Agent-Based Recommender Systems

Olli Niinivaara

Helsinki 17.5.2004
Sofware Agent Technology Course Paper

UNIVERSITY OF HELSINKI
Department of Computer Science

University of Helsinki
Department of Computer Science

Sofware Agent Technology
Olli Niinivaara: Collaborative Social Recommender Agents
Report, 48 pages, 3 index and 8 appendix pages
May 2004

The main possibilities and challenges of agent-based recommender systems are examined.

Key words and terms:
software agents, recommender systems, matchmaking, user modeling, reputation, trust

Contents

1. Introduction

Software agents have been a promising paradigm for long. Still, the killer applications are rare. Most of research is quite technical, producing more or less general models, theories and platforms, but large-scale adoption of these techniques still awaits. Here software agent technology is considered from a top-down view: the approach is to define application possibilities and challenges first, and leave the particular implementation techniques as secondary. The goal is to present a real application area for the more technically-oriented to tackle and to assure the sceptics of the practical usefulness of software agent technology.

The problem to be dealt with solved is a general selection problem. When there are many resources (be it information, products, services, other people, or whatever), one has to make decision which one(s) to choose. In this paper possibilities for automatic assistance in the selection process by agent-based recommender systems are examined. The main tasks of recommender systems are to help users to:

- **evaluate resources** (e.g. by gathering and aggregating opinions conserning the resource)
- **find evaluations of a resource** (e.g. by recommending experts of the field who can help)
- **evaluate the evaluaters** (e.g. their expertise, reputation and trustfulness)

Most information agents[Klu99] help people to find more information, thereby only increasing the ever-present information overload. Recommender systems can be seen as an automatic remedy for this problem. Agent technology becomes invaluable by appreciating the facts that we want the systems to take personal preferences into account, we want the systems to infer and intelligently aggregate opinions and relationships from heterogenous sources and data, we want systems to be scalable, privacy-protecting, open systems and we want to get the recommendations with least possible work on users' behalf.

In chapter 2 the selection problem is more explicitly defined, some potential application domains are presented and recommender systems are reviewed. Chapter 3 is about user modeling as the basis for personalized recommendations. In chapter 4 the benefits of distribution are discussed, multi-agent systems are presented as the enabling technology and similarities between agent matchmaking and social matchmaking are noted. The social dimension is brought into picture in chapter 5, where the possibilities of social network modeling and analysis are considered and the potential in joining reputation systems

research efforts with reputation research in multi-agent systems is emphasized. In chapter 6, a short, but important pop in to an even wider perspective is made. Chapter 7 presents a state-of-the-art example of research utilizing some of the key ideas. Chapter 8 is for conclusions.

2. Resource selection

2.1 The selection problem

Let's label as resources those available entities that are potentially useful in some situation. Examples of resources include people, organizations, documents, information channels, products and services. To use a resource, one must find and identify it. One of the most efficient ways of doing this is via Internet. Found resources should usually be evaluated and selected before use. This might be for example because more resources are found than could be used, because access to the resource takes time, money or involves other barriers, or because selection of a wrong resource might be harmful. Currently much of the Internet's potential is lost because the evaluation and selection of found resources cannot be done as efficiently as the searching. The openness of solutions worsens the situation. If any one can participate in whatever way one likes, the overall quality of available resources will be low. As a consequence, finding and selecting the best resources becomes harder. When the visibility of quality resources diminishes, it gives an incentive for quality-producing actors to use some other system, which degrades the usefullness of the system even more. The obvious solution would be to restrict participation or usage. Here a more sophisticated solution is sought for, namely using information technology (agent-technology, to be precise) to assist in filtering the quality resources out from the mass.

A resource can be directly evaluated by inspecting it and by trying it. These operations probably give the best estimates, but they may be very time-consuming or expensive, need some special skills or not easily carried out, as in the case of persons as resources. Therefore means of indirect evaluation are needed. A resource can be indirectly evaluated by inspecting descriptions of the resource or descriptions of usage of the resource. This inderection brings two advantages. First, while the resources themselves might not be directly available on the Internet, descriptions of them usually can be. Second, via descriptions of usage, one can learn from experience and especially thereby avoid costly experiences. But adding a level indirection carries also costs. One must search not only resources, but descriptions of them, too. The material to be handled grows considerably, because there can be several descriptions for one resource. Worse still, the descriptions themselves must be somehow evaluated. In addition to the content of the resource, one must now consider also the trustability of the description. Again direct inspection is not always feasible, which creates a need for yet another layer of indirection, and so on.

Automation can be used to manage the costs of indirection. Use of descriptions makes the resources visible for automatic handling. With the use of some heuristics, many kinds of information resources that include references to other resources can be used as useful sources for evaluation. Computational power makes it feasible to manage huge amounts of descriptions with interrelationships and to fruitfully analyze the relationship networks.

Let's call people, organizations and communities collectively as actors. Actors form a special class of resources. If there is not a particular kind of resource available, it is useful to find an actor that can produce the needed resource. Especially when there is no descriptions of a resource available, it is useful to find an actor that can produce the needed description. Other way round, resources and descriptions can normally be quite efficiently evaluated and their trustability assessed by checking their producer. Because of these interesting properties, this report especially concerns the automatic assistance possibilities for finding and evaluating actors and taking actor information into account when evaluating other kinds of resources.

2.2 Recommender systems

Recommender system is a term that very well describes systems that take advantage of the existence of actor information in order to implement automated selection assistance. Therefore this report is all about recommender systems. One very recent and simple definition of recommender systems is as follows: "Recommender systems use the opinions of members of a community to help individuals in that community identify the information or products most likely to be interesting to them or relevant to their needs"[Kon04]. The term recommender systems has become widely used due to the publication of the Communications of the ACM special issue in March 1997 on the topic[Com97]. There Resnick and Varian suggested the term "recommender system" as a replacement for earlier used "collaborative filtering". They define recommender systems by giving a short description of typical recommender system, which goes as follows: "In a typical recommender system people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients. In some cases the primary transformation is in the aggregation; in others the system’s value lies in its ability to

make good matches between the recommenders and those seeking recommendations"[ReV97].

By looking at definitions of the design space of recommender systems[ReV97, TeH01], there seems to be following items common to all recommender systems:

Resources: The targets of the recommendation process.

Recommenders: Those entities that radiate opinions about resources. In practice recommenders are actors and in principle they could also be artificial agents (see below). Recommenders may also be resources at the same time.

Descriptions: Those information resources that include opinions. The descriptions may explicitly be designed for expressing opinions, or they may include opinions about resources implicitly, in which case they must be inferred.

Preferences: Recommendation seeker's position towards resources.

Algorithms for Computing Recommendations: The system's means for automatically evaluating resources by using descriptions and preference information.

Recommendations: The concrete results of the evaluation process for the recommendation seeker. The recommendations may be presented for example by filtering out resources, informing about new resources, displaying recommendations along the resources, or ordering resources, simply to an an ordered list, or perhaps by presenting sophisticated 2D or 3D visualizations[1,2].

It is not always easy to decide, if some system is a recommender system or not. As an example, let's consider Google<Goo>, a world-famous search engine. URLs that are found by a crawling process are resources. Those pages behind URLs that include references to other information resources via URLs are descriptions. Because the inner workings of Google are not public, one can only guess that top level URLs (server addresses) act as recommenders[Cla03][3]. Algorithm for computing recommendations exists [Pag98, Cla03]. Recommendations are presented by ordering search results. Only thing that is completely missing is the preferences - recommendation seekers have no infuence to what is recommended to them and how it is done. All in all, Google is a recommender system, but not a full-blown one. In practice it will often be a matter of opinion, if some system can be classified as a recommender system or not. For example, many systems are nowadays classified as reputation systems[Res00], apparently when the key point is, that the opinions concern other actors. Altough these systems routinely miss the concept explicit user

preferences (ie. having only one, global reputation estimate), these systems may be tought as one kind of recommender systems, and probably one the of most important, too.

Now, in the domain of recommender system, this report especially deals with the use of agents as an enabling technology. Probably the most efficient solution would be to use agents as recommenders. This would mean that agents could either inspect the resources or try them by themselves. However, existing quality measures and evaluation algorithms only work in some limited domain. For example, agents could be used to check the syntactic and some of the semantic validity of some information resources and to test some properties of executable resources, as programs and digital services. An example of this kind of solution is presented by Vivacqua[Viv99]. There the agents can for themselves figure out the expertise of people by exploiting the fact that people's work is to produce java code, which is easily evaluated automatically. Because the replacement of human opinions with artificial opinions in a general case has a problem space of its own, in this report the possibility of using agents as original opinion sources will not be handled.

There's anyway plenty of room for application of agent technologies. It seems that there is a natural fit between agent technologies and recommender systems. Some research areas are important to both, such as machine learning, adaptivity support, user modeling and filtering techniques. For agent technologies, recommender systems offer a practical and important application domain with useful concepts. For recommender systems, agent research offer ways to manage autonomy, pro-activity, distribution, reputation and trust, etc. These possibilities will be examined in following chapters.

2.3 Application domains

Selection between resources is important in all areas of life. Therefore the potential of recommender systems is almost limitless. To get a view of the possibilities and to review some related work, three application domains are presented. These are the domains of knowledge management, socialware, and e-commerce. At the end, the potential of open systems as a long-term goal for recommender systems is discussed.

Knowledge management means the handling (gathering, storing, fetching, etc.) of organization's information assets with the help of information technology. Important features of this domain include the importance of actors as resources and the restrictedness of the environment.

Importance of actors (mainly co-workers in this case) comes from the fact that only a small fraction of organization's knowledge can be codified into (digital) information resources. The majority of knowledge remains in the heads of the people as their personal views and insights and their capabilities and skills. Also knowledge of others' influential capabilities, their positions in social networks and their connections inside the organization and to outside is not easily managed with ordinary knolwedge management methods. Nardi et al.[Nar02] report the following tasks that the workers mostly needed assistance in: remembering the identities of the people in their social networks, particularly those who are important but are contacted infrequently; remembering connections between different people in the network; remembering or recording details about contacts, including current whereabouts and activities; remembering which documents had been exchanged with whom and when; remembering task status, such as that a report has to be finished by a certain date; obtaining awareness information for distant contacts, such as whether they are available for a phone call; and comfort using multiple communication media. Assistance for managing relations between people in an organization is therefore an important recommender system application area.

The restrictedness of organizational environments means that all the components like programs, communication protocols and data formats can be controlled. In addition, users can be teached and made to use the system. Because organization's work methods and focus areas are also restricted, it is more easy to build systems that can analyze the available data and make intelligent inferences. These features togeteher makes knowledge management applications easier to build, test and use in real contexts than general Internet applications, which makes them an attractive application alternative for agent-based recommender systems.

