Feature Combination hybrid, but further testing will provide better indication. The next challenge is to incorporate information about interrelation between posts to the computation of predictions (i.e. which posts link to which other posts), using data gathered by Technorati.

References

- [Aïmeur *et al.*, 2005] Aïmeur, E., Brassard, G., Paquet, S. Personal Knowledge Publishing: Fostering Interdisciplinary Communication. *IEEE Intelligent Systems*, Vol. 20, no. 2, pp. 46-52, March/April 2005.
- [Basu *et al.*, 1998] Basu, C., Hirsh, H., Cohen, W. Recommendation as Classification: Using Social and Content Based Information in Recommendation. *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. July 1998, Madison, WI, USA, pp. 714-720.
- [Burke, 2002] Burke, R. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, Vol. 12, no. 4, pp. 331-370, 2002.

[Miller, 2005] Miller, B. N. Toward a Personal Recommender System. *Workshop: Beyond Personalization 2005, International Conference on Intelligent User Interfaces*. January 2005, San Diego, CA, USA, pp. 97-98.

[Resnick *et al.*, 1994] Resnick, P., Iacovou, N., Sushak, M., Bergstrom, P., Riedl, J. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. *Proceedings of the 1994 Computer Supported Collaborative Work Conference*. October 1994, Chapel Hill, NC, USA, pp. 175-186.

[Sarwar *et al.*, 2001] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. Item-based Collaborative Filtering Recommendation Algorithms. *Proceedings of the Tenth International Conference on World Wide Web*. May 2001, Hong Kong, Hong Kong, pp. 285-295.

[URL 1] <http://www.technorati.com/>

[URL 2] <http://teachingtheindiekidstodanceagain.blogspot.com/>

[URL 3] <http://fluxblog.org/>

[URL 4] <http://tofuhut.blogspot.com/>

[URL 5] <http://www.lessig.org/blog/>

[URL 6] <http://moloko.itc.it/paoloblog/>

[URL 7] <http://www.dooce.com/>

[URL 8] <http://www.flickr.com/>

[URL 9] <http://del.icio.us/>

A Proposal for an Agent-based Architecture for Context-aware Personalization in the Semantic Web

Wolfgang Woerndl and Georg Groh

Technische Universitaet Muenchen, Institut fuer Informatik

Boltzmannstr. 3, 85748 Garching b. Muenchen

Germany

{woerndl,grohg}@in.tum.de

Abstract

In this short paper we describe the goals and questions associated with our project Cairsw eb. In this project we aim at the construction of a framework of agents for personalized context-sensitive Information Retrieval in a massively distributed information landscape, in which questions of privacy and access control on the one hand and semantic crawling / “explicitification” on the other hand are the main focus. By “explicitification” we mean the process of generating semantically rich interfaces to information sources on the web with the help of Semantic Web standards.

1 Introduction

The goal of the Semantic Web is to associate meta data with (Web) resources and make their meaning understandable to agents and other “intelligent” programs. While static and semi-static Web Pages already begin to adopt meta data schemes such as Dublin Core, the „Hidden-Web“ remains largely inaccessible to seamless information management services such as context-sensitive information retrieval. The Semantic Web can contribute to the solution of the Hidden-Web-problem by providing declarative access interfaces to Hidden-Web-Data-Sources using standardized languages such as OWL [McGuinness and van Harmelen, 2004]. Formerly, related approaches have been suggested in the fields of data-mining and in database-wrapper-induction [Kushmerick, 2000]. In contrast to these proprietary approaches, the Semantic Web allows for sufficient declarative expressiveness on the basis of a stack of standards which allow for a uniform information access.

Besides that, “explicitification” of contents in the form of declarative meta data which uses standardized Semantic Web standards has several further advantages if these meta data are backed up by sufficiently expressive ontologies. These ontologies need to be accepted by a large share of the Internet’s main content providers. Especially in the field of mobile information management, a semantically rich meta-data-infrastructure can contribute to personalized and context-sensitive information services. Context-adapted personal information needs stated in declarative form can be much more suitably matched against a distributed pool of available information sources with an increased semantic

depth when these sources also “explicitify” their semantics in a declarative standardized format.

A usage scenario for context-aware applications is personalization in a mobile environment. Users want to access relevant information with their mobile phones or PDAs. For example, a user might want to search for a restaurant to her liking in her vicinity that accepts her credit card. The context and profile of the user includes her current position, her eating preferences, other user profile attributes and maybe restaurant ratings of friends in this example. The Semantic Web provides a foundation for such services but a lot of further investigation is necessary on the way to implementing such “intelligent” services.

Our project “Cairsw eb” (Context-Aware Information Retrieval in the Semantic WEB) aims at providing personalized services based on obtaining and managing user context. The context is then matched with Semantic Web information sources that we enrich automatically with semantic “explicitification”. The project is still in the earlier stages, this paper outlines our ideas.

The rest of this paper is organized as follows. In section 2 we explain our architecture. Then we explain the basic framework for semantic “explicitification” of information sources and describe the research questions that we aim at. Section 4 gives an overview of related work. Finally, we conclude with a brief summary.

2 Architecture

The first step in our project is to develop an architecture for the scenario explained above. The agenda includes the following questions:

- Which components (agents) are needed for context-aware information retrieval in the Semantic Web?
- What is the specific task of the agents and how can we design the interfaces between the components?
- How can we model and manage “context” of users?
- How can we “explicitify” the meaning of information sources?

- And finally, how can we match user context and information services to provide meaningful services?

We propose an agent-based architecture because the components in our scenario (various information sources and personalization programs) are inherently distributed and autonomous. Our proposal for the architecture is shown in Figure 1.