Hattori et al. call socialware multi-agent systems that assist in various social activities on network communities[Hat99]. Socialware can be seen as an instance of recommender systems, where the purpose is to facilitate the selection of actors, in this ccase both people and communities. Hattori et al. list three major issues for

support. The first issue is to link people with others and with communities, that share similar interests. The second issue is to support smooth communications, including support for visualizing and sharing common contexts, as well as identifying the flow of communications. Third issue is to find what relationships people have. Socialware concepts have been developed further for example in Shine[Yos00], which is a multi-agent platform for network community support systems. Yet another relevant example are Wang's self-organizing communities formed by middle-agents[Wan02]. Areas closely related to socialware are communityware, in which the issue is to make people aware of each other[SKX98], and social networking systems, as Friendster[Boy04],<Fri>, where the issue is to find new people by explicitly modeled relationship networks[4]. Another interesting example is Sugawara et al.'s system to support job searching in a teleworking community[SYR03], which provides a list of suitable job candidates which are ranked as the best possible business opportunities for the teleworker.

E-commerce is today the most important application area of recommender system techniques. Buyers have to decide between products (or services). Often the situation is such that the products cannot be tried or even inspected before buying. Therefore there is a clear need for systems that make the evaluation easier. On the other hand, sellers have their own motivations for promoting recommender systems - they would surely sell more, if they could recommend needed products for customers. Probably the most famous example of a system with recommending features is the eBay<eBa>, which attributes its high rate of successful transactions to its reputation system, the Feedback Forum. After a transaction is completed, the buyer and seller have the opportunity to rate each other (1, 0, or -1) and leave comments. Each participant has his running total of feedback points attached visibly to his screen name, possibly a pseudonym. Yahoo! Auction<Yah>, Amazon<Ama> and other auction sites feature reputation systems like eBay’s, with variations such as a rating scale from 1-5, or using several measures (friendliness, prompt response, quality product, etc), or averaging rather than totaling feedback scores. Bizrate.com<biz> rates registered retailers by asking consumers to complete a survey after each purchase. So-called “expert sites” (e.g. Expertcentral<Exp>, Askme<Ask>) provide Q&A forums in which experts provide answers for questions posted by clients in exchange for reputation points and comments. Product review sites (e.g. Epinions<Epi>) offer rating services for product reviewers — the better the review, the more points the reviewer receives. Still, very few of the systems are real recommender system using both user preferences and

others' opinions. Even fewer are built upon agent technology. Therefore there's a huge potential in the commercial domain for more sophisticated solutions.

Most recommender systems so far have been monolithic and more or less ad-hoc systems designed for some restricted domain. This is understandable, as the more restricted the domain is, the easier it is to design and implement something useful. However, recommender systems are systems with network effects: the more resources, users, recommendations, etc. available, the more useful the system becomes. The real world is not divided into domains, but connections and relations can be found everywhere. Therefore the real challenge and ultimate goal is to design recommender systems as open systems. Towards this goal, there should be general recommender system components, which would act as building blocks offering functionality independently of any particular domain and open recommender platforms, which would allow interoperation between implementations and subsystems. In the following chapters, openness is taken as a goal and domain-dependent issues are ignored. Of course this does not reduce the applicability of ideas in any particular application area.

3. User modeling

3.1 Why user modeling

In order to make useful recommendations, the recommender system must know something about the particular user's preferences and about other recommenders opinions. Therefore there's need for two kinds of user modeling. Of these, the term user modeling commonly means only the modeling of the local user. To distinguish the two, the modeling of other users will be called actor modeling in following.

(Local) user modeling is needed, because different persons need different kinds of resources. Therefore the aim of user modeling is to personalize the system so that it is useful for different users. The result of modeling may be called a user stereotype or a user profile. Stereotype tells user's type in some predefined taxonomy (most common example probably being the novice user - normal user - expert user classification), whereas profile is a more detailed description with several attributes with varying value ranges. For user modeling in a recommender system a simple stereotype seems not to be enough, because of the great variety of quality preferences and available resources. For actor modeling the stereotype seems to be an adequate solution, though. This is because there's probably not enough information available to build a detailed actor profile anyway and the final recommendation is usually determined as sum of many actors' opinions, in which case great detail for one actor is not so important. As an example, Ogata and Yano's PeCo agent can offer following five stereotypes of other users[OgY99]:

Collaborator: The collaborator is a user who usually accepts the request during this system use. The collaborator is often an expert about the request.

Semi-collaborator: The semi-collaborator is a user who potentially has the capability for cooperation about the request. It is assumed that a semi-collaborator receives the answer from others rather than accepting requests.

Mediator: The Mediator is the user who usually forwards the request to his/her friends.

Requestor: The requestor is a user who asks a question and s/he becomes a stating point of exploration of social networks.

Non-collaborator: The non-collaborator is a user who almost always rejects the requests.

Unknown user: If a user has never received or sent a request, the user is unknown for the system.

Here's another example of actor stereotyping[GPD03]:

An annoying user: one whose suggestions are not accepted by the majority of users

A passive user: one that does not review nor evaluate any document, but reads the supplied documents

An active user: one that participates in the evaluation workflow. It may be:

An active reviewer: one that provides an evaluation on a document

An active suggestor: one that frequently provides documents to the community

The easiest way to model a user is to explicitly ask about her preferences. Unfortunately this doesn't seem to work in practice. Users preferences may and do change. Therefore it would take some work to keep the profile up to date manually. Users might not even be capable of explicitly stating their preferences by setting some attribute values. It is commonly agreed, that user profiles should be built and maintained more or less automatically. One interesting idea is to design a generic user modeling shell, which would offer user modeling services to all kinds of applications. Kobsa states the following tasks as relevant for generic user modeling shells[Kob01]:

- the representation of assumptions about one or more types of user characteristics in models of individual users (e.g. assumptions about their knowledge, misconceptions, goals, plans, preferences, tasks, and abilities)

- the representation of relevant common characteristics of users pertaining to specific user subgroups of the application system

- the classification of users as belonging to one or more of these subgroups, and the integration of the typical characteristics of these subgroups into the current individual user model

- the recording of users' behavior, particularly their past interaction with the system

- the formation of assumptions about the user based on the interaction history

- the generalization of the interaction histories of many users into stereotypes

- the drawing of additional assumptions about the current user based on initial ones

- consistency maintenance in the user model

- the provision of the current assumptions about the user, as well as justifications for these assumptions

- the evaluation of the entries in the current user model, and the comparison with given standards

There's still many problems with generic systems, and to date, systems have been research prototypes (see, for example DOPPELGÄNGER[Orw96]). Therefore recommender systems probably must incorporate their own modeling methods in the foreseeable future.

the Importance is user modeling for recommender systems is reflected by the fact, that in article by Montaner, López and de la Rosa,[MLd03] user modeling techniques are taken as the main classification criteria for recommender agents. There 8 different dimensions for user modeling are identified. Implementations of user modeling vary in:

- User profile representation

- Information filtering method

- Initial profile generation

- User profile-item matching technique

- Profile learning technique

- User profile matching technique

- Relevance feedback

- Profile adaptation technique

3.2 Implicit feedback

the main challenge of user modeling is to build a useful profile without disturbing the user. Implicit feedback denotes the techniques of modeling the user by unobtrusively watching the user interactions with the system and inferring the profile out of them (these are the task of a user modeling agent).

There exists some work which tries to present and classify the main methods. Nichols[Nic97] lists following actions as potential sources for inferring preferences (in the list, resources mean both actual resources or resource descriptions):

Purchase: buys a resource

Assess: evaluates or recommends a resource

Repeated Use: e.g. multiple check-out stamps

Save / Print: saves resource to personal storage or prints it out

Delete: deletes a resource

Refer: cites or otherwise refers to resource

Reply: replies to resource

Mark: add to a 'marked' or 'interesting' list (for example bookmarking)

Examine / Read: looks at the resource

Consider: looks at abstract (or recommendations)

Glimpse: sees title or surrogate of a resource in list

Associate: returns a resource in search but never glimpses

Query: association of terms from search queries

One of the best and most up-to-date reviews of the area is the Kelly and Teevan's study[KeT03]. There a classification of implicit feedback methods is given, building upon the work by Oard and Kim[OaK98]. Classification is based on two axes; behaviour category and minimum scope. Behaviour category refers to underlying purpose of the observed behaviour using four types: *Examine*, *Retain*, *Reference*, and *Annotate*. Minimum scope refers to the smallest possible scope of the resource being acted upon using three types: *Segment*, *Object*, and *Class*. This classification yields following classes (some of the resulting classes do not have examples, and are omitted):

Examine Segment: View, Listen, Scroll, Find, query

Examine Object: Select

Examine Class: Browse

Retain Segment: Print

Retain Object: Bookmark, Save, Delete, Purchase, Email

Retain Class: Subscribe

Reference Segment: Copy-and-Paste, Quote

Reference Object: Forward, Reply, Link, Cite

Annotate Segment: Mark up

Annotate Object: Rate, Publish

Annotate Class: Organize

Create Segment: Type, Edit

Create Object: Author

In addition to these, there exists numerous other possibilities. For example, Claypool et al. talk about capturing the number of mouse clicks, the time spent moving the mouse, the number of clicks on the horizontal and vertical scroll bars and the time spent and number and duration of pressing some special action keys, as page up or up arrow. They even mention the possibility of using external interest indicators, as heart-rate, perspiration, temperature, emotions and eye movements[Cla01]. Another source of indicators worth considering is the use of context information, as user's (geographic) position, environment, point of time and date, the current task, etc.[ByC01, LaN92, SAW94]. These sources may become more important in the future by the advent and popularization of mobile devices and agents. also worth considering are the possibilities of using some searching techniques, like the Just-In-Time paradigm [BuH99, RhM00] and Reconnaissance Agents [LFW01] also for implicit feedback and evaluation purposes. All in all, this is a quite new research area having much promise and much challenges. An example of new developments is the idea of combining user profile data via the use of ontologies[Gim02].

When it comes to actor modeling, implicit feedback methods do not seem like bearable solutions, because of the privacy issues involved. It just is not a good idea to publish any information about actors, which is gathered by automatic surveillance techniques, at least without asking permission.

4. Distribution

4.1 The needs for distribution

Most of the implemented recommender systems are either stand-alone or client-server systems. Stand-alone systems can only infer the recommendations by inspecting some existing information source, such as USENET, e-mail messages or www pages. Because these sources weren't originally designed for inferring recommendations, possibilities for stand-alone recommendation are very limited. There is no possibility for explicitly stating opinions and no possibilities of changing information (actor descriptions, user preferences, relationship information, recommendations, etc.) between agents. Client-server systems usually have other limitations, as being useful only for some definite purpose, being closed, being proprietary (the recommendations algorithms are not publicly known), being controlled by some stakeholder naturally having some selfish motivations, etc. But there are some even more fundamental reasons for abandoning stand-alone and client-server systems in favour of distributed systems. Foner names the fundamental problems of centralized architectures in his seminal article describing Yenta, a multi-agent matchmaking system[Fon97]:

- Scaling a centralized architecture to large numbers of users is difficult; in systems which must correlate user interests, straightforward approaches to this problem generally require a quadratic-order matching step somewhere.

- If the system requires either high availability (due to constant demand for its services) or high trustability (because it handles potentially sensitive information, such as personal data), a centralized server provides a single point where either accidental failure or deliberate compromise can have catastrophic consequences.

On a more general level, Sycara et al.[Syc96] on their part discuss about following advantages of decentralized systems: the information sources may be already distributed; it would be wasteful to replicate agent information gathering and problem solving cpabilities for each user and each application instead of sharing them when needed; existing information services, whose implementation, query language and communication channels are beyond the control of user applications, can be more easily incorporated in problem-solving; agents can interact flexibly in new configurations "on demand"; the system is able to degrade gracefully even when some of the agents are out of service temporarily,; cross-

validation of data and results is made possible; and legacy systems can be utilized by using wrapper agents.

While centralized architecture might be acceptable in some restricted domain, as in certain knowledge management, e-commerce applications, it cannot be taken as basis for open systems. Distributed architecture allows creation of subcommunities for recommending among certain actor groups or recommending on certain resource types, at the same allowing grand-scale interconnection, thus enabling network effects to build up. Distributed, open architecture also allows users to design and implement their own recommending schemes and recommendation algorithms which can interoperate with other customized systems. For these reasons, techniques for distribution are examined next by introducing the area of multi-agent systems.

4.2 Multi-agent systems

Recommender systems have coined and developed by the old and established information research community. Therefore the connections to agent technology have not always been advertized very much, probably sometimes not even been noticed. Often it has been up to system designers and reviewers to decide, how agent-based a system is. Recommender systems incorporate techniques as inferring user preferences and smart reasoning based on the available data[5], which are key requirements for pro-active, autonomous operation for achieving intelligent user assistance. The connections are even clearer, when one keeps in mind, that information agents are described as autonomous computational software entities that are especially meant for (1) to provide a proactive resource discovery, (2) to resolve information impedance of information consumers and providers, and (3) to offer value-added information services and products[Klu99]. The heterogenity of approaches in the field is at least partially a result of the inherent problematics of defining agenthood in the first place. However, the importance of agent technology is fully revealed, when one starts to build distributed recommender systems with the aid of multi-agent systems (MAS) methodology. Then the agent-like capabilities can no longer be embedded into the system, but system components themselves appear as first-class agents.

Multi-agent systems have their history as a branch of distributed artificial intelligence[6]. The focus is not so much on working details of one particular agents, but on the interplay

between agents; how to find and connect agents, how to distribute the system, how to communicate, how to control the processes without centralized decision-making, how to achieve coordination and co-operation, etc. At least four reasons can be identified for adopting multi-agent systems methodology in recommender system design:

-By creating one or more agents to represent an actor, actors can be brought into the system as first-class entities. As a result, some MAS methods become almost immediately applicable. Very important issue is the close relation between agent matchmaking and actor matchmaking, which is the issue of the next chapter.

-MAS research has produced general mechanisms for distribution. By adopting these, recommender systems designers can concentrate on more high-level issues and take the underlying robust communication infrastructure for granted.

-Multi-agent platforms, as the FIPA platform, allow sophisticated dialogues, as bargaining, negotiation and argumentation to be held between agents. These would act as superb data sources for inferring opinions and recommendations.

-The issues of trust and reputation have not been traditionally considered in the recommender systems research, but are of utmost importance in open recommender architectures. On the other hand, these concepts have received quite a lot of attention in the MAS community. This reason alone would be enough to seek the synergies between MAS and recommender systems. The challenges and possibilities are further considered in chapter 5.2.

4.3 Agent matchmaking

Altough we are focusing on recommender systems and therefore ultimately interested in social matchmaking, ie. recommending useful actors, the area of agent matchmaking is directly relevant. First, by using agents as representatives of actors, the matchmaking of actors and agents become tasks that support each other and, second, before matchmaking via reasoning mechanisms, in distributed system there's always first the problem of even finding and identifying potential resources.

The coordination of agent societies is usually realized by using middle-agents. Middle-agents are agents that help others to locate and connect to agent providers and services. As such, they can be seen as the corner stones of a distributed agent infrastructure, upon which

recommender systems (let's say) can be built. Klusch and Sycara[KlS01] name as the core skills of any middle-agent the following:

-providing basic mediation services to the considered agent society

-coordinating these services according to given protocols, conventions, and policies

-ensuring reliable service mediation in terms of leveled quality of services as well as trust management within and across multiagent system borders

They also state, that in open environments in the Internet a middle agent has to provide basic mediation services for the:

-processing of agent capability and service descriptions

-semantic interoperation between agents and systems

-management of data and knowledge

-distributed query processing and transactions

One example of an agent matchmaking infrastrucure is the RETSINA[Syc99a, Syc99b, KlS01]. The RETSINA is being used to develop distributed collections of intelligent software agents that cooperate asynchronously to perform goal-directed information retrieval and information integration in support of a variety of decision making tasks. System mediation basically relies on service matchmaking. RETSINA uses an agent capability description language called LARKS (Language for Advertisement and Request for Knowledge Sharing). Application domain knowledge in advertisements and requests is specified as local ontologies written is a specified concept language ITL or by using WordNet. LARKS is fairly expressive and capable of supporting automated inferences. The implemented matchmaking process for LARKS specifications employs different techniques from information retrieval, AI and software engineering techniques are used to compute the syntactical and semantic similarity among advertized and requested agent capability descriptions. There's a matching engine that contains five different filters for (1) keyword-based context matching, (2) TF-IDF based profile comparison, (3) concept-based similarity matching, (4) type-inference rule-based signature matching, and (5) theta-subsumption based constraint matching of finite Horn clauses.

5. The social dimension

5.1 Social networks

The importance of computers as communication devices and the importance of communication on the whole increase together, probably partially due to some positive feedback loops. For an individual, it means that the signicance of social abilities and social connections increase. The importance is well described by the phrase: "it’s not what you know, it’s who you know". For organizations it means increasing significance of both differentation (e.g. distribution and specialization) and integration (e.g. coalition forming) - moving towards the information society and increasing globalization, for short[7].

Sociologists call the connection potential of individuals and communities social capital[Col88]. The term refers to the network position of the actor and consists of the ability to draw on the resources contained by members of the network. Basically, the more mappings a person has in the social network and the more mappings these people have, the more knowledge, influence, and power the original person will control. Social capital can have a substantial influence on a person’s life, affecting such aspects as job searches and potential for promotions. Social capital can attributed to one individual or to a bigger community, such as an organization. As a community-level attribute, it can be observed for example as social relations and networks of the community, widespread social trust, or norms of reciprocity. Individual social capital includes both intrinsic abilities (as charisma) and the result of individual social capital investements (as status). As these are practically indistinguishable, these can be bundled together. So, the properties constituting individual's social capital include things like publicity, influence, charisma, credibility, status, position, power, centrality, visibility, trustability, recognition and popularity.

As the significance and complexity of connections grows, it seems natural to seek computer support for managing them. Resnick uses the term “sociotechnical capital” to refer to productive resources that inhere in patterns of social relations that are maintained with the support of information and communication technologies[Res04]. From a sociotechnical capital point of view, a system should enable users to:

-share and exchange resources

-create connections

-form interest groups

-find other people and experts

-evaluate each other

-utilize others' knowledge

-understand relations to each other

-coordinate communication processes and interdependent actions

It is indisputable, that recommender systems will be one of the key "sociotechnologies". From the above list, all of them (maybe excluding the last one) can be taken as possible tasks for recommender systems, as it all comes down to selecting who to listen to and contact, and how to make the right people listen to you and select you.

They key enabling technologies are social network modeling and social network analysis. Social network modeling means the process of inferring the existing social relationships from available data. Social network analysis means the process of inferring properties and metrics from the network model. Examples of metrics that capture network properties of individual actors are actor connectedness, range, prominence, betweenness, isolation, popularity, and centrality Examples of overall network characteristics include network density, heterogeneity, and centralization. These metrics can be used to calculate recommendations. For a representative set of possible metrics, see the paper by Yaltaghian and Chignell[YaC02].

Social networks can be divided into two classes; knowledge networks and (true) social networks. Knowledge networks represent the extent to which the same or disparate knowledge is distributed among the various members in the system. Cognitive knowledge networks represent individuals' cognitive perceptions of "who knows what" within the system[CZC98]. Efficient ways of modeling knowledge networks are examining what kinds of information resources an actor has authored and published, and examining the citation structures of these resources, for example URLs or references. (True) social networks model the real relationships between actors. The difference to knowledge networks is clear. For example two researchers may be focusing on same research questions and may have very much overlap in their knowledge, but they may never have even heard of each other. On the other hand, two persons may be friends with each other even if they are interested in totally different things and disagree with each other on most

subjects. For the time being the best data for modeling (true) social networks is probably extracted from e-mail communication.

Pujol et al. describe one way for extractinging the web of trust from available resources[PSD02]. For simplicity purposes one can see that model as a set of terms describing the knowledge contained in the personal pages. Once these models are obtained for each member of the community the social network is built by the RelationshipMaker agent by means of a similarity function that takes into account the strength of association between any two members in the community. Members of the community appear as nodes in the social network graph and the directed edges are calculated using following data:

E-mail(a->b) is true when email address of member b exists in the web pages of member a

Resources(a,b) is the set of resources (files than can be reached through the Web) that belong to member b and they have been found in the personal web page of member a.

Depth(R,x) is the depth of the resource R in the personal web page of member x.

Name(a->b) returns the number of occurrences of the name of member b within the personal web pages of member a.