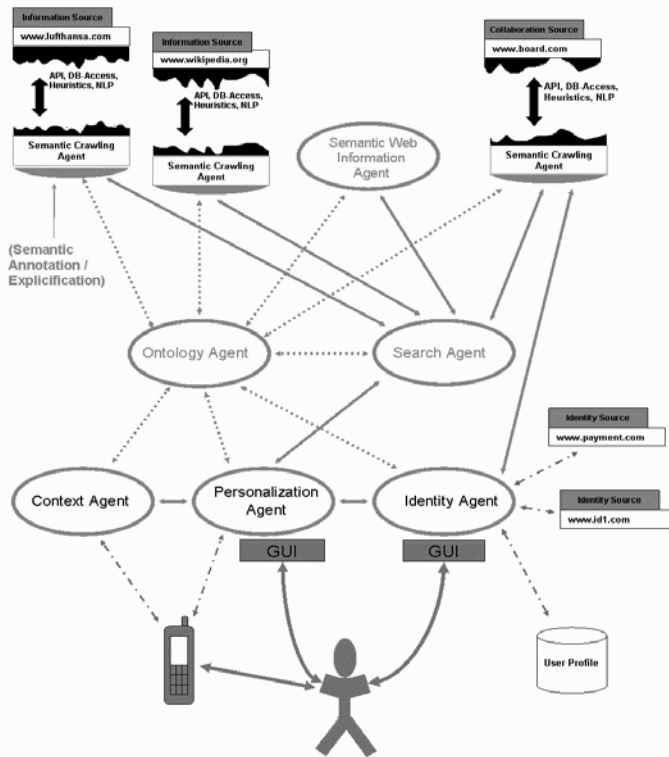


Fig. 1: Architecture

In general, the following aspects have to be considered when designing the agents [Woerndl, 2003]:

- Interoperability and heterogeneity: handle multiple ontologies, for example different user profile models, and integrate mapping mechanisms between ontologies
- Customization of information retrieval: this is done in our approach by including context and profile information in search queries
- Segmentation of search: how can a search request be divided into separate, “smaller” queries?
- Balance the trade-off between expressiveness and (computational) complexity

On the user side, our architecture includes Personalization, Context and Identity Agents (Fig. 1). The Personalization Agent is the “Semantic Web browser” for the user. It

provides a user interface which allows for a management of a user’s information needs.

The difference between the Context and Identity Agent is that the latter manages user profile information that is rather static and does not change very often. The user profile includes demographic and personal information such as Email addresses, interests, ratings, past transactions and so on. User profile information can be explicitly entered and updated by the user or implicitly obtained by the Personalization Agent (e.g. by observing the user’s Web clicks). In any case, privacy is very important when managing information about users. The fundamental idea is to separate the user profile from services that are using it. In doing so, the user can keep control of her profile and restrict access to her personal data. The Identity Agent coordinates different sources of user profiles, authentication data (e.g. passwords and certificates) and controls access to it. We have developed an authorization mechanism to user profiles in federated identity management that can be used in this scenario [Woerndl, 2004]. Another task of the Identity Agent is to manage different identities of one user, for examples “work” and “private” roles with different user profile attributes.

Unlike the profile, the user’s context includes very dynamic and transient information such as her current physical location, her last search request, her current devices (e.g. if she is currently using a laptop computer or a PDA), possibly her “virtual location” (what is she doing right now?) and so on. Context information is never entered by the user. It is the task of the Context Agent to acquire relevant context information.

We believe it is useful to make the distinction between static (“identity”/“profile”) and dynamic (“context”) user information, because context requires substantially different management (since it is very transient and dynamic) and is more difficult to acquire. An example for context is the current location of the user which is useful for content adaptation at the moment. On the other hand it is not beneficial to persistently retain user locations over time in the profile. When “old” context is important, it may be transferred to the Identity Agent und stored in the user profile.

The Search Agent receives search requests from the Personalization Agent. A search request consists of the query, the (whole) context and (the relevant part of) the user profile. The query may be explicitly entered by a user but could also be generated automatically and proactively by the Personalization Agent based on the profile or context of the user.

To perform searches, the Search Agent keeps an index of information sources (just like a traditional keyword-based search engine). The Search Agent is responsible for proactively matching the user’s information needs with the information provided by the information source. It also is responsible to answer queries using the knowledge about the available information sources.

The Ontology Agent manages different ontologies that are used by the other agents. This does include possible mappings between ontologies. The Ontology Agent com-

municates with all other agents that use ontologies which is not depicted in Fig. 1 for simplicity reasons.

We divide information sources into three types (see Fig. 1):

- Semantic Web Information Agents provide access to data using Semantic Web technologies such as ontologies
- (Traditional) Web and other information sources which are annotated with meta data by Semantic Crawling Agents
- Collaboration sources such as information in bulletin boards and discussion forums

Collaboration sources are also annotated by Semantic Crawling Agents. The difference between information and collaboration sources is that collaboration sources are assigned to user identities. User identities can be pseudonyms in a discussion board or a real identity proven by a digital certificate. Hence, it is possible to use this information to improve the personalization. For example, a user (resp. identity) Alice states that a restaurant is recommended in a collaboration source, and the personalization agent of Bob can derive that Alice is a “friend” of Bob. The personalization agent of Bob can then use this information to assign a higher value to the recommendation of Alice in the personalization process. This approach could also foster the building of trust in the Semantic Web. Within our group, an extensive modelling framework for personal relations has been developed [Galla, 2004].

The Semantic Crawling Agents are intended as a means for describing the content of information sources in a declarative way. Either an information source (e.g. a company, organization or community) chooses to take care for the semantic “explicitification” of its contention its own. In this case, the Semantic crawling agent is provided by the information source itself. Or the semantic Crawling agents are specifically tailored to each information source by third party organizations, communities or companies. An example could be an independent travel agency that would semantically crawl and explicitify (Hidden-Web-)information-sources of airlines, hotels and public transport systems to provide uniform access to these sources with a high semantic connectivity.

In the next section of this paper we focus in more detail on the Semantic Crawling Agents and the explicitification of information sources.

3 “Explicitification”

In section 2 we discussed the general outline of our approach. One central element of the architecture were the Semantic Crawling Agents. The conventional Semantic Web scenario demands that every information source should explicitify its contents by annotating them with semantically rich meta data by itself or (even more radical) that the information sources switch their whole content representation to Semantic Web standards. The great disadvantage is that this very costly and radical step offers the individual information source no immediately perceivable added value,

because without a critical mass of other sources taking the same measures the semantic annotations will not be usable.

Instead of that, we propose that the existing structures on the Web will not have to be altered but that rather proprietary Semantic Crawling Agents will take care of creating semi-automated annotations of the various hidden and not hidden contents of the information sources.

Every information source is assigned such a custom-tailored agent and it is up to the respective source whether it will provide such an agent or whether this will be left to some external information-broker. In contrast to large scale search engines which use semantically rather simple heuristics (e.g. PageRank and counting words in case of Google) and rather simple crawling schemes, such information brokers would specialize on interrelating information from comparatively few information sources but with much increased semantic depth. From the point of view of functional granularity, these brokers can be seen as part of the “missing link” between usual search engines and topically specialized, unified query systems for books or flight-bookings which span all major publishers / airlines.

What is the advantage for e.g. the third party travel agency of using (and possibly creating) several Semantic Crawling agents semantically explicitifying airline-, hotel- and travel-information sources compared to just using e.g. existing cross-company airline- and hotel-booking systems or compared to gathering and integrating the required information in a large database? If the information needs of such a third party information source (such as the travel agency in the example above) can readily be satisfied by such means, the semantical explicitification process does not provide any advantages at all.

The approach of semantically declaring information explicitly unfolds its potential only in a massively distributed information landscape where new and possibly “unexpected” information can be incorporated into e.g. a search process. In the example from above, information systems about local weather or tickets for events may be effortlessly integrated without the need for gathering and maintaining this information in a local information space.

Semantic Crawling agents provide a Semantic-Web-interface to the information contents of the single (Hidden-Web-)information sources. They are not intended as a new form of traditional crawlers which “gather” all data from e.g. a web-page and deliver them to a central indexing and storage institution (e.g. a search engine). Thus they will not require extensive storage capacities to do their job.

The questions that need to be investigated include:

- Is it sufficient to explicitify fact-oriented information-sources with RDFS triples [Hayes, 2004]?
- Do we need the expressiveness of OWL [McGuinness and van Harmelen, 2004]? Which of the three OWL levels of expressiveness will be required?
- How can models of statistical natural language processing together with parsing and tagging techniques be used to explicitify more text-oriented information sources beyond the tradi-

tional information retrieval models (such as the vector model [Baeza and Ribeiro-Neto, 1999])?

- How can existing results on database- and information-source-wrapper-induction be used for the construction of Semantic Crawling Agents?

4 Related Work

Kushmerick [2000] discusses the matter of database- or general information-source-wrapper-induction. Wrappers are closely related to Semantic Crawling Agents and introduce views on existing data- or information-sources. The paper and its related publications discuss automatic induction of such wrappers.

A number of approaches use an agent-based architecture to personalize information access, for example Fredo [Bothelo *et.al.* 2003], GraniteNights [Grimnes *et.al.* 2003] or SmartPush [Kurki *et.al.* 1999]. Fredo is a generic domain-independent broker that creates value added information taking into account the preferences specified by its clients. Fredo uses ontology services and yellow pages services to discover a set of agents that can provide information relevant to its clients' requests.

GraniteNights is a multi-agent application which allows a user to schedule evening appointments. The architecture is comparable to our approach. Agents within the application infrastructure are organised according to a series of pre-defined roles such as information agents (which are wrappers for RDF resources [Manola and Miller, 2004]) and a profile agent (for managing user data, such as preferences).

SmartPush is an approach for information filtering. Documents are thereby associated with meta data in RDF. These applications take advantage of the Semantic Web to provide personalized information access. Their agent-based architecture is comparable to approach. However, Fredo, GraniteNights or SmartPush do not explicitly incorporate context information.

Chen *et.al.* [2004] present a "smart meeting room system" that provides services for meeting participants based on Semantic Web technologies. Part of their architecture is a context broker agent that maintains a shared context model. While their approach to handling context and profiles is comparable to our ideas, their application scenario and the resulting services are quite different.

UbisWorld [Heckmann, 2001] is an interesting approach to model parts of the real world like an office or a city and dynamically adapt access to information sources (see <http://www.w2m.org>). Part of the project is a "General User Model and Context Ontology (GUMO)" [Heckmann *et.al.* 2005] that we want to evaluate for usage in our scenario.

5 Conclusion

The goal of our project Cairsworld is to investigate questions in connection with the construction of an architecture of agents which provide for a personalized and context-sensitive Management of Information. In order to fulfil this goal, a number of agent types are suggested that take over specialized duties. Among them the Personalization Agent

the Search Agent and the Semantic Crawling Agents are of special interest.

The main contribution of this short paper in the early phase of our project is to propose an agent-based architecture for adaptive and context-aware information retrieval in the Semantic Web, identify research issues within this architecture and outline some ideas for solutions. Thus, most of our work is still pretty sketchy and preliminary at this time but we are working on substantiating and implementing our ideas. We have implemented some prototype Personalization Agents using the JADE framework (<http://jade.tilab.com/>). Next steps include more work on the user and context model. We also plan to implement Semantic Crawling Agents by utilizing the respective Web Services API's of Amazon, Ebay and Google as a starting point.

References

- [Baeza and Ribeiro-Neto, 1999] Ricardo Baeza-Yates, Berthier Ribeiro-Neto: *Modern Information Retrieval*, Addison Wesley, 1999
- [Bothelo *et.al.*, 2003] Luis Bothelo, Hugo Mendes, Pedro Figueiredo, Rui Marinheiro. Send Fredo off to Do This, Send Fredo off to Do That. In *Proc. Cooperative Information Agents VII*, pp. 152-159, 2003.
- [Chen *et.al.*, 2004] Harry Chen, Tim Finin, Anupam Joshi, Lalana Kagal, Filip Perich and Dipanjan Chakraborty. Intelligent Agents Meet the Semantic Web in Smart Spaces. *IEEE Internet Computing*, 8(6): 69-79, Nov. 004
- [Grimnes *et.al.*, 2003] Gunnar Aastrand Grimnes, Stuart Chalmers, Pete Edwards, Alun Preece. GraniteNights – A Multi-agent Visit Scheduler Utilising Semantic Web Technology. In *Proc. Cooperative Information Agents VII*, pp. 137-151, 2003.
- [Galla, 2004] Michael Galla: *Social Relationship Management in Internet-based Communication and Shared Information Spaces*. PhD Thesis, May 2004, Technische Universität München, Faculty for Computer Science
- [Hayes, 2004] Patrick Hayes. *RDF Semantics*. W3C Recommendation. Feb. 2004
- [Heckmann, 2001] Dominik Heckmann. Ubiquitous User Modeling for Situated Interaction. In *Proc. of the 8th International Conference on User Modeling*, pp. 280-282, 2001
- [Heckmann *et.al.*, 2005] Dominik Heckmann, Tim Schwartz, Boris Brandherm, Michael Schmitz and Margeritta von Wilamowitz-Moellendorff. GUMO - The General User Model Ontology. In *Proc. of UM 2005: International Conference on User Modeling*, 2005.
- [Kurki *et.al.*, 1999] Teppo Kurki, Sami Jokela, Reijo Suilonen and Marko Turpeinen. Agents in Delivering Personalized Content Based on Semantic Metadata. In: *AAAI Symposium on Intelligent Agents in Cyberspace*, pp. 84-93, AAAI Press, 1999.