Usually relationship information is not already available, but it must be somehow extracted from available data. However, there are some new developments, where the user can explicitly state relationship information. Examples include the social networking systems such as Friendster[4]. These are not very interesting, because they are closed, proprietary systems for some restricted application area. More interesting is the FOAF (Friend-Of-A-Friend) project [Dod04, Dum02, <FOA>], which is about creating a Web of machine-readable homepages describing people, the links between them and the things they create and do. What makes it especially interesting is that, according to Golbeck and Hendler[GoH04], FOAF space now comprises millions of files. FOAF specifies a machine-readable XML/RDF/OWL description format[BrM04] for describing oneself and relationships to other actors as work organization, projects and other people. There already exists tools for using these descriptions, probably the most interesting in this context being the FOAFBot<Bot>. FOAFBot is an IRC bot that provides access to a knowledge base created by spidering FOAF files. It can sit on an IRC channel and provide basic informational help about the members of a community. The bot can be interrogated with simple questions about the properties of community members. People can be identified either by their IRC nick, full name, or email address:

<edd> foafbot, edd's name

<foafbot> edd's name is 'Edd Dumbill', according to Dan Brickley, Anon35, Niel Bornstein, Jo Walsh, Dave Beckett, Edd Dumbill, Matt Biddulph, Paul Ford

You can see the answer the bot found, along with the owners of the files it found the answer in. Other properties which can be asked for in the same way as name include: email nick image workplace phone homepage. You can also ask for URLs, which shows the URLs of the FOAF files belonging to that person. The bot responds to queries by private message, as well as being addressed in public in a channel. Example of a FOAF description can be found in the appendix 1.

5.2 Reputation and trust

"Reputation systems are the worst way of building trust, except for all those other ways that have been tried from time to time."[Res00].

There's a growing interest in the so called reputation systems, especially among the e-Commerce domain. A reputation system collects, distributes, and aggregates feedback about participants' past behavior. Though few of the producers or consumers of the ratings know each other, these systems help people decide whom to trust, encourage trustworthy behavior, and deter participation by those who are unskilled or dishonest [Res00]. A working example of the concept is the eBay on-line auction site<eBa>. Reputation systems are one kind of recommender systems. In addition to normal recommender system properties actors have to be persistent, so that there is an expectation of future interaction.

Despite the promise of reputation systems, there remain significant challenges requiring further research and commercial development. Resnick et al. name among others[Res00]:

-People may not bother to provide feedback at all. It seems especially difficult to elicit negative feedback.

-It's difficult to assure honest reporting. For example, in order to accumulate positive feedback a group of people might collaborate and rate each other positively, artificially inflating their reputations.

- Difficulty in distributing feedback stems from lack of portability between systems. Amazon.com initially allowed users to import their ratings from eBay. eBay protested vigorously, claiming that their user ratings were proprietary. Ultimately Amazon discontinued its rating-import service. Efforts are underway to construct a more universal framework. For example, virtualfeedback<vir> provides a rating service for users across different systems, but it has yet to gain wide public acceptance.

Now, multi-agent systems researches have been extensively studying the notions of reputation and trust, as they form some of the key ingredients for reaching coordination and cooperation in a world of autonomous, even selfish entities. There's wealth of work considering both theoretical aspects of reputation and practical frameworks for implementing systems. The connections to the information retrieval -driven research in recommender and reputation systems are clear. It seems a very promising idea to combine these efforts. Agents can provide feedback on behalf of actors, mine useful recommendations from large data sets including also implicit feedback information, reveal dishonest reporting by examining relationships between recommenders, offer platforms for open, distributed opinion management schemes, and so on.

Mui, Halberstadt, and Mohtashemi point out the contextualization and personalization aspects of reputation[MHM02]. First, reputation is clearly a context-dependent quantity. For example, one’s reputation as a computer scientist should have no influence on his or her reputation as cook. As existing commercial reputation systems, as eBay and Amazon, provide only one reputation rating per trader or per book reviewer, there's much room for improvement. Because it might be quite hard for users to calculate and understand complex reputation schemes, there's definitely need for agents to handle the complexities of contextualization on behalf of users. Second, reputation can be viewed as global or personalized quantity. Clearly, people do need different resources and appreciate different qualities. Therefore one global estimate seems inadequate. Again, in the case of Amazon or eBay, reputation is a function of the cumulative ratings on users by others, and global reputation is assumed. Agent technology is needed for taking individual preferences into account.

Reputation is a complex concept. Mui, Halberstadt, and Mohtashemi offer a useful typology of reputation[MHM02]. There's two kinds of direct reputation, called encounter-derived reputation and observed reputation, and three kinds of indirect reputation, called prior-derived reputation, group-derived reputation and propagated reputation. Encounter-

derived reputation means the direct observation of actor, while interacting with it. Observed reputation means the reputation estimate that is derived from information about others' encounters with the actor in case. Prior-derived reputation means the reputation that is assigned to actor just because of their identity or type. With new actors, some prior-derived repuation estimate must be used. Typical examples are to use worst possible reputation or neutral reputation. Group-derived reputation means assigning reputation estimate by examining which groups the actor belongs to. Examples include examining actor's social network or assigning firm's reputation to it's employee or vice versa. Propagated reputation means that reputation is estimated by process of asking reputation estimates from other actors. They can ask estimates from yet another actors and the reputations of the estimaters themselves can be taken into account. Of the mentioned reputation types, the propagated reputation seems most promising for incorporating multi-agent technology, by letting the agents to estimate each other (and therefore the actors they are representing) in a distributed and cooperative fashion. The example in chapter 7 presents a framework based on propagated reputation.

The Regret system[SaS02] is one example of using reputation in an agent environment. The existence of some kind of social network is supposed. The Regret system recognizes four kinds of reputation with reliability estimates:

The most reliable reputation is the outcome reputation. Outcome is defined as the outcome of a dialogue between two agents as:

- An initial contract to take a particular course of action and the actual result of the actions taken.

- An initial contract to fix the terms and conditions of a transaction and the actual values of the terms of the transaction.

Outcome reputation is calculated directly from an agent's outcomes database. Deceptions are punished severely and recent outcomes are given more relevance. In the typology given above, outcome reputation is a kind of encounter-derived reputation.

Second is the witness reputation. Beliefs about the reputation of others can be shared and communicated by the members of a society. The reputation that an agent builds on another agent based on the beliefs gathered from society members (witnesses) is called

witness reputation. Reliability of witnesses is estimated by examining their position in the social network. In the typology given above, witness reputation is a kind of observed reputation.

Neighbourhood reputation comes third. Neighbourhood reputation is calculated by examining the target agent's position in the social network. The main idea is that the behaviour of neighbours in the social network and the kind of relation they have with the target agent can give some clues about its possible behaviour. In the typology given above, neighbourhood reputation is a kind of group-derived reputation.

Least reliable is the system reputation. The idea behind system reputations is to use the common knowledge about institutional structures and the role that the agent is playing for that institutional structure as a mechanism to assign default reputations to the agents. An institutional structure is a social structure the members of which have one or several observable features that unambiguously identify them as members of that social structure. The fact that there are observable features to identify its members is what differentiates an institutional structure from other social structures. In the typology given above, system reputation is a kind of prior-derived reputation, because the reputation of institutions in Regret system are initially set default values that cannot change.

Because of the varying reliability, agents give more relevance to the outcome reputation in detriment of the others. If the outcome reputation has a low degree of reliability (for instance because the agent does not have enough information) then the agent will try to use the witness and the neighbourhood reputations. Finally, if its knowledge of the social relationships is short, the agent will try to use the system reputation.

6. A Wider perspective

The idea of using computers to assist in social tasks, as in matchmaking and creating social relationships, opinion forming, evaluating other people and assessing their usefulness, in deciding who to contact and who trust, etc., sounds intriguing. With almost unlimited possibilities comes also the danger that things could somehow start to go seriously wrong[8]. The thought of giving autonomous agents, no matter how intelligent, power of decision over one's social status and decision-making arsenal, may feel somewhat scary for many. One thing is clear. The design of such systems is no way a merely technical engineering task. We should spend some time in analysing phase[9] before moving to implementation and deployment. To take a step toward understanding these systems from a wider perspective, some dimensions that should be considered during system design are presented next.

Timeliness vs. Trustability

There seems to be a fundamental trade-off between timeliness and trustability: the newer a resource, the less there can be experience gathered conserning it. This means that reaching for newer resources always increases the risks, no matter what technologies are available. Because very often the potential of a resource diminishes as time goes by, one has to find a balance between potential benefits and risks involved.

Privacy vs. Functionality

Sacrificing privacy permits increased functionality. The more the system knows about resources, preferences and opinions, the better recommendations it can deliver. On the other hand, people do not want all information conserning them to be public. It may be easiest to let users explicitly and exactly define, what information can be published.

Rationality vs. Creativity

It is rational to consult past experience and stick to solutions, that are found to be adequate. On the other hand, totally new solutions can only be found by partially abandoning familiar patterns. the system-wide consequence of optimizing rationality is to stall progress, and too much weight on creativity will lead to chaos. This is an important trade-off in designing social network topologies. Roughly put, too much connections to outside

world will hinder the creativity of the community, but too little connections will lead to reinventing the wheels and to suboptimal operation.

Speciality vs. Generality

Automatic search and evaluation should guide people not only to select resources that are familiar, similar to other resources, and having the same qualities as earlier chosen resources. It would be very useful to have techniques for broadening one's scope without being instantly driven to serious information overload.

Remembering vs. Forgiving

The better different behaviours can be recorded, the more incentive there is to behave properly, and the easier it is to avoid those behaving badly. However, some incident in the past should not lead to persistent labeling. In addition, people routinely make mistakes. While information itself is not easily erased from a distributed system, it's relevance should diminish somehow. One of the most efficient methods worth considering would be to separate actors' real and digital identity from each other by the use of pseudonyms[10].

7. An example: Singh et al.'s framework

Singh et al.'s framework for agent-based referral systems is now presented as an example. Example text has been gathered from multiple sources[YoS00, YoS03a, YoS03b, YuS02, YuS03, YVS99, YVS03]. The aim of the framework is to help people find the right people or the right services via agent-based collaborative social matchmaking. Features of the framework include full distribution achieved by multi-agent technologies and reputation and trust management based on referrals. Also, most previous approaches, such as ReferralWeb[KSS97], model the social network statically, whereas Singh et al.'s solution can model the relationship of agents dynamically by allowing them to adapt through neighbor changes.

An agent-based referral network is a multiagent system whose member agents give referrals to one another (and are able to follow referrals received from other agents). the referrals-based approach considers interactions among the agents participating in a community. The agents help one another and evaluate each other’s effectiveness. Good interactions reinforce their social relationships and bring them closer, whereas bad interactions weaken their social relationships. The agents decide with whom to interact. Intuitively, agents base their decisions on specific feedback or generic policies set by their users, but in terms of its interactions with other agents, each agent is autonomous. That is, an agent may or may not respond to another agent by providing a service or a referral. When an agent does respond, there are no guarantees about the quality of the service or the suitability of a referral. Likewise, no agent should necessarily be trusted by others: an agent unilaterally decides how to rate another principal.