- [Kushmerick, 2000] Nicholas Kushmerick. Wrapper Induction: Efficiency and Expressiveness, *Artificial Intelligence*, 118(1-1), pp.15-68, 2000
- [Manola and Miller, 2004] Frank Manola, Eric Miller. *RDF Primer*. W3C Recommendation, Feb. 2004
- [McGuinness and van Harmelen, 2004] Deborah L. McGuinness and Frank van Harmelen. *OWL Web Ontology Language Overview*. W3C Recommendation, Feb. 2004
- [Woerndl, 2003] Wolfgang Woerndl. Requirements for Personal Information Agents in the Semantic Web. In *Proc. 4th Conference on Distributed Applications & Interoperable Systems (DAIS2003)*, Nov. 2003
- [Woerndl, 2004] Wolfgang Woerndl. Authorization of User Profile Access in Identity Management. In *Proc. IADIS International Conference WWW / Internet*, Oct. 2004

An NCM-based Framework for Navigation Recommender Systems

Tong Zheng and Randy Goebel

Department of Computing Science

University of Alberta

Edmonton, Alberta, Canada

{tongz, goebel}@cs.ualberta.ca

Abstract

Within the growing literature on web mining, there is a relatively coherent thread of ideas focused on improving web navigation through recommender systems. In this paper we focus less in the recommendation algorithms than in the broader questions of how such recommender systems can be constructed and evaluated.

we focus on the idea of web usage mining, and present a general framework for deploying the mining results as plug-and-play components in a dynamic recommender system. The offline learning module and the online recommendation module are clearly decoupled in our framework. The generalized objects created by the application of learning methods are called Navigation Compression Models (NCMs), and we show that these NCMs can be easily plugged into the recommender system to make corresponding recommendations.

We also present a simulation-based approach to evaluate the value of the recommender systems conforming to this NCM-based framework as well as the effectiveness of different NCMs. The evaluation is done non-intrusively by running a simulation on previously recorded web usage logs to measure the potential navigation improvement.

We evaluate the improvement of user navigation using a quantitative measure called *navigation improvement (NI)*, which indicates whether we are actually “improving” the user’s navigation by reducing the number of hyperlinks traversed to find “relevant” pages.

1 Introduction

With the rapid and continuous growth of the World Wide Web (WWW or Web), finding useful information in this enormous space becomes ever more difficult and time consuming. This has led to the fast emergence of web recommender systems as an effective approach to help the web users address this problem.

The basic idea of web recommender systems is to help users find the information they want on the Web by providing

suggestions to potentially useful documents or hyperlinks. And web mining was before long introduced into this scenario for its ability to discover “interesting” knowledge (or patterns) from the vast amount of web data that can then be used to generate dynamic recommendations within such recommender systems.

Within the growing literature on web mining, there is a relatively coherent thread of ideas focused on improving web navigation through recommender systems [Mobasher *et al.*, 1999; Zhu *et al.*, 2003; Olston and Chi, 2003; Baraglia and Silvestri, 2004].

A usage-based web personalization system was proposed in [Mobasher *et al.*, 1999] that makes personal recommendations to individual users based on the results of a number of data mining techniques such as association rules and URL clusters. SUGGEST [Baraglia and Silvestri, 2004] is also a usage-based web recommender system but focuses on eliminating the offline data preprocessing and learning modules so that the entire process can be performed online. [Zhu *et al.*, 2003] proposed another interesting approach which learns *information content* words from the content of users’ current navigation path together with the browsing features, and then uses these words to dynamically retrieve “relevant” web pages as recommendations. The ScentTrails technique [Olston and Chi, 2003] took a different approach: a keyword-based search is first performed on a set of keywords provided by the user, and then the “information scent” of those search result pages is spread back to the user’s current location as an indication for recommendations.

Web recommender systems can be categorized based on different characteristics. For example, there exist server-side and client-side recommender systems. While server-side recommender systems help users navigate in a specific web site [Mobasher *et al.*, 1999; Olston and Chi, 2003; Baraglia and Silvestri, 2004], the goal of client-side recommender systems is to better capture the interests of an individual user and help the user find useful information among the entire web [Zhu *et al.*, 2003].

Recommender systems can also be categorized based on the methods used to generate recommendations. Some widely-used methods include information retrieval, collaborative filtering [Resnick *et al.*, 1994], and web mining. And web-mining-based recommender systems can be further classified as web-usage-mining-based [Mobasher *et al.*, 1999;

Baraglia and Silvestri, 2004], web-content-mining-based [Olston and Chi, 2003]¹, and hybrid-web-mining-based [Zhu *et al.*, 2003].

In this paper we focus on server-side navigation recommender systems based on web usage mining. The idea is to make personal recommendations to each individual user based on the knowledge learned from previous navigation patterns of aggregate users.

A typical web-usage-mining-based recommendation system requires at least two essential components: (1) a knowledge base, which contains the knowledge learned from aggregate user behaviors, and (2) a recommender, which makes recommendations to each individual user based on the user's current status and related knowledge retrieved from the knowledge base. To be more specific, the knowledge base is a set of learning results obtained by applying various data mining techniques to the web usage data. In our framework, these learning results are generalized as Navigation Compression Models (NCMs).

While NCMs represent an abstract form of user navigation patterns, there is a significant problem within these patterns: a traveled path is not necessarily a desired path. Therefore, we propose a recommendation mechanism which ignores those auxiliary pages and makes recommendations only on "relevant" pages. However, it is not always clear whether a page is relevant or not. Therefore, it is often the case that the relevance of pages has to be determined using heuristics or models learned from pre-labeled training samples.

Like other recommender systems, we require some evaluation method that can measure the value of the recommendations. Precision and recall are two widely-used criteria for measuring the accuracy of recommendations. But they are not capable of evaluating the actual improvement in user navigation, which we believe is a more important measure for assessing the real value of recommendations in this domain. Like the idea of data mining is to discover knowledge that is useful and previously *unknown*, the goal of navigation recommender systems should be to help users find the information they *can not find* or *can not find quickly* by themselves.

Note that an accurate recommendation is not necessarily a "useful" recommendation. For example, if the recommended page is exactly the page the user originally planned to visit next, the recommendation is actually useless.² In our framework, we evaluate the improvement of user navigation using a quantitative measure called *navigation improvement (NI)*, which indicates whether we are actually "improving" the user's navigation by reducing the number of hyperlinks traversed to find "relevant" pages.

The rest of this paper is organized as follows. Section 2 describes the architecture of the framework as well as the idea, the creation and use of Navigation Compression Models. Section 3 describes the dynamic recommendation mechanism employed by the online recommendation module. Section 4

¹Here we consider a broad concept of web content that also includes web structure.

²For simplicity, we omit the possibility that the user navigation could be "improved" if the user spend less time finding the desired page from the recommendations than from the current page.

introduces the simulation-based evaluation method used to measure the navigation improvement. Section 5 describes some preliminary experiments and Section 6 concludes this paper.

2 Architecture of the NCM-based Framework for Recommender Systems

A typical architecture for web-usage-mining based recommender systems [Mobasher *et al.*, 1999] generally requires a module to support data capture and preprocessing, a module to support the specification and deployment of a repertoire of learning methods, a module to support the recommendation process using the learning results, and, perhaps less common, an explicit module designed to support evaluation of any combination of the first three modules.

In the deployment of this architecture to the navigation improvement problem, the crucial middle component is that object to be created by various learning algorithms, and then inspected to see whether the learning algorithm has found something "interesting" that can be used to provide navigation improvements, as measured by appropriate evaluation methods. We call the objects created by the application of various learning methods Navigation Compression Models (NCMs).

The idea and name arise from the principle behind learning. All learning creates an abstraction of the initial input data, which somehow represents the input data in some other representation. For example, a classification hierarchy over a number of input instances, is a compressed representation of that input. Similarly, our Navigation Compression Models are simply some representation of an abstract navigation space, determined as a function of selected attributes of a collection of user navigation paths on actual websites.

Our framework clearly decouples the offline data preprocessing and learning modules from the online recommendation module, as depicted in Figure 1. In this architecture, NCMs can be seen as the output of the offline learning module and as well as a plug-and-play component in the online recommendation module.

The recommendation module employs the same mechanism on different NCMs to generate dynamic recommendations for the user based on the pages the user has already visited. For this purpose, each NCM should be formulated as a function that generates recommendations based on a given path:

$$NCM = f(\text{path} \Rightarrow \text{recommendations})$$

where the recommendations can be as simple as the frequently visited pages, or as complex as the predictions of web documents that might satisfy the user's information needs.

For example, association rules can capture the co-occurrence relationships between different web pages based on users' navigation activities. An association-rule-based NCM can be applied for dynamic recommendations as follows: whenever the pages in the antecedent of the association rule have fully appeared in the user's current path, we recommend those "interesting" but unvisited pages in its consequence. As another example, a page cluster represents a

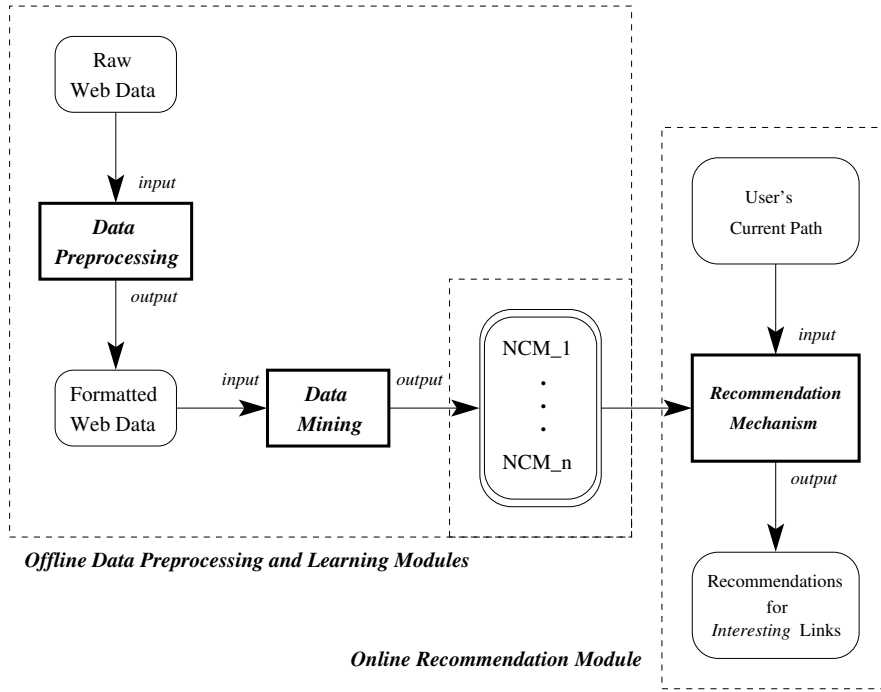


Figure 1: Architecture of NCM-based Framework for Recommender Systems

set of web pages which are similar to each other in terms of certain metrics such as usage patterns or page content. A cluster-based NCM can be applied as follows: if one or more pages in the cluster have been visited in the user’s current path, we recommend those “interesting” but unvisited pages in the cluster.