From user's point of view, the agent returns to the user a ranked list of these experts based on similarity. The user can select a few of them to send the query. Each of them may respond with an answer to the query, or referrals to other users, or give no response at all. What the agent needs to learn about others is their potential usefulness for a given query, i.e., the likelihood of getting a good answer or a referral that (in as few steps as possible) yields a good answer. To do so effectively presupposes certain representation and reasoning capabilities on the part of each agent.

The acquaintances are the agents that the given agent would contact and the agents that it would point (refer) others to. Each agent maintains models of its acquaintances, which describe their expertise and sociability. Expertise can be understood as quality of the services an agent provides and the agent's abilities to act in a trustworthy manner, and sociality as agent's abilities to refer to other trustworthy agents and to the quality of the referrals they provide. Each agent evaluates others based on a linear combination of their expertise and sociability. That is, the relevance of a neighbor to a given query depends not only on the similarity of the query to the user's expertise, but also on the weight assigned to sociability versus expertise. Both of these elements are learned based on service ratings from the user.

The agents in a referral network act in accordance with the following abstract protocol. An agent begins to look for a trustworthy provider for a specified service. The agent queries some other agents from among its neighbors. A queried agent may offer to provide the specified service or may give referrals to other agents. The querying agent may accept a service offer, if any, and may pursue referrals, if any. Using acquaintance models, an agent applies its neighbor selection policy to decide which of its acquaintances to keep as neighbors. Key factors include the quality of the service received from a given provider, and the resulting value that can be placed on a series of referrals that led to that provider. An agent's total belief about another agent combines the local belief (if any) with testimonies received from others reached (if any). Total belief is used for deciding whether the agent being considered is trustworthy. To prevent non-well-founded cycles, agents are restricted from propagating their total beliefs. The neighborhood relation among the agents induces the structure of the given society. In general, as described above, the structure is adapted through the decisions of the different agents.

The referrals approach has some natural advantages[11]:

- The model addresses the important challenge of finding trustworthy agents, which is nontrivial in open systems. First, referrals can apply even in the absence of centralized authorities and even when regulations may not ensure that services are of a suitable quality. Second, because service needs are often context-sensitive, a response from an agent can potentially benefit from the knowledge that it has of the other's needs.

- Because agents maintain models of others, they are able to annotate their links to other agents in terms of those models. For example, an agent A may believe that agent B is the

best source for information on computer science and C is the best source for information on cooking.

- Referrals are generated dynamically. Thus, instead of merely looking at a static Web page, it is possible to model computations wherein (as it were) the page is produced on demand. The responder (acting as the producer of a Web page) can consider its relationship with the requestor in deciding how to respond. Importantly, the referrals approach, because it involves requests among the participants, can apply on the so-called Deep Web, whereas a conventional mining approach would apply only to the static Web.

- The problem of searching a large network has been studied for peer-to-peer (P2P) networks. Typically, a P2P node broadcasts a search request to its peers, who propagate the request to their peers, and so on. By contrast, in a referral system, referrals are sent back to the requesting agent, who can adaptively direct or end the search. in P2P techniques, the routing table for each node is usually fixed and thus the network is not reconfigurable. Most importantly, the techniques are not applicable to social networks, which cannot be partitioned by IP address.

- Traditional collaborative filtering involves a server aggregating the choices of several users and making recommendations to a user based on the choices of users similar to the given user. This approach has the limitation of identifying the user providing a rating to the server, while not revealing the source of recommendations. Referral approach, by contrast, is decentralized and lets the users control to whom they reveal their ratings.

- Most approaches involving any variant of collaborative filtering do not distinguish between users' interests and expertise. They tend to assume that the users will form clusters easily. However, when the users have different interest and expertise, then the people who need to know them are in general different from the people they need to know. This makes the clustering less clean. In fact, Singh et al. observed that the quality of the social network for helping find information improves when the clustering decreases. This is at odds with techniques such as collaborative filtering, wherein recommendations to a user are produced based on the decisions made by others who are similar to the user, i.e., in the user's cluster. Singh et al.'s results don't directly contradict collaborative filtering, but they do suggest that different methods might be more applicable when high-quality referrals in decentralized systems are desired.

The Multiagent Referral System, MARS, is a prototype system based on the above ideas. MARS is intended to be an accurate, dynamic, and evolving multiagent system that can achieve the effect of informal social networks that exist in an organization or community. MARS introduces a two-pronged solution. One, it applies information retrieval techniques as a basis for learning and action. Two, it allows agents to exchange profile information during interaction; some learning takes place with virtually every interaction among the agents. MARS agents give and take referrals as described above. They also include an interface in which text queries can be entered by users. Each user is associated with a personal agent. The queries by the user are first seen by his agent who decides the potential contacts to whom to send the query. After consultation with the user, the agent sends the query to the agents for other likely people. The agent who receives a query can decide if the query suits its user and let the user see that query. In addition to or instead of just forwarding the query to the user, the receiving agent may respond with referrals to other users. If the receiving agent or user wish they can discard the query and never respond to it in any way.

A challenge for referral systems is how to bootstrap them. MARS uses a server where new users register themselves along with their topics of expertise and with which they can find existing users based on their self-stated interests. An agent may contact the registration server as a fall back mechanism if it cannot find a suitable contact on its own. MARS is implemented using IBM's Agent Building and Learning Environment (ABLE) for its reasoner [Big02, <Age>] on a distributed system of Communicators (roughly, high-end PDAs with cellular phones) for use in a practical social network.

Singh et al. note, that their present approach does not fully protect against spurious ratings generated by malicious agents. It relies upon there being a large number of agents who offer honest ratings to override the effect of the ratings provided by the malicious agents. In future work, they plan to study the special problems of lying and rumors in extensions of the present evidential framework. They also plan to study evolutionary situations where groups of agents consider rating schemes for other agents. The purpose is not only to study alternative approaches for achieving more efficient communities, but also to test if the framework is robust against invasion and, hence, is more stable.

8. Conclusions

When there are many resources, one has to make decision which one(s) to choose. Currently much of the Internet's potential is lost because the evaluation and selection of found resources cannot be done as efficiently as the searching. The openness of environments does not make the situation any easier. Agent-based recommender systems are one remedy for the ever increasing information overload. The main tasks of recommender systems are to help users to:

- evaluate resources (e.g. by gathering and aggregating opinions conserning the resource)
- find evaluations of a resource (e.g. by recommending experts of the field who can help)
- evaluate the evaluaters (e.g. their expertise, reputation and trustfulness)

There's various uses for software agent technology in achieving the mentioned tasks. User modeling is needed for taking personal preferences into account in recommending. Distribution, matchmaking, and reputation and trust management can be achieved with multi-agent system technologies. Intelligent reasoning capabilities are needed for social network modeling and analysis. Possibilities and challenges of these tasks were discussed in the paper. It was shown, that there is much relevant research being done for achieving these goals.

As general design issues worth considering, trade-offs in timeliness vs. trustability, privacy vs. functionality, rationality vs. creativity, speciality vs. generality, and remembering vs. forgiving were identified.

Singh et al.'s framework for agent-based referral systems was presented as an example of state-of-the-art research concerning some of the essential issues mentionded in the paper.

To sum up, recommender systems seem like one of the most promising application area for software agent technologies. There seems to be a natural fit between these areas.

9. Notes

1 The presentation of resources can be done using interface agents, but possibilities of them are not considered in this paper.

2 One must be aware that descriptions are sometimes called recommendations, but this is problematic because explicit presentation of opinions is then presupposed.

3 The lack of actors as special resource class is not a feature of Google, but really a deficiency of the whole www architecture. Dedicated Universal Resource Identifier schemes hopefully correct the situation in the future.

4 In addition to Friendster<Fri>, other applications of social networking worth mentioning include Ecademy<Eca>, LinkedIn<Lin>, orkut<ork>, Ryze<Ryz>, and Spoke<Spo>.

5 Old, but still relevant review of the basic social filtering algorithms is the work by Shardanand and Maes[ShM95].

6 For a concise introduction to distributed artificial intelligence, check Vlassis' article[Vla03].

7 For a more detailed discussion about these phenomena, consult Wellman's article[Wel02].

8 The most probable problem scenario is that those, who can't or won't use information technology, will drop out.

9 Unfortunately the progress tends often to be so fast, that there's not much time for sitting back.

10 Friedman and Resnick offer some valuable insights about the use of pseudonyms[FrR00].

11 It must be noted, that Singh et al. see the referral-base approach so important, that treat referrals as the key organizing principle for large-scale multiagent systems.

10. References

[Big02] Bigus, J., et al., ABLE: a toolkit for building multiagent autonomic systems - Agent Building and Learning Environment. IBM Systems Journal, Sept., 2002.

<http://articles.findarticles.com/p/articles/mi_m0ISJ/is_3_41/ai_91469710>

[BrM04] Brickley, D., Miller, L., FOAF Vocabulary Specification. Namespace Document, 1 May 2004.

<http://xmlns.com/foaf/0.1/>

[Boy04] Boyd, D., Friendster and Publicly Articulated Social Networking. Proceedings of the Conference on Human Factors in Computing Systems 2004 (CHI'04). ACM, 2004. 1279-1282.

[BuH99] Budzik, J., Hammond, K., User Interactions with Everyday Applications as Context for Just-in-time Information Access. Proceedings of the 2000 International Conference on Intelligent User Interfaces, ACM Press, 2000, 44-51.

[ByC01] Byun, H., Cheverst, K., Exploiting User Models and Context-Awareness to Support Personal Daily Activities. (Online) Proceedings of Workshop on User Modeling for Context-Aware Applications at the 8th International Conference on User Modeling (UM2001), Sonthofen, Germany, July 13, 2001.

[Cla01] Claypool, M., et al., Implicit Interest Indicators. Proceedings of the Intelligent User Interfaces 2001 (IUI'01), New Mexico, USA. ACM, 33-40.

[Cla03] Clausen, A., Online Reputation Systems: The Cost of Attack of PageRank. Honours thesis, The University of Melbourne, 2003.

<http://members.optusnet.com.au/clausen/reputation/rep-cost-attack.pdf>

[Col88] Coleman, J., Social Capital in the Creation of Human Capital. American Journal of Sociology, 94 supplement, (1988), 95-120.

[Com97] Communications of the ACM, 40, 3, (1997).