Note that different NCMs are not necessarily exclusive in the recommendation process. With a properly defined mechanism, the recommendations generated from different NCMs might be combined to provide recommendations with even higher quality (e.g., [Wei *et al.*, 2003]).

3 Dynamic Recommendation Mechanism

The dynamic recommendation mechanism determines how the recommendations are generated and presented to the user. In our case, the recommendations are generated by applying NCMs obtained from the learning process to the user’s current path. While the user navigates in the web site, the user path changes continuously, and the recommendations are updated along the way.

How recommendations are presented to the user is also an important issue. First, the number of recommended hyperlinks must be limited. As previously discussed, finding the useful page in a huge recommendation list could be more difficult and time-consuming than simply navigating without any recommendation. For this purpose, we use a specific parameter to represent the maximum number of recommendations that can be presented to the user in one screen, which is denoted as r_limit . The choice of r_limit reflects the user’s preference and should be application-driven.

Since we want to display the most important recommendations at the top of the recommendation list, the second issue of the dynamic recommendation mechanism is to determine the importance, or priority, of each recommended hyperlink. We determine this priority based on the following three criteria, which are applied in order:

1. The exact position in the user path where the recommendation is generated. More recently generated recommendations are given higher priority. This is based on an assumption that in most cases, the user interest at a specific point of time is more highly related to the documents visited close to that point.
2. The quality of the matching between the current user path and the NCM used to generate the recommendation. For example, for association-rule-based NCMs, this matching quality can be defined as length of the antecedent of the rule. For cluster-based NCMs, this matching quality can be defined as the number of common pages between the cluster and the user’s current path. Recommendations generated by NCMs with higher matching quality are given higher priorities. This determination is based on our assumption that the more information used for learning, the more accurate results we should obtain.
3. The significance of the NCM used to generate the recommendation. For example, for association-rule-based NCMs, this significance can be defined based on the support and confidence of the rule, e.g., as their Geometric Mean ($\sqrt{sup. \times conf.}$) or Harmonic Mean ($\frac{2 \times sup. \times conf.}{sup. + conf.}$). For cluster-based NCMs, this signifi-

cance can be defined as the similarity between the recommended document and the center of the cluster used to generate it. NCMs with higher significance values are considered more useful. Therefore, the recommendations they generate are given higher priorities.

Note that the idea of recommending only “relevant” pages can be implemented by NCMs or by the recommendation module. First, the NCMs can be trained to make predictions only on those pages that have previously been identified as content pages. Or the recommendation module can determine whether a page is a relevant page in real time based on the user’s current information needs, e.g., by matching the content of the page to the user’s search keywords. The former approach is employed in our experiments.

4 Simulation-based Evaluation

While we have various learning methods to create NCMs and a mechanism to generate recommendations using NCMs, a proper evaluation method is crucial to assess the improvement of the navigation. However, there has been little effort directed at this problem, and so far we know of no evaluation method which can evaluate navigation improvement quantitatively.

4.1 Evaluation Methodology

The basic idea behind the evaluation for navigation improvement is that we can compare the compressed navigation paths with the corresponding paths without compression. However, we can not expect to obtain both of these navigation paths from the same user without the previous navigation experience somehow affecting the later one. Such being the case, we can envisage an ideal experiment conducted as follows:

Suppose we have a group of user subjects with the same or similar educational background, similar interests, and the same level of web experience. Each subject is asked to fulfill a fixed number of tasks. Each task can be described as finding some specific information starting from a given entry point. Moreover, each task is randomly assigned to a fixed number of users such that half the users are helped with the dynamic recommender system, and half the users are not. In this way we can collect compressed navigation paths together with the corresponding uncompressed paths from different, but similar, users.

Such an experiment involves certain problems like user subject composition and task distribution, but these problems are not impossible to address. However, our objective requires a different approach: since intrusiveness and privacy issue has become a big concern in today’s WWW communities, we want to minimize the use of intrusive data collection, even for the purpose of evaluation. Based on this consideration, we propose a non-intrusive, quantitative evaluation approach, which is designed to use the commonly collected web log data for both training and testing, as shown in Figure 2. With respect to this idea, the evaluation procedure can be described as follows:

1. Transform original log data (both training and testing) into user sessions. We acknowledge that this work involves enormous challenges in heuristic identification of individual users, sessions, and “relevant” pages (or content pages).
2. Apply a selection of learning algorithms to the training sessions to obtain NCMs. These NCMs have the ability to predict the user’s possible interests and therefore can be used for making dynamic recommendations.
3. Apply the NCMs to the testing sessions through a simulation procedure and generate a new set of compressed sessions.
4. Finally, the value of the NCMs is determined by using a family of evaluation functions based on the number of traversal links saved and the potential cost of recommendations.

4.2 Evaluation Function

Measuring the efficiency of navigation is not a straightforward task. While this measurement can be perceived by humans through various kinds of visualizations [Niu *et al.*, 2003], we also require some quantitative measures which can be obtained automatically from the navigation trails with and without the help of dynamic recommendations. As noted above, these quantitative measurements are obtained by comparing the original and compressed sessions.

One possible approach to the development of such quantitative measurements is based on the number of hyperlinks traversed. Based on this idea, we propose a specific measure called *navigation improvement (NI)*, which indicates a quantitative improvement in the navigation efficiency of the compressed sessions over the original ones. Intuitively, *NI* can be computed as:

$$NI = \frac{N_{org} - N_{com}}{N_{org}} \quad (1)$$

where N_{org} is the number of traversal links in the original sessions, and N_{com} is the number of traversal links in the compressed sessions.

For example, suppose we have obtained an association rule $\{A \rightarrow \underline{D}\}$, where \underline{D} is a content page. Then an original session $S_{org} = \{A, B, C, \underline{D}, E\}$, where B and C are auxiliary pages and \underline{D} is the target page, can be compressed by skipping B and C to obtain $S_{com} = \{A, \underline{D}, E\}$. The navigation improvement for this session would be:

$$NI(S) = \frac{N_{S_{org}} - N_{S_{com}}}{N_{S_{org}}} = \frac{5 - 3}{5} = 40\%$$

However, this simple measure does not take into account the cost of dynamic recommendations. In an extreme example, suppose the user is provided with one hundred recommendations, and only the last one is useful in guiding the user to the next content page by skipping one auxiliary page. Such being the case, most users will probably consider this recommendation set useless because it will take them less effort simply browsing to the content page with a few more clicks rather than picking it up from the huge recommendation list.

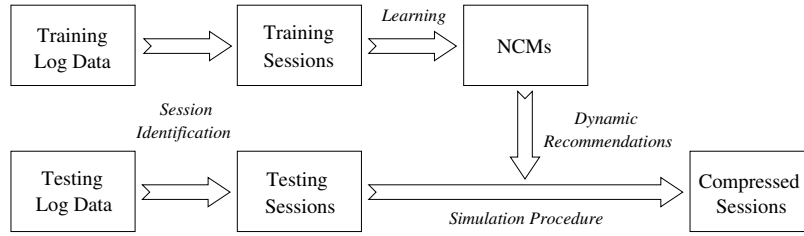


Figure 2: Evaluation Procedure

So another quantitative method could compute improvement by estimating the cost of the recommendations and subtracting it from the NI . The basic idea is that the more recommendations we provide to the user, the more cost we should subtract from the NI . Here we define a specific parameter to represent the cost of one recommendation, which is denoted as r_cost . So $\frac{1}{r_cost}$ recommendations will cancel out the benefit of one saved traversal link.

Note that the determination of r_cost actually reflects the user’s (or web designer’s) consideration on the tradeoff between size of the recommendation space and proportion of “interesting” information contained in that space. A larger recommendation space generally contains more “interesting” information, but also introduces more “uninteresting” information (or noise) into that space. Therefore, a higher setting of r_cost indicates that lower noise level is preferred, and a lower setting of r_cost indicates that the total amount of “interesting” information is a bigger concern than the noise level.

Based on the above idea, a *cost-based navigation improvement* (NI_c) can be computed as:

$$NI_c = \frac{N_{org} - N_{com} - n_r \times r_cost}{N_{org}} \quad (2)$$

where n_r is the number of distinct recommendations provided to the user.

In the previous example, given $n_r = 4$ and $r_cost = 0.1$, which means that during the session there were four distinct recommendations provided to the user, and ten recommendations will cancel out the benefit of one saved traversal link, then the cost-based navigation improvement of the session would be:

$$\begin{aligned} NI_c(S) &= \frac{N_{S_{org}} - N_{S_{com}} - n_r \times r_cost}{N_{S_{org}}} \\ &= \frac{5 - 3 - 4 \times 0.1}{5} \\ &= 32\% \end{aligned}$$

4.3 Simulation Procedure

Our evaluation is designed to obtain the compressed sessions by simulating recommendation-engaged user navigation behaviors on the original sessions, instead of collecting data from real web usage.

The user action model we use is simple, which assumes that the user will follow all the “correct” recommendations.

Here “correct” means that the recommended page is a content page to be visited in the session, and there is no other content page between the current position and the recommended page. Though our assumption on user action seems “optimistic”, the measured navigation improvement isn’t necessarily the best possible result. For example, in real usage the user may jump to any recommended page without such restriction that the content pages must be visited in a particular order.

An example is shown in Figure 3 which illustrates how our simulation mechanism works. This example uses association-rule-based NCMs, and each page is represented as a page ID (content pages in parentheses).

Note that content pages play an important role in this process. First, we recommend only those pages recognized as content pages in the training set. Second, we have to make sure that the compressed sessions keep all those content pages in the original sessions so that the compression actually makes sense.

5 Experiments

In our experiments we used a set of publicly available server access logs from the music machines web site (currently at “<http://machines.hyperreal.org/>”). These web logs are used by the “adaptive web sites” project [Perkowitz and Etzioni, 1998] and made available online at “<http://www.cs.washington.edu/ai/adaptive-data/>”.

The music machines web site provides a wide variety of information related to music equipments - including images, softwares, schematics, as well as tips and comments from musicians. In our experiments, we have collected six months of server access logs from the music machines web site, from October 1, 1998 to March 31, 1999.

The data preparation process we used is similar to that as described in [Cooley *et al.*, 1999]. The results show that each month the music machines web server generated approximately one million records and 40000 *useful* sessions ($1 < session\ length \leq 100$) with an average session length of approximately 7.43.

The identification of content pages is itself a difficult problem. In our experiments, we used the *Reference Length* method [Cooley *et al.*, 1999], and we applied a clustering algorithm to determine the cutoff viewing time (assuming that any content page has a longer viewing time than any auxiliary page) instead of assigning an arbitrary value to it.

In each case of the experiments, the previous one-month

Original Session : 1, 2, (3), 4, (5), 6, (7), 8, (9), 10
 Recommendation Content Page Set : (5), (7), (9)
 Test Content Page Set : (3), (5), (7), (9)
 NCM : (a) 1 → 9
 (b) 2 → 5
 (c) 1, 2 → 4, 7
 (d) 1, 3 → 9 (*significance* = 0.5)
 (e) 2, 3 → 5 (*significance* = 0.8)

Navigation Path	Recommendation List	NCMs Applied
1	(9)	(a)
1, 2	(7), (5), (9)	(c), (b)
1, 2, (3)	(5), (9), (7)	(e), (d)
1, 2, (3), (5)	(9), (7)	
1, 2, (3), (5), (7)	(9)	
1, 2, (3), (5), (7), (9)	(null)	
1, 2, (3), (5), (7), (9), 10	(null)	

Compressed Session : 1, 2, (3), (5), (7), (9), 10

Figure 3: Simulation Procedure Example

logs were used as the training set, and the later one-month logs were used as the test set. We report experiments with two kinds of NCMs applied exclusively, one from association rules and another one from co-occurrence oriented page clusters generated using PageGather [Perkowitz and Etzioni, 1998]. A common limitation of these two kinds of NCMs is that they can only recommend those pages which have been previously visited in the training web logs. In all the cases, we use $r_limit = 10$.