[CZC98] Contractor, N., Zink, D., Chan, M., IKNOW: A tool to assist and study the creation, maintenance, and dissolution of knowledge networks. In Toru Ishida (Ed.), Community Computing and Support Systems, Lecture Notes in Computer Science 1519. Springer, 1998. 201-217.

[Dod04] Dodds, L., Introduction to FOAF. XML.com, February 04, 2004.
<http://www.xml.com/pub/a/2004/02/04/foaf.html>

[Dum02] Dumbill, E., XML Watch: Finding friends with XML and RDF. IBM Developer Works, June 2002.
<http://www-106.ibm.com/developerworks/xml/library/x-foaf.html>

[Fon97] Foner, L., Yenta: A Multi-Agent, Referral-Based Matchmaking System. Proceedings of ACM Autonomous Agents'97, California, USA, 1997. 301-307.

[FrR00] Friedman, E., Resnick, P., The Social Cost of Cheap Pseudonyms. Journal of Economics and Management Strategy, 10, 2, (2001), 173-199.

[GoH04] Golbeck, J., Hendler, J., Accuracy of Metrics for Inferring Trust and Reputation in Semantic Web-based Social Networks. Submitted to EKAW'04, 2004.
<www.mindswap.org/papers/GolbeckEKAW04.pdf>

[GPD03] Gómez-Sanz, J., Pavón, J., Díaz-Carrasco, Á, The PSI3 Agent Recommender System. Cueva Lovelle, J., et al. (Eds.): ICWE 2003, Lecture Notes in Computer Science 2722. Springer, 2003, 30-39.

[Gim02] Gimenez-Lugo, G. ,et al., Enriching Information Agents' Knowledge by Ontology Comparison: a Case Study. In Garijo, F., Riquelme Santos, J., Toro, M., (Eds.): Advances in Artificial Intelligence – Proceedings of IBERAMIA 2002, 8th Ibero-American Conference on AI, Seville, Spain, November 12-15, 2002, Lecture Notes in Computer Science 2527. Springer, 2002, 546-555.

[Hat99] Hattori, F., et al., Socialware: Multiagent Systems for Supporting Network Communities. Communications of the ACM, 42, 3, (1999), 55-61.

[KeT03] Kelly, D., Teevan, J., Implicit Feedback for Inferring User Preference: A Bibliography. SIGIR Forum, 37, 2, (2003), 18-28.

[KlS01] Klusch, M., Sycara, K., Brokering and Matchmaking for Coordination of Agent Societies: A Survey. In Omicini, A., et al., (Eds.), Coordination of Internet Agents: Models, Technologies, and Applications. Springer 2001. 197-224.

[Klu99] Klusch, M., (Ed.)., Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet. Springer, 1999.

[Kob01] Kobsa, A., Generic User Modeling Systems. User Modeling and User-Adapted Interaction, 11, (2001), 49-63.

[Kon04] Konstan, J., Introduction to Recommender Systems: Algorithms and Evaluation. ACM Transactions on Information Systems, 22, 1, (2004), 1-4.

[KSS97] Kautz, H., Selman, B., Shah, M., ReferralWeb: Combining Social Networks and Collaborative Filtering. Communications of the ACM, 40, 3, (1997).

[LaN92] Lamming, M., Newman, W., Activity-Based Information Retrieval: Technology in Support of Personal Memory. Vogt, F., (Ed.), Proceedings of Information Processing 92 (Personal Computers and Intelligent Systems), Vol. 3, Elsevier Science, Amsterdam, 1992, 68-81.

[LFW01] Lieberman, H., Fry, C., Weitzman, L., Exploring the Web with Reconnaissance Agents. Communications of the ACM, 44, 8, (2001), 69-75.

[MHM02] Mui, L., Halberstadt, A., Mohtashemi, M., Notions of Reputation in Multi-Agent Systems: A Review. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, (AAMAS'02), Bologna. ACM Press, 2002, 280-287.

[MLd03] Montaner, M., López, B., Lluís de la Rosa, J., Taxonomy of Recommender Agents on the Internet. Artificial Intelligence Review, 19, (2003), 285–330.

[Nar02] Nardi, B., et al., Integrating communication and information through ContactMap. Communications of the ACM, 45, 4, (2002), 89-95.

[Nic97] Nichols, D., Implicit Rating and Filtering. Proceedings of the 5th DELOS Workshop on Filtering and Collaborative Filtering, Hungary, 1997, 31-36.

[OaK98] Oard, D., Kim., J., Implicit Feedback for Recommender Systems. In Kautz, H., (ed.), Papers from the 1998 AAAI Workshop on Recommender Systems, Technical Report WS-98-08, The AAAI Press, 1998, 80-82.

[OgY99] Ogata, H., Yano, Y., Combining Social Networks and Collaborative Learning in Distributed Organizations. World Conference on Educational Multimedia, Hypermedia and Telecommunications 1999(1), 119-125. [Online].

<http://dl.aace.org/4228>

[Orw96] Orwant, J., For want of a bit the user was lost: Cheap user modeling. IBM Systems Journal 35, 3&4, (1996), 398-416.

[Pag98] Page, L., et al.,The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

<http://dbpubs.stanford.edu/pub/1999-66>

[PSD02] Pujol, J., Sanguesa, R., Delgado, J., Extracting Reputation in Multi Agent Systems by Means of Social Network Topology. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, (AAMAS'02), Bologna. ACM Press, 2002, 467-474.

[Res00] Resnick, P., et al., Reputation Systems: Facilitating Trust in Internet Interactions. Communications of the ACM, 43, 12, (2000), 45–48.

[Res04] Resnick, P., Impersonal Sociotechnical Capital, ICTs, and Collective Action Among Strangers. To appear in Dutton, W., et al. (Eds.), Transforming Enterprise. MIT Press, 2004.

<http://www.si.umich.edu/~presnick/papers/xforment/chapter.pdf>

[RhM00] Rhodes, B., Maes, P., Just-in-time Information Retrieval Agents. IBM Systems Journal, 39, 3&4, (2000), 685-704.

[ReV97] Resnick, P., Varian, H., Recommender Systems. Communications of the ACM, 40, 3, (1997), 56-58.

[SaS02] Sabater, J., Sierra, C., Reputation and Social Network Analysis in Multi-Agent Systems. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, (AAMAS'02), Bologna. ACM Press, 2002. 475-482.

[SAW94] Schilit, B., Adams, N., Want, R., Context-aware Computing Applications. Proceedings of the Workshop on Mobile Computing Systems and Applications, IEEE, 1994, 85-90.

[ShM95] Shardanand, U., Maes, P., Social information filtering: Algorithms for automating “Word of Mouth”. Proceedings of the Conference on Human Factors in Computing Systems (CHI’95). ACM, New York, 1995. 210–217.

[SKX98] Schlichter, J., Koch, M., Xu, C., Awareness - The Common Link Between Groupware and Communityware. In Ishida, T., (Ed.), Community Computing and Support Systems. Springer, 1998. 77-93.

[Syc96] Sycara, K., et al., Distributed Intelligent Agents. IEEE Expert, 11, 6, (1996), 36-46.

[Syc99a] Sycara, K., et al., Matchmaking among Heterogeneous Agents on the Internet. Proceedings of the 1999 AAAI Spring Symposium on Intelligent Agents in Cyberspace.

<http://www.ri.cmu.edu/pubs/pub_2621.html>

[Syc99b] Sycara, K., et al., Dynamic Service Matchmaking Among Agents in Open Information Environments. ACM SIGMOD Record 28, 1, (1999), 47-53.

[SYR03] Sugawara, K., Yu, Y., Ragsdale, B., Design of An Agent-based Middleware for Job Matchmaking in Teleworking Community. Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne, Australia. ACM Press, 2003. 1128-1129.

[TeH01] Terveen, L., Hill, W., Beyond Recommender Systems: Helping People Help Each Other. In Carroll, J., (Ed.), HCI in the New Millennium. Addison Wesley, 2001. 475-486.

[Viv99] Vivacqua, A., Agents for Expertise Location. Proceedings of the AAAI Spring Symposium on Intelligent Agents in Cyberspace. Technical Report SS-99-03, Stanford, CA, USA, March 1999. 9-13.

<http://citeseer.ist.psu.edu/205446.html>

[Vla03] Vlassis, N., A Concise Introduction to Multiagent Systems and Distributed AI. University of Amsterdam, The Netherlands, 2003.

<http://carol.science.uva.nl/~vlassis/cimasdai/>

[Wan02] Wang, F., Self-organising Communities Formed by Middle Agents. Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'02), Bologna, Italy. ACM Press, 2002. 1331-1339.

[Wel02] Wellman, B., Designing the Internet for a Networked Society. Communications of the ACM, 45, 5, (2002), 91-96.

[Yos00] Yoshida, S., et al., Building a Network Community Support System on the Multi-Agent Platform Shine. in Zhang, C., Soo, V.-W., (Eds.), PRIMA 2000, Lecture Notes in Artificial Intelligence 1881. Springer, 2000. 88-100.

[YoS03a] Yolum, P., Singh, M., Emergent Properties of Referral Systems. Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne, Australia. ACM Press, 2003. 592-599.

[YoS03b] Yolum, P., Singh, M., Dynamic Communities in Referral Networks. Web Intelligence and Agent Systems, 1, 2, (2003), 105-116.

[YuS02] Yu, B., Singh, M., Distributed Reputation Management for Electronic Commerce. Computational Intelligence, 18, 4, (2002), 535-549.

[YuS03] Yu, B., Singh, M., Searching Social Networks. Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03), Melbourne, Australia. ACM Press, 2003, 65-72.

[YVS99] Yu, B., Venkatraman, M., Singh, M., A Multiagent Referral System for Expertise Location. Working Notes of the AAAI Workshop on Intelligent Information Systems, 66-69.
<http://www.cs.cmu.edu/~byu/papers/aaai-99-iis.ps>

[YVS03] Yu, B., Venkatraman, M., Singh, M., An Adaptive Social Network for Information Access: Theoretical and Experimental Results. Applied Artificial Intelligence, 17, 1, (2003), 21-38.

Web references

(links checked 17.5.2004)

<Age> Agent Building and Learning Environment
<http://www.alphaworks.ibm.com/tech/able>

<Ama> Amazon
<http://www.amazon.com/>

<ask> askme
<http://www.askme.com/>

<Biz> Bizrate
<http://www.bizrate.com/>

<Bot> FOAFBot
<http://usefulinc.com/foaf/foafbot>

<eBa> eBay

<http://www.ebay.com/>

<Eca> Ecademy
<http://www.ecademy.com/>
<epi> epinions
<http://www.epinions.com/>

<exp> expertcentral
<http://www.expertcentral.com/>

<FOA> The Friend of a Friend (FOAF) project
<http://www.foaf-project.org/>

<Fri> Friendster
<http://www.friendster.com/>

<Goo> Google
<http://www.google.com/>

<Lin> LinkedIn
<https://www.linkedin.com/>

<ork> orkut
<http://www.orkut.com/>

<Ryz> Ryze
<http://new.ryze.com/>

<Spo> Spoke
<http://www.spokesoftware.com/>

<vir> virtualfeedback
< http://www.virtualfeedback.com/>

<Yah> Yahoo! Auction
<http://auctions.yahoo.com/>

Appendix 1

This is an example FOAF description, downloaded 15.5.2004 from <http://www.ideaspace.net/users/wkearney/foaf.xrdf>.

```
<?xml version="1.0"?>
<!-- xml-stylesheet href="http://www.ideaspace.net/users/wkearney/foaf.xsl" type="text/xsl" -->
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:dcterms="http://purl.org/dc/terms"
    xmlns:wot="http://xmlns.com/wot/0.1/"
    xmlns:rss="http://purl.org/rss/1.0/"
    xmlns:air="http://www.daml.org/2001/10/html/airport-ont#"
    xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#"
    xmlns:wn="http://xmlns.com/wordnet/1.6/"
    xmlns:mbti="http://www.ideaspace.net/users/wkearney/schema/mbti/0.1#"
    xmlns:zodiac="http://www.ideaspace.net/users/wkearney/schema/astrology/0.1#"
    xmlns:ks="http://www.ideaspace.net/users/kearney/schema/kitchensink/0.1#"
    xmlns:bio="http://purl.org/vocab/bio/0.1/"
  >

<!-- So what the heck is this?  It's a Friend of a Friend document -->
<!-- you can learn more about Foaf at http://xmlns.com/foaf/0.1/ -->
<!-- Please DO NOT list e-mail addresses for others (just yourself).  Use an SHA1 hash instead -->
<!-- If you need to make a hash, you can use my generator at: http://feeds.archive.org/misc/hash -->
<!-- and be SURE the hash is correct! -->

<foaf:Person rdf:ID="wkearney99">
<foaf:name>Bill Kearney</foaf:name>
<foaf:title>Mr.</foaf:title>
<foaf:firstName>William</foaf:firstName>
<foaf:surname>Kearney</foaf:surname>
<foaf:mbox rdf:resource="mailto:wkearney99@hotmail.com" />
<foaf:nick>Bill</foaf:nick>
<foaf:nick>wkearney</foaf:nick>
<foaf:nick>wkearney99</foaf:nick>
<foaf:mbox_sha1sum>98783d46199c7733d0e452c93ba0cff0baa4b61b</foaf:mbox_sha1sum> <!-- my main
address -->
<foaf:mbox_sha1sum>61db4304e2d21b2b58092dbfe6fc4924c2743850</foaf:mbox_sha1sum>
```

```
<foaf:mbox_sha1sum>ff604cc25629a0b86f7cdc9e1f9101f506494f3b</foaf:mbox_sha1sum>
<foaf:weblog rdf:resource="http://www.ideaspace.net/users/wkearney"/>
<foaf:weblog rdf:resource="http://www.syndic8.com/~wkearney/blogs/syndic8"/>
<foaf:homepage rdf:resource="http://www.ideaspace.net/users/wkearney"/>
<foaf:img rdf:resource="http://www.ideaspace.net/users/wkearney/images/SM.JPG" />
<foaf:phone rdf:resource="tel:301-365-5685" />
<foaf:workplaceHomepage rdf:resource="http://www.syndic8.com/"/>
<foaf:workInfoHomepage rdf:resource="http://www.syndic8.com/~wkearney/blogs/syndic8/"/>
<foaf:currentProject rdf:resource="http://www.syndic8.com/" />
<foaf:currentProject rdf:resource="http://feeds.archve.org/" />
<foaf:interest rdf:resource="urn:isbn:026258120"/>
<dcterms:issued>1964-05-20</dcterms:issued>
<foaf:weblog rdf:resource="http://www.ideaspace.net/users/wkearney"/>
<foaf:weblog rdf:resource="http://www.syndic8.com/~wkearney/blogs/syndic8" />
<foaf:homepage rdf:resource="http://www.ideaspace.net/users/wkearney"/>
<rdfs:seeAlso rdf:resource="http://www.syndic8.com/~wkearney/blogs/syndic8/xml/index.rdf" />
<foaf:depiction rdf:resource="http://www.ideaspace.net/users/wkearney/images/109-0913_IMG-SM.JPG" />
<foaf:depiction rdf:resource="http://www.ideaspace.net/users/wkearney/images/with_cerf_sm.JPG" />

<mbti:MBTI>ENTP</mbti:MBTI>
<zodiac:Sign>Gemini</zodiac:Sign>

<!-- so now I can say, yes, this foaf has everything and the kitchen sink it it -->
<ks:kitchenSink rdf:resource="http://www.blancoamerica.com/final_files/511-745.html" />

<ks:ewn rdf:value="1e-10"/>
<!-- whats your ewn?
http://diveintomark.org/archives/2003/04/21/whats_your_winer_number.html -->

<contact:nearestAirport>
<wn:Airport air:icao="KDCA" air:iata="DCA" />
</contact:nearestAirport>

<foaf:depiction rdf:resource="http://www.ideaspace.net/users/wkearney/images/109-0913_IMG-SM.JPG" />
<foaf:depiction rdf:resource="http://www.ideaspace.net/users/wkearney/images/with_cerf_sm.JPG" />

<foaf:knows rdf:resource="#aswartz" />
<foaf:knows rdf:resource="#ben" />
<foaf:knows rdf:resource="#bitworking" />
<foaf:knows rdf:resource="#clkeller" />
<foaf:knows rdf:resource="#danbri" />
<foaf:knows rdf:resource="#dean" />
<foaf:knows rdf:resource="#djadams" />
<foaf:knows rdf:resource="#dwiner" />
<foaf:knows rdf:resource="#eric" />
<foaf:knows rdf:resource="#f8dy" />
<foaf:knows rdf:resource="#iand" />
```

```
<foaf:knows rdf:resource="#jbond" />
<foaf:knows rdf:resource="#jeffbarr" />
<foaf:knows rdf:resource="#lgonze" />
<foaf:knows rdf:resource="#morbus" />
<foaf:knows rdf:resource="#morten" />
<foaf:knows rdf:resource="#philor" />
<foaf:knows rdf:resource="#rael" />
<foaf:knows rdf:resource="#rubys" />
<foaf:knows rdf:resource="#sbpalmer" />
<foaf:knows rdf:resource="#simonstl" />
<foaf:knows rdf:resource="#tappnel" />
<foaf:knows rdf:resource="#uche" />
<foaf:knows rdf:resource="#vcerf" />

<rdfs:seeAlso
rdf:resource="http://www.ecademy.com/module.php?mod=network&amp;op=foafrdf&amp;uid=1" />
<rdfs:seeAlso rdf:resource="http://www.ideaspace.net/users/wkearney/foaf.xrdf" />
<rdfs:isDefinedBy rdf:resource="http://www.ideaspace.net/users/wkearney/foaf.xrdf" />

<bio:event>
<bio:Birth>
<bio:date>1964-05-26</bio:date>
<bio:place>Washington, DC, United States of America</bio:place>
</bio:Birth>
</bio:event>

</foaf:Person>
<foaf:Project rdf:ID="ideaspace">
<foaf:name>ideaspace</foaf:name>
<rdfs:seeAlso rdf:resource="http://www.ideaspace.net/foaf.xrdf" />
</foaf:Project>
<foaf:Project rdf:ID="feedarchive">
<foaf:name>The Internet Archive of Newsfeeds</foaf:name>
<rdfs:seeAlso rdf:resource="http://feeds.archive.org/foaf.xrdf" />
</foaf:Project>

<!-- People -->
<!-- These are linked to me via the use of the rdf:ID inside foaf:knows elements up in my foaf:Person -->
<!-- This is yet another really cool thing about RDF -->

<foaf:Person rdf:ID="jbond">
<foaf:name>Julian Bond</foaf:name>
<foaf:mbox_sha1sum>fc521285feac8d1de0a488166aeeb9dbf99070e1</foaf:mbox_sha1sum>
<rdfs:seeAlso
rdf:resource="http://www.ecademy.com/module.php?mod=network&amp;op=foafrdf&amp;uid=1" />
</foaf:Person>
<foaf:Person rdf:ID="ben">
```

```
<foaf:name>Ben Hammersley</foaf:name>
<rdfs:seeAlso rdf:resource="http://www.benhammersley.com/benfoaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="jeffbarr">
<foaf:name>Jeff Barr</foaf:name>
<foaf:mbox_sha1sum>06d61a89aab9736b97ae7c0b0a092e93f877f608</foaf:mbox_sha1sum>
<foaf:depiction rdf:resource="http://www.syndic8.com/jeffhead.jpg" />
</foaf:Person>
<foaf:Person rdf:ID="danbri">
<foaf:name>Dan Brickley</foaf:name>
<rdfs:seeAlso rdf:resource="http://rdfweb.org/people/danbri/rdfweb/webwho.xrdf" />
</foaf:Person>
<foaf:Person rdf:ID="iand">
<foaf:name>Ian Davis</foaf:name>
<foaf:mbox_sha1sum>69e31bbcf58d432950127593e292a55975bc55fd</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://internetalchemy.org/iand/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="morbus">
<foaf:name>Morbus Iff</foaf:name>
<foaf:mbox_sha1sum>c772b0edbe3ae910b758efd35b1d2e10ce22342e</foaf:mbox_sha1sum>
<foaf:depiction rdf:resource="http://www.oreillynet.com/images/people/morbus_iff.jpg" />
<foaf:homepage rdf:resource="http://www.disobey.com/dnn/" />
</foaf:Person>

<rdf:Description rdf:about="http://www.oreillynet.com/images/people/morbus_iff.jpg">
<dc:title>Kevin Hemenway</dc:title>
<dc:description>Picture of the infamous Morbus Iff</dc:description>
</rdf:Description>

<foaf:Person rdf:ID="f8dy">
<foaf:name>Mark Pilgrim</foaf:name>
<foaf:mbox_sha1sum>85d089d9dc87139d5542aa4ee2822bf65e56b55e</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://diveintomark.org/public/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="simonfell">
<foaf:name>Simon Fell</foaf:name>
<foaf:mbox_sha1sum>06829dcf0c70a0234242083bf641d8868c01b988</foaf:mbox_sha1sum>
<foaf:mbox_sha1sum>7af148f3cdf2c18c23680327c77d79ebd6d2148d</foaf:mbox_sha1sum>
<foaf:mbox_sha1sum>a46ff5dc4a2975a5f4d9d3b6d38e2f28be4baf13</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://www.pocketsoap.com/weblog/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="simonstl">
<foaf:name>Simon St. Laurent</foaf:name>
<foaf:mbox_sha1sum>966c4daa7a0fe2eefe698b806fb87971c82fe08d</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://simonstl.com/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="bitworking">
```