The experimental results are shown in Table 1. In this table, each navigation improvement (both NI and NI_c) is computed on those sessions with at least one content page.

The results show that cluster-based NCMs perform much better than association-rule-based NCMs when there is no recommendation cost or when the recommendation cost (r_cost) is set low. However, setting a higher r_cost will reduce the NI_c of cluster-based NCMs more than the association-rule-based NCMs.

Further investigation showed that this is because cluster-based NCMs tend to generate many more recommendations than association-rule-based NCMs. In other words, while cluster-based NCMs generate more “correct” recommendations, at the same time they also generate many more “incorrect” recommendations. For example, in a one-month experiment 500 clusters generate an average of 188000 recommendations (in which 15300 are “correct”), while 2000 rules generate an average of only 112000 recommendations (in which 10200 are “correct”).

Note that our experimental results are indeed determined by a number of heuristics and parameter settings, including the heuristics for user identification, session identification, content page identification, and the settings of r_cost and r_limit . Different choices on these heuristics and parameter settings can undoubtedly lead to different results. The appropriate choices of these heuristics and parameters should be task-determined, and in most cases can only be obtained empirically.

6 Summary and Conclusions

We have presented a general framework for navigation recommender systems based on the application of web usage mining, and have developed the idea of Navigation Compression Models (NCMs) to represent the results of learning “better” navigation paths from web usage data.

We proposed a dynamic recommendation mechanism using different NCMs, a simulation-based approach to non-intrusively evaluating the value of the recommender system as well as the effectiveness of NCMs, and a family of evaluation functions which measure the navigation improvement based on the number of traversal links and the potential cost of recommendations.

There are several directions for extending this work. The identification of content pages is still a challenging problem. In addition to the NCMs used in this paper, other kinds of NCMs created with different learning methods can also be applied to examine their effects on navigation improvement. Also, it will be interesting to examine the performance of incorporating multiple NCMs into the recommendation process. There could be other valuable evaluation functions, and the recommendation mechanism can be further refined, e.g., giving larger penalties to more important recommendations by assigning higher r_cost to recommendations at the top of the recommendation list.

Note that the dynamic recommendation mechanism used in our experiments makes recommendations *whenever it can*. But we can imagine a more “intelligent” mechanism that makes recommendations *only when the user needs help*. Such a functional change involves the determination of the user’s “lostness moment”. There has been some interesting work in this problem [Gwizdka *et al.*, 2004], which might help better understand user navigation and lead to a more “intelligent” recommender system.

The experiments we have conducted used heuristically identified content pages and a simplified user model. We hope to collect further “real” data with user-labeled content pages and recommendation-enabled user access logs, and use that

Table 1: Experimental Results

Association-rule-based NCMs					
Training Data	Test Data	#Rules	NI	$NI_c (r_cost = 0.05)$	$NI_c (r_cost = 0.08)$
10/1998	11/1998	2008	4.86%	2.88%	1.69%
11/1998	12/1998	2015	4.69%	2.60%	1.35%
12/1998	01/1999	2057	4.43%	2.39%	1.17%
01/1999	02/1999	2077	4.19%	2.31%	1.18%
02/1999	03/1999	2062	4.73%	2.63%	1.37%
mean±standard deviation			4.58±0.27%	2.56±0.22%	1.35±0.21%

Co-occurrence oriented Cluster-based NCMs					
Training Data	Test Data	#Clusters	NI	$NI_c (r_cost = 0.05)$	$NI_c (r_cost = 0.08)$
10/1998	11/1998	491	7.50%	4.14%	2.12%
11/1998	12/1998	498	7.60%	4.21%	2.18%
12/1998	01/1999	535	7.29%	3.88%	1.83%
01/1999	02/1999	509	7.12%	3.95%	2.05%
02/1999	03/1999	494	7.39%	3.85%	1.73%
mean±standard deviation			7.38±0.19%	4.00±0.16%	1.98±0.19%

data to validate the various methods and heuristics used in our experiments.

Acknowledgments

Our work is supported by the Canadian Natural Sciences and Engineering Research Council (NSERC), and by the Alberta Ingenuity Centre for Machine Learning (AICML).

References

- [Baraglia and Silvestri, 2004] R. Baraglia and F. Silvestri. An Online Recommender System for Large Web Sites. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence (WI'04)*, pages 199–205, September 2004.
- [Cooley *et al.*, 1999] R. Cooley, B. Mobasher, and J. Srivastava. Data Preparation for Mining World Wide Web Browsing Patterns. *Journal of Knowledge and Information Systems*, 1(1), 1999.
- [Gwizdka *et al.*, 2004] J. Gwizdka, I. Spence, H. Takeshita, and K. Seergobin. Quantifying Navigation, 2004. Exhibition Poster, CASCON 2004.
- [Mobasher *et al.*, 1999] B. Mobasher, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. Technical Report TR99-010, Department of Computer Science, Depaul University, 1999.
- [Niu *et al.*, 2003] Y.H. Niu, T. Zheng, R. Goebel, and J.Y. Chen. WebKIV: Visualizing Structure and Navigation for Web Mining Applications. In *2003 IEEE/WIC International Conference on Web Intelligence (WI'03)*, Halifax, Canada, October 13-17 2003.
- [Olston and Chi, 2003] C. Olston and E.H. Chi. Scent-Trails: Integrating Browsing and Searching on the Web. *ACM Transactions on Computer-Human Interaction*, 10(3):177–197, September 2003.
- [Perkowitz and Etzioni, 1998] M. Perkowitz and O. Etzioni. Adaptive Web Sites: Automatically Synthesizing Web Pages. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, 1998.
- [Resnick *et al.*, 1994] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.
- [Wei *et al.*, 2003] Y.Z. Wei, L. Moreau, and N.R. Jennings. Recommender Systems: A Market-based Design. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS'03)*, pages 600–607, Melbourne, Australia, 2003.
- [Zhu *et al.*, 2003] T.S. Zhu, R. Greiner, and G. Häubl. An Effective Complete-Web Recommender System. In *The 12th International World Wide Web Conference (WWW2003)*, Budapest, HUNGARY, May 2003.