<foaf:name>Joe Gregorio</foaf:name>
<foaf:mbox_sha1sum>ad7039cbaf2bf89fd96b902236f5f0cb05f18d2f</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://bitworking.org/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="rubys">
<foaf:name>Sam Ruby</foaf:name>
<foaf:mbox_sha1sum>703471c6f39094d88665d24ce72c42fdc5f20585</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://www.intertwingly.net/public/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="tappnel">
<foaf:name>Timothy Appnel</foaf:name>
<foaf:mbox_sha1sum>79416fdee3a23db80c29835a05ac9eb5841c9f3b</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://www.mplode.com/tima/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="djadams">
<foaf:name>DJ Adams</foaf:name>
<foaf:mbox_sha1sum>7e8bb4281e49eef206e2591078b2e88527d9f1a4</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://www.pipetree.com/~dj/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="aswartz">
<foaf:name>Aaron Swartz</foaf:name>
<foaf:mbox_sha1sum>8d85da981271c356b073fb0fcc9dc63bfc471fe4</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://www.aaronsw.com/about.xrdf" />
</foaf:Person>
<foaf:Person rdf:ID="morten">
<foaf:name>Morten Frederiksen</foaf:name>
<foaf:mbox_sha1sum>65b983bb397fb71849da910996741752ace8369b</foaf:mbox_sha1sum>
<foaf:mbox_sha1sum>b425d60a76a7b7fa5e62823df718adff50fea351</foaf:mbox_sha1sum><!-- mof-spork -->
<foaf:mbox_sha1sum>50b597e43b6d05ad292f3eeb059b6b2716446263</foaf:mbox_sha1sum><!-- mof-syn-sub -->
<foaf:mbox_sha1sum>461179310021b2185ad7f67f14e5d4deb2107c47</foaf:mbox_sha1sum><!-- mof-sucr -->
<rdfs:seeAlso rdf:resource="http://xml.mfd-consult.dk/foaf/morten.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="sbpalmer">
<foaf:name>Sean Palmer</foaf:name>
<foaf:mbox_sha1sum>c66b1cf8ee172f06b7b41f9e6257623ba14eab5f</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://infomesh.net/sbp/info.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="rael">
<foaf:name>Rael Dornfest</foaf:name>
<foaf:mbox_sha1sum>f7b3d6c9a426fb55c0cbd29d7d5c813570934612</foaf:mbox_sha1sum>
</foaf:Person>
<foaf:Person rdf:ID="philor">
<foaf:name>Phil Ringnalda</foaf:name>
<foaf:mbox_sha1sum>619982223ce2d9158afc47173fe29da0f1a4771c</foaf:mbox_sha1sum>

```
<rdfs:seeAlso rdf:resource="http://philringnalda.com/rdf/foaf-rdf.xml" />
</foaf:Person>
<foaf:Person rdf:ID="eric">
<foaf:name>Eric Vitiello Jr.</foaf:name>
<foaf:mbox_sha1sum>5f93ed6a7346e6b8a767ebd87c407d4765b4e228</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="http://www.perceive.net/" />
<rdfs:seeAlso rdf:resource="http://www.perceive.net/xml/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="dean">
<foaf:name>Dean McKenzie</foaf:name>
<foaf:mbox_sha1sum>75889a7dc0485897a44c8cab0f45a94755c17447</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://deanmckenzie.org/foaf.xrdf" />
</foaf:Person>
<foaf:Person rdf:ID="clkeller">
<foaf:name>Chris Keller</foaf:name>
<foaf:mbox_sha1sum>4671feae8ef3bafe8c4bdac213b17c9edf49b15e</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://www.ideaspace.net/users/clkeller/foaf.rdf" />
</foaf:Person>
<foaf:Person rdf:ID="dwiner">
<foaf:name>Dave Winer</foaf:name>
<foaf:mbox_sha1sum>edb2e2ba4d800af4bccccce49a92964fe9953aeb</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="http://www.scripting.com/" />
<foaf:depiction rdf:resource="http://www.yassfarm.com/images/winerlog/boyssaunasm.gif" />
</foaf:Person>

<foaf:Person rdf:ID="uche">
<foaf:name>Uche Ogbuji</foaf:name>
<foaf:mbox_sha1sum>22895cddb2a430bae696dc80eb6a21a10d5cfaf3</foaf:mbox_sha1sum>
<foaf:img rdf:resource="http://www-106.ibm.com/developerworks/i/p-uche.jpg" />
</foaf:Person>

<foaf:Person rdf:ID="vcerf">
<foaf:name>Vinton Cerf</foaf:name>
<foaf:mbox_sha1sum>910a4f5ef7f0e062ae1fcab1ca981d14ef40563e</foaf:mbox_sha1sum>
<foaf:depiction rdf:resource="http://www.ideaspace.net/users/wkearney/images/with_cerf_sm.JPG" />
<foaf:currentProject rdf:resource="http://www.ipnsig.org/home.htm"/>
</foaf:Person>

<foaf:Person rdf:ID="lgonze">
<foaf:name>Lucas Gonze</foaf:name>
<foaf:mbox_sha1sum>a5bac82a92eadb58f027754cb2d64231602d218c</foaf:mbox_sha1sum>
<rdfs:seeAlso rdf:resource="http://gonze.com/foaf.rdf" />
/foaf:Person>
<foaf:Person rdf:ID="ggershman">
<foaf:name>Gregory Gershman</foaf:name>
<foaf:mbox_sha1sum>3ecadabf6fe71fa67e2838de2bf9c2372b6c72d6</foaf:mbox_sha1sum>
<foaf:homepage rdf:resource="http://www.blogdigger.com/blog/"/>
```

```
<rdfs:seeAlso rdf:resource="http://www.blogdigger.com/blog/foaf.rdf" />
</foaf:Person>

<!--  This is a blank person template. Using it is easy, follow these steps
                invent an ID string you'll use to refer to this person,
                like joeuser or jqpublic or something that will only get used in this file
                enter than into the foaf:Person rdf:ID string
                Put the person's display name into the foaf:name element, like "Joe User"
                Enter their first name and last name strings.
                Make a hash of the person's email address.  Do NOT put their e-mail address in here
                To make a hash, use my generator at http://feeds.archive.org/misc/hash
                If they have a foaf file of their own then put the URL of it into the rdf:seeAlso rdf:resource string
                Do NOT put someone's un-hashed e-mail address in your Foaf file.
                Only put that in your own Foaf file and then only if you explicity want to make it public

                <foaf:Person rdf:ID="">
                <foaf:name></foaf:name>
                <foaf:firstname></foaf:name>
                <foaf:surname></foaf:name>
                <foaf:mbox_sha1sum></foaf:mbox_sha1sum>
                <rdfs:seeAlso rdf:resource="" />
                </foaf:Person>
                -->

<rss:channel rdf:about="http://www.syndic8.com/~wkearney/blogs/syndic8/xml/index.rdf">
<dc:title>Syndication News</dc:title>
<dc:description xml:lang="en">Recent news in XML syndication</dc:description>
<dc:language xml:lang="en">en</dc:language>
<dc:creator rdf:resource="#wkearney99" />
</rss:channel>

<rdf:Description rdf:about="http://www.ideaspace.net/users/wkearney/images/109-0913_IMG-SM.JPG">
<dc:title>Bill Kearney</dc:title>
<dc:description>Head shot picture of Bill Kearney</dc:description>
<dc:creator rdf:resource="#wkearney99" />
<dc:date>2001-10</dc:date>
</rdf:Description>

<rdf:Description rdf:about="http://www.ideaspace.net/users/wkearney/images/with_cerf_sm.JPG">
<dc:title>Vinton Cerf and Bill Kearney</dc:title>
<dc:description>Picture of Vinton Cerf and Bill Kearney at NOVAlug meeting</dc:description>
<dc:creator rdf:resource="#wkearney99" />
<dc:date>2002-10</dc:date>
<dc:coverage>Reston, VA USA</dc:coverage>
<foaf:location><foaf:Geo rdf:value="Reston, VA USA" /></foaf:location>
/rdf:Description>
```

```
<rdf:Description rdf:about="http://www.syndic8.com/~wkearney/blogs/syndic8">
<dc:title>Syndication News from Bill Kearney</dc:title>
</rdf:Description>

<rdf:Description rdf:about="http://www.ideaspace.net/users/wkearney/index.rdf">
<dc:title>ideaspace</dc:title>
</rdf:Description>

<wot:PubKey wot:hex_id="79F14C94" wot:length="1024" >
<wot:fingerprint>2868 8C59 FBEA FABE 123D 1C69 FDBB AC3D 79F1 4C94</wot:fingerprint>
<wot:identity>
<wot:User>
<foaf:Person rdf:resource="#wkearney99" />
<wot:signed>
<wot:SigEvent>
<foaf:Person rdf:resource="#wkearney99" />
<wot:sigdate>2002-07-31</wot:sigdate>
<wot:signerLongHexID>3E15EF2F73228FE4</wot:signerLongHexID>
</wot:SigEvent>
</wot:signed>
</wot:User>
</wot:identity>
</wot:PubKey>

<!--            Read Edd's page for more info: http://usefulinc.com/foaf/signingFoafFiles          -->

<!--            On my redhat linux box it's just "gpg -a -detach-sign foaf.xrdf"                  -->
<!--              replace the single dash in "-detach-sign" with two leading dashes  -->
<!--            To use pgp the command line is "pgp -sba foaf.xrdf"                               -->

<rdf:Description rdf:about="">
<!-- This 'blank' rdf:about section contains metadata that describes THIS very document -->
<dc:creator rdf:resource="#wkearney99" />     <!-- created by me -->
<dc:date>2002-05-01</dc:date>                 <!-- timestamp of it's creation -->
<dcterms:modified>2003-10-22</dcterms:modified>          <!-- timestamp of it's last update -->

<foaf:maker rdf:resource="#wkearney"/>
<!-- digital signature for this file -->
<!-- wot:assurance rdf:resource="http://www.ideaspace.net/users/wkearney/foaf.xrdf.asc" -->
</rdf:Description>
</rdf:RDF>
